Kernel Proposal Network for Arbitrary Shape Text Detection

Shi-Xue Zhang, Xiaobin Zhu, Jie-Bo Hou, Chun Yang, and Xu-Cheng Yin, *Senior Member, IEEE*

***Abstract*— Segmentation-based methods have achieved great success for arbitrary shape text detection. However, separating neighboring text instances is still one of the most challenging problems due to the complexity of texts in scene images. In this article, we propose an innovative kernel proposal network (dubbed KPN) for arbitrary shape text detection. The proposed KPN can separate neighboring text instances by classifying different texts into instance-independent feature maps, meanwhile avoiding the complex aggregation process existing in segmentation-based arbitrary shape text detection methods. To be concrete, our KPN will predict a Gaussian center map for each text image, which will be used to extract a series of candidate kernel proposals (i.e., dynamic convolution kernel) from the embedding feature maps according to their corresponding keypoint positions. To enforce the independence between kernel proposals, we propose a novel orthogonal learning loss (OLL) via orthogonal constraints. Specifically, our kernel proposals contain important self-information learned by network and location information by position embedding. Finally, kernel proposals will individually convolve all embedding feature maps for generating individual embedded maps of text instances. In this way, our KPN can effectively separate neighboring text instances and improve the robustness against unclear boundaries. To the best of our knowledge, our work is the first to introduce the dynamic convolution kernel strategy to efficiently and effectively tackle the adhesion problem of neighboring text instances in text detection. Experimental results on challenging datasets verify the impressive performance and efficiency of our method. The code and model are available at https://github.com/GXYM/KPN.**

***Index Terms*— Arbitrary shape text detection, deep neural network, dynamic convolution kernel, kernel proposal.**

Manuscript received July 21, 2021; revised December 2, 2021 and January 14, 2022; accepted February 14, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA09701; in part by the National Science Fund for Distinguished Young Scholars under Grant 62125601; and in part by the National Natural Science Foundation of China under Grant 62076024, Grant 62172035, Grant 62006018, and Grant 61806017. *(Shi-Xue Zhang and Jie-Bo Hou contributed equally to this work.) (Corresponding author: Xiaobin Zhu.)*

Shi-Xue Zhang, Xiaobin Zhu, Jie-Bo Hou, and Chun Yang are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: zhangshixue111@163.com; zhuxiaobin@ustb.edu.cn; houjiebo@gmail.com; chunyang@ustb.edu.cn).

Xu-Cheng Yin is with the School of Computer and Communication Engineering, and Institute of Artificial Intelligence, University of Science and Technology Beijing, Beijing 100083, China, and also with USTB–EEasyTech Joint Laboratory of Artificial Intelligence, Beijing 100083, China (e-mail: xuchengyin@ustb.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3152596.

Digital Object Identifier 10.1109/TNNLS.2022.3152596

I. Introduction

SCENE text detection is a fundamental and critical task in computer vision because it is a key step in various text-related applications, including translation, text-visual question answering, text recognition, and text mining. With the rapid development of deep learning-based object detection [1]–[3] and segmentation [4], [5], scene text detection has witnessed great progress [6]–[8]. Arbitrary shape scene text detection, as one of the most challenging tasks in text detection, has attracted ever-increasing interest in both research and industrial communities. Except for the challenges existing in the general scene text detection tasks, arbitrary shape text detection should address additional challenging problems, such as varied scales, curved, and arbitrary shapes.

Recently, segmentation-based methods [9]–[11] have achieved promising performance in detecting arbitrary shape text. Benefiting from the adaptability of pixel-level prediction (e.g., pixel-wise text/nontext classification mask), segmentation-based methods [10], [12], [13] based on fully convolutional neural networks (FCNs) can easily adapt to irregular texts. However, segmentation-based text detection methods always suffer from separating neighboring text instances in complex scene images, as shown at the top of Fig. 1(a), which may be caused by inaccurate annotations or similar appearances between neighboring text instances. To solve this problem, many segmentation-based methods [9]–[11], [13], [14] adopt complicated postprocessing operations to the group and separate text instances based on their predicted masks. PSENet [13] adopts kernels with different scales for each text instance and gradually expands the minimal scale kernel to reconstruct individual text instances. The other methods [9]–[11], [14] try to learn pixel-pair embedding vectors for clustering text pixels into different text instances during postprocessing. Although the adhesion problem can be alleviated to some extent in [9]-[11], [13], and [14], they all suffer from the formidable computational cost in complex postprocessing. To improve model efficiency, DB [12] abandons complex postprocessing and shrinks annotated boundaries with the Vatti clipping algorithm [15] to obtain probability maps. Then, the probability maps will be used to restore separated boundaries for text instances by the inverse transformation of the Vatti clipping algorithm. Unfortunately, the text boundaries generated by fixed scaling rules in DB [12] tend to be inaccurate on text with different scales. Besides, some mask region-based convolutional neural

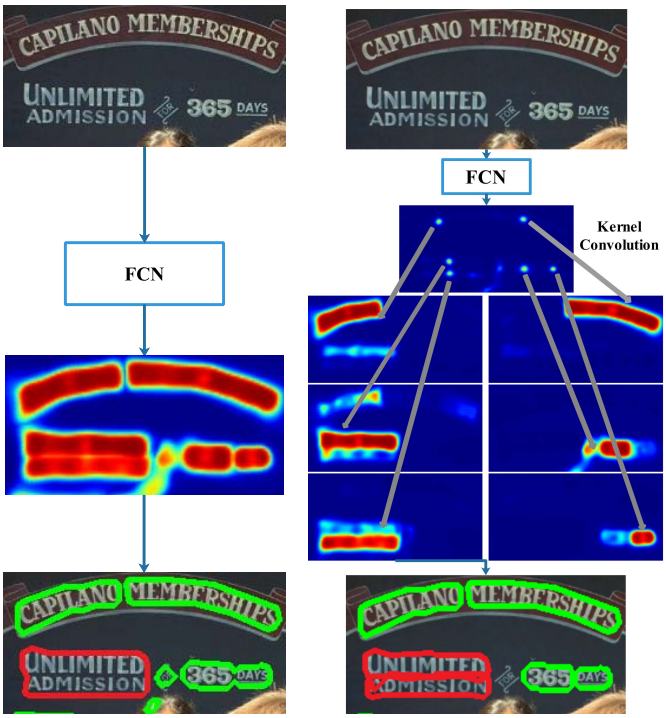


Fig. 1. Comparison with FCN-based methods. (a) FCN-based methods suffer from covering adjacent instances (red contour in the bottom image). (b) Demonstrates that our KPN can successfully disperse texts on feature maps, thus generating separated yet accurate masks for text instances.

network (R-CNN) [16]-based methods [17]–[19] first detect bounding rectangles of candidate texts, followed by pixel-wisely segmenting text instances in their corresponding boxes. Although Mask R-CNN-based methods can alleviate the false detection of covering adjacent instances, their performance greatly relies on the extracted bounding boxes' quality and computational-cost RoI operation.

In this article, we propose an innovative kernel proposal network (dubbed KPN) for arbitrary shape text detection. The proposed KPN can separate neighboring text instances by classifying different texts into instance-independent feature maps meanwhile avoiding the complex aggregation process existing in segmentation-based arbitrary shape text detection methods. To be concrete, our KPN will predict a Gaussian center map for each text image, which will be used to extract a series of candidate kernel proposals (i.e., dynamic convolution kernel) from the embedding feature maps according to their corresponding keypoint positions. However, the premise for accurately separating neighboring text is that these kernel proposals should be independent. To enforce the independence between kernel proposals, we propose a novel orthogonal learning loss (OLL) via orthogonal constraints. Specifically, our kernel proposals mainly contain important self-information learned by the network and position information by position embedding, instead of the shared information of all texts. After removing the noise kernel proposals via predefined rules, the remaining ones will individually convolve all embedding feature maps for classifying individual texts into instance-independent maps. In this way, our KPN can effectively separate neighboring text instances and improve the robustness against unclear boundaries. To the best of our knowledge, our work is the first to introduce the dynamic convolution kernel strategy to efficiently and effectively tackle the adhesion problem of neighboring text instances in text detection.

In summary, our main contributions are fourfold.

1) We develop a postprocessing-free segmentation-based arbitrary shape text detection framework that cleverly avoids computational-cost postprocessing for achieving effectiveness and efficiency.
2) We propose a novel dynamic convolution kernel strategy for text detection in which kernel proposals contain important self-information and position information, resulting in the effectiveness of separating neighboring texts and improving the robustness against tiny intervals or unclear boundaries.
3) We propose a novel OLL that directly enforces the independence between kernel proposals via orthogonal constraints.
4) Experiments conducted on publicly available datasets demonstrate the effectiveness and efficiency of the proposed method.

The rest of this article is organized as follows: Section II overviews the related work. Section III elaborates our method. In Section IV, we demonstrate experimental results on several datasets. Finally, we conclude our work in Section V.

II. Related Work

A. Regression-Based Methods

Regression-based methods generally predict text boxes by regressing offsets of bounding rectangles based on predefined anchors or original pixels, which can be roughly divided into two categories: anchor-based methods and anchor-free methods. Most of the anchor-based methods try to design or learn appropriate anchors for accurately regressing text boxes. Both TextBoxes [20] and TextBoxes++ [21] adopt a series of anchors with different aspect ratios for covering texts with varied lengths. Specifically, TextBoxes++ regresses offsets of quadrilateral four points by predefined anchors for adapting to arbitrarily oriented texts. RRPN [7] adopts rotated anchors (three scales, three ratios, and six angles) and rotated RoI pooling (RRoI pooling) for arbitrarily oriented text detection. Anchor-free methods [6], [22] try to regress texts without predefined anchors for improving model adaptability. As a classical anchor-free method, EAST [6] adopts an FCN [4] branch for classification, and a regression branch for directly regressing bounding quadrilateral via IoU loss [23]. HAM [24] proposes a hidden anchor mechanism to integrate the advantages of the anchor-based method into the anchor-free method. However, the regression distance and angle in most regression-based methods are strictly confined to quadrilateral text, making it challenging to handle arbitrary shapes.

B. Connected Component-Based Methods

Component-based detection [25], [26] is also an important direction in the field of object detection. In scene text detection, connected component-based (CC-based) methods [8], [26]–[30] usually detect individual text parts or characters first,

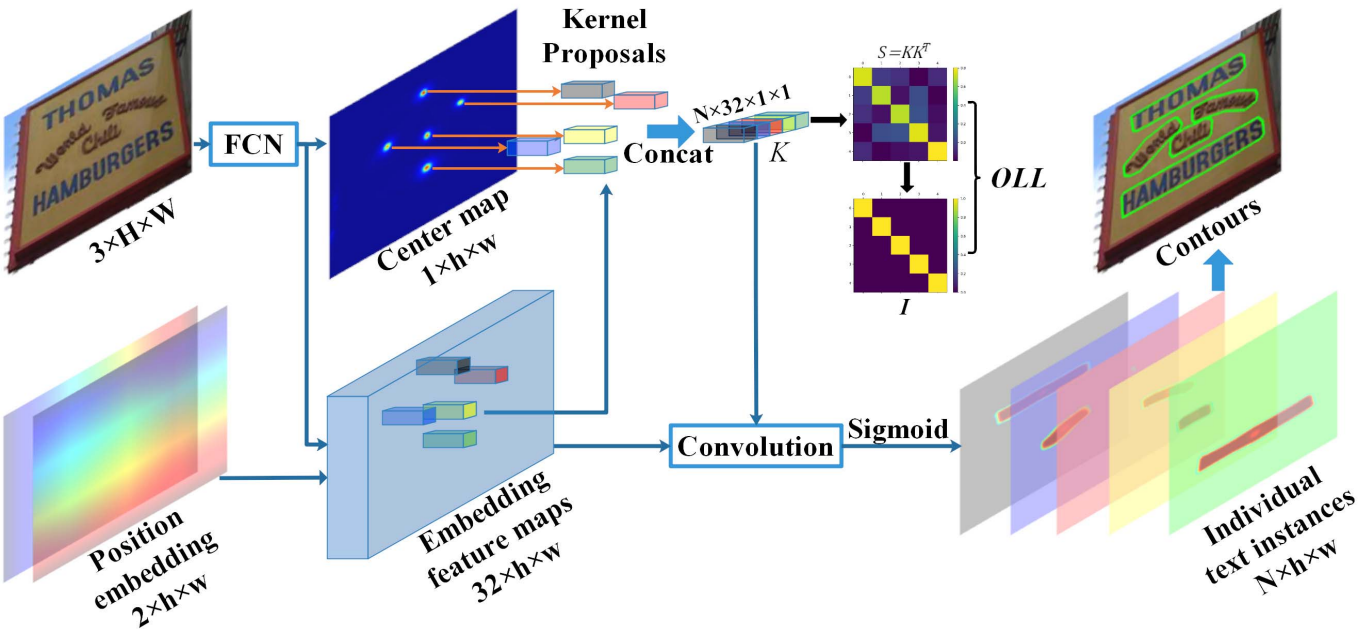


Fig. 2. Framework of our method. The feature extractor FCN is detailed in Fig. 3. The network will dynamically predict N (N **is not fixed**) key points and get the corresponding embedding vectors for generating **kernel proposals**. To enforce the independence between kernel proposals, we propose a novel OLL via orthogonal constraints to make $S = KK^T$ close to an identity matrix (I). Finally, we predict text masks for generating contours. Examples of key point generation: (a) predicted heat maps; (b) CCs after thresholding; and (c) key points that with the scores in each CC.

followed by postprocessing of link or group for generating final texts. Connectionist text proposal network (CTPN) [26] modifies the framework of faster R-CNN [1] to extract horizontal text components of fixed width for connecting dense text components and generating horizontal text lines. SegLink [27] decomposes each text into two detectable elements, i.e., segment and link, where a link indicates that a pair of adjacent segments belong to the same text. Character region awareness for text detection (CRAFT) [29] detects text regions by exploring affinities between characters. TextDragon [30] first detects local bounding boxes of texts and then groups them into different text instances via their geometric relations. Zhang *et al.* [8] adopted a graph convolution neural network (GCN) to learn and infer linking relationships between different text components for grouping them into final text instances. Although CC-based methods can learn a flexible representation for adapting to arbitrary shape text, the complex postprocessing for grouping text components is always time-consuming.

C. Contour-Based Methods

Contour-based [31]–[37] try to directly model the text boundary for detecting the arbitrary shape text. ABCNet [33] and FCENet [34] model contours of text instances with curve modeling (Bezier-curve and Fourier-curve), which can well fit closed contour with progressive approximation. TextRay [35] formulates text contours in the polar system and proposes a single-shot anchor-free framework to predict geometric parameters and output simple polygon detections. PCR [36] proposes a progressive contour regression framework to detect arbitrary-shape scene texts. TextBPN [37] proposes an adaptive boundary deformation model to perform boundary deformation iteratively. However, compared with segmentation-based methods, the performance and efficiency of the contour-based methods are unsatisfactory.

D. Segmentation-Based Methods

Segmentation-based methods can get the contours of text instances from segmentation masks. In [17], [19], and [38], they first predict the bounding boxes of texts and then segment the pixelwise mask of text in each box. Some other methods [9], [11], [14] utilize FCN [4] to predict text masks and predict extra embedding vectors to cluster pixels in text masks. PixelLink [9] predicts eight linkages to judge the connectivity and then link pixels using a disjoint-set data structure. TextField [14] adopts a direction field to group pixels with morphological postprocessing. PSENet [13] shrinks the text region into various scales for generating more distinct boundaries and then gradually expands the minimal scale kernel to the text instance with the complete shape. Tian *et al.* [11] assumed each text instance as a cluster and predicted an embedding map via pixel clustering. Overall, segmentation-based methods can easily adapt to texts in arbitrary shapes. However, the existing methods still struggle with high complex postprocessing to cluster pixels into texts instances.

E. Dynamic Kernel Methods

In the instance segmentation task, AdaptIS [39] iteratively predict point proposals to generate instances. At each iteration, AdaptIS picks one point proposal and then uses the AdaIN mechanism [40] to generate a corresponding instance. By providing different parameters to AdaIN, AdaptIS can vary the network output for the same input. However, AdaptIS predicts instances in inference iteratively with low efficiency. Based on fully convolutional one-stage (FCOS) [41], CondInst [42] use the conditional convolution

(i.e., dynamic convolution) to generate instance-sensitive filters to encode object instance information. SOLOv2 [43] generates and separates instances simultaneously. More specifically, it dynamically proposes $s \times s$ (the input image is divided into $s \times s$ grids) kernels to convolve the feature maps for generating $s \times s$ instances in $s \times s$ channels. Therefore, if we set the "resolution" ($s \times s$) of kernels in SOLOv2 to $w \times h$ (width and height of feature maps) as the same as the general "resolution" in scene test detection, the memory cost of SOLOv2 in feature maps will be $(w \times h)^2$, which is computation and memory formidable. Moreover, CondInst and SOLOv2 still rely on time-consuming non-maximum suppression (NMS) for removing abundant duplicate boxes.

III. Our Method

A. Overwiew

As shown in Fig. 2, our method mainly consists of four components: feature extraction, kernel proposal, instance-independent feature map extraction, and contour generation. We utilize an FCN [4] combined with a feature pyramid network (FPN) [2] to extract informative feature maps. The architecture details of our feature extraction subnetwork are elaborated in Fig. 3. To extract kernel proposals, we first get the CC of each text in the predicted center map and select the pixel with the highest score in each component as the key point. The embedding feature maps in the corresponding position of key points are predicted kernel proposals, which will be convolved by the predicted kernel proposals for generating instance-independent feature maps. The predicted feature maps will be binarized by a predefined threshold to get the contours of the detected texts.

B. Kernel Proposal

For segmenting text instances in arbitrary shapes, a majority of existing methods adopted the popular FCN structure. However, they often suffer from texts with tiny intervals or unclear boundaries, as shown in Fig. 1(a). To overcome these problems, some methods adopt embedding strategies to link [9], [14] or cluster [10], [11] pixels into different text instances. The success of these methods lies in finding a metric function $F(*,*)$ to judge if two pixels belong to the same text, which can be formulated as

$$F(e_i, e_j)\begin{cases} 1, & T(p_i) = T(p_j) \\ 0, & T(p_i) \neq T(p_j) \end{cases} \tag{1}$$

where p_i and p_j denote the pixels in position i and j, respectively; $T(*)$ denotes the corresponding text of the pixel; e_i and e_j denote the embedding features of p_i and p_j, respectively; $F(*,*)$ is a learnable function.

From our observation, if we find an appropriate $F(*,*)$ as a binary classifier, we can classify the pixels of different text instances into instance-independent feature maps. Fortunately, it can be well implemented by a convolution layer together with a sigmoid function. Specifically, we can find a suitable convolution kernel (i.e., dynamic kernel) for one text instance, which can convolve all the embedding feature maps to classify the pixels belonging to this text instance to an instance-independent feature map. It can help us segment each text instance efficiently.

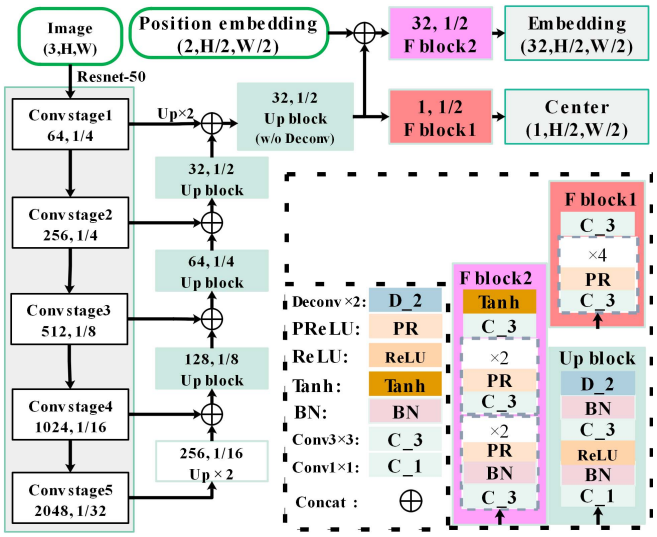


Fig. 3. Feature extraction subnetwork has two output branches: one channel for center map and the other 32 channels for embedding feature maps.

As mentioned earlier, we try to utilize a group of convolution kernels for classifying different text instances into instance-independent feature maps. In traditional convolution methods, the number of kernels is fixed, making the channels of output feature maps are also fixed. However, the number of text instances often varies in different scene images. Thus, the specified number of instance-independent feature maps may induce missed or repeated detection. Inspired by dynamic kernel methods [39], [43], we propose a dynamic convolution kernels strategy (kernel proposal) to separate text instances into different instance-independent feature maps. Specifically, we first predict N key points for N text instances, then extract the feature maps according to their corresponding positions to generate kernel proposals (i.e., dynamic convolution kernels). Different from other dynamic kernel methods, our kernel proposals are orthogonal to each other, which are a set of orthogonal basis vectors. In this design, each kernel proposal contains the self-information and position information of its own text instance, not the information of other text instances. The kernel proposals will convolve the embedding feature maps to generate individual feature maps for each text instance, which is formulated as

$$O = K * E = \begin{bmatrix} k_0 \\ \dots \\ k_i \\ \dots \\ k_N \end{bmatrix} * E = \begin{bmatrix} p_0 \\ \dots \\ p_i \\ \dots \\ p_N \end{bmatrix} \tag{2}$$

where O represents the output feature maps of, in which each channel corresponds to prediction (p_i) of one text; k_i denotes the ith kernel proposal. The convolution operation $*$ decomposes the high-dimensional vector E onto the orthogonal basis K. E denotes the embedding feature maps which contains the shared features extracted by backbone (F_s) and position embedding feature (F_p) as

$$E = F\text{block}_2(F_s \oplus F_p) \tag{3}$$

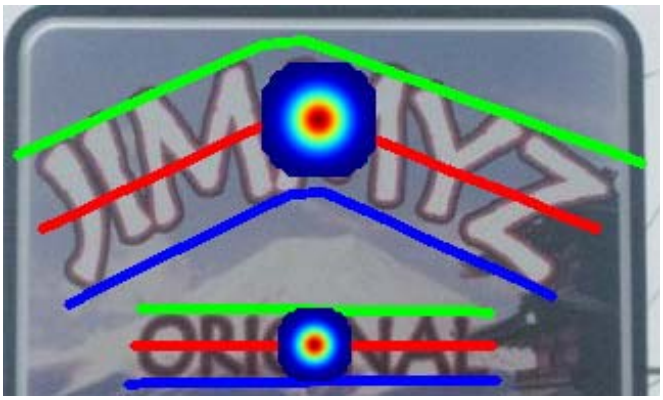


Fig. 4. Overview of key points labeling: top lines (green lines), bottom lines (blue lines), center lines (red lines), and center Gaussian heat maps for text instances.

where $\oplus$ donates a concatenate operation, as shown in Fig. 3. After full training, F_s is mainly contains the features of text instances, and the nontext features tend to 0.

C. Key Points Generation

If we directly utilize the function $F(*,*)$ in 1 to classify all pixel pairs, the computational complexity will be $(w \times h)^2$. To reduce the computational complexity, we only select one key point from the predicted Gaussian center maps as kernel proposals for each text instance, where one Gaussian center may correspond to one text instance. These kernel proposals contain important self-information learned by the network and key point location information by position embedding, which is the basis of our method to efficiently separate neighboring texts.

Unlike objects in general instance segmentation, text instances do not have a closed boundary, which means that it contains a lot of regions similar to the background. It may cause our method to be sensitive to the quality of key points. To solve this problem, we adopt a broad key point for text instance. Specifically, we designate the representative center point of text instance as a center key point. As shown in Fig. 4, we follow TextSnake [44] to find the top line (green line) and the bottom line (blue line) of text, then calculate the centerline (red line) and extract the middle point in the centerline as the center point. Apparently, it is challenging to predict one single accurate center point. Thus, we adopt the Gaussian heat-map strategy as in [45] and [46], as shown in Fig. 4. The radius is the minimum distance of the center point to the boundary.

In key point labeling, we will get a heat map, as shown in Fig. 5(a). However, there still contain many redundant points for a text instance. Thus, we compute the CC of each center via thresholding [Fig. 5(b)]. We will select the point with the highest score in each CC as the final predicted key point. Kernel proposals are extracted from the embedding feature maps corresponding to the selected key points.

D. Position Embedding

Our feature extraction subnetwork constructed on the popular FCN [4] (RestNet-50 [47] as backbone) and FPN [2] in the text detection domain. However, existing segmentation-based methods via conventional convolutional pixel embeddings cannot easily distinguish identical copies of an instance [48]. Inspired by [11], [43], [48], and [49], we also introduce a position embedding strategy in our network for keeping location information of key points, as shown in Fig. 2. Specifically, we adopt two channels of feature maps by embedding x**-axis** and y**-axis** position information for every pixel.

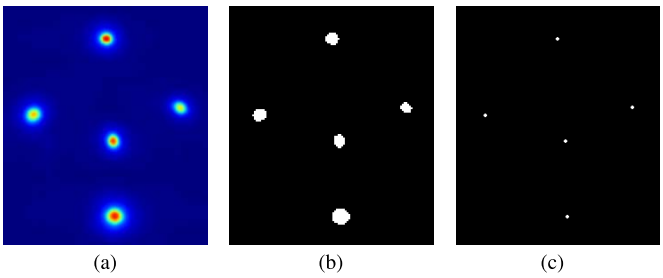


Fig. 5. Examples of key point generation: (a) predicted heat maps; (b) CCs after thresholding; (c) key points that with the scores in each CC.

In addition, we normalize the value of position embedding to range $[-1, 1]$, i.e., the pixel in x-axis 0 is set to -1, and the pixel in x-axis is set to 1. Mathematically, the position embedding of x_i and y_i can be formulated as

$$x_i = \left\{-1 + \frac{2i}{w-1} | i \in (0, w-1)\right\} \tag{4}$$

$$y_i = \left\{-1 + \frac{2i}{h-1} | i \in (0, h-1)\right\} \tag{5}$$

where w and h, respectively, represent the width and height of the output feature maps.

E. Loss Function

We optimize the prediction of the Gaussian center map and the segmentation of each instance with the following loss function:

$$L_c(y, \hat{y}, \alpha) = \alpha * L_{\text{dice}}(y, \hat{y}) + L_{\text{focal}}(y, \hat{y}) + L_{\text{OHEM}}(y, \hat{y}) + L_{\text{BBCE}}(y, \hat{y}) \tag{6}$$

where L_{dice} denotes dice loss [50]; L_{focal} denotes focal loss [51]; L_{OHEM} denotes cross entry loss with online hard example mining (OHEM) [52], in which the ratio between negative and positive pixels is 3:1; L_{BBCE} denotes balance binary cross entropy loss; y denotes a prediction and $\hat{y}$ is its ground-truth; and α is used to select L_{dice}. For each instance, the loss function of Gaussian center map ($\mathcal{L}_c^{\text{gc}}$) and the loss function of the segmentation ($\mathcal{L}_c^s$) are, respectively, defined as

$$\mathcal{L}_c^{\text{gc}} = L_c(y, \hat{y}, 0) \tag{7}$$

$$\mathcal{L}_c^s = L_c(y, \hat{y}, 1). \tag{8}$$

To separate different text instances effectively, we first need to restrict the kernel proposals to be independent of each other in one image. In this way, the kernel proposals only include the main features of their own text, not the shared features of texts. Consequently, by individually convolving all embedding feature maps with kernel proposals, our method can generate unique feature maps for individual text instances.

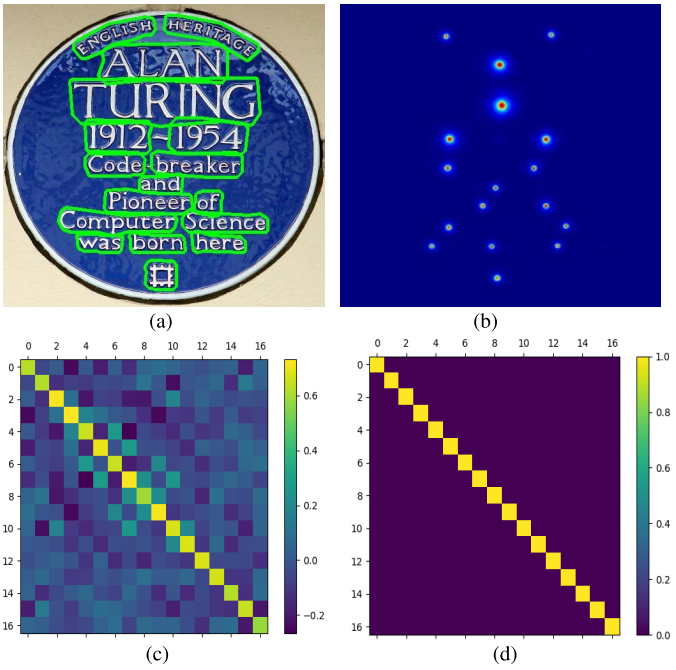


Fig. 6. (a) Detected contours in the image. (b) Prediction of Gaussian center map. (c) Similarity matrix (S) between kernel proposals. (d) I is an identity matrix.

In the same image, the similarity matrix between kernel proposals can be expressed as

$$S = KK^{\mathrm{T}} = \begin{bmatrix} k0 \\ \cdots \\ k_i \\ \cdots \\ k_N \end{bmatrix} \times \begin{bmatrix} k0 \\ \cdots \\ k_i \\ \cdots \\ k_N \end{bmatrix}^{\mathrm{T}} \tag{9}$$

where K is the set of kernel proposals in one image; S is a similarity matrix ($N \times N$) for these kernel proposals, as shown in Fig. 6(c); and N is the number of kernel proposals. To constrain the kernel proposals to be independent of each other, we propose an OLL (L_{OLL}) as

$$\mathcal{L}_{\mathrm{OLL}} = L_{\mathrm{dice}}(S, I) + L_{\mathrm{BCE}}(S, I) \tag{10}$$

where I is an identity matrix, as shown in Fig. 6(d); L_{dice} denotes dice loss; and L_{BCE} denotes binary cross entropy loss. Dice coefficient is usually used to calculate the similarity of two samples. L_{BCE} can also be used to constrain the similarity between samples to approach orthogonal relationships.

Finally, the total loss of our method can be formulated as

$$\mathcal{L} = \mathcal{L}_c^{\mathrm{gc}} + \mathcal{L}_c^s + \lambda_O * \mathcal{L}_{\mathrm{OLL}} \tag{11}$$

where λ_O is set to 0.1.

IV. Experiments

A. Datasets

To verify the effectiveness of the proposed KPN, we conduct experiments on three benchmark datasets: Total-Text [53], CTW-1500 [54], and International Conference on Document Analysis and Recognition (ICDAR) 2015 [55].

Total-Text [53] consists of 1255 training and 300 testing images. It is collected from various scenes, including text-like background clutter and low-contrast texts, and are word-level annotated by polygons for curve text detection task.

CTW-1500 [54] consists of 1000 training and 500 testing images. It contains both English and Chinese texts with text-line level polygon annotations for curved text detection.

ICDAR 2015 [55] consists of 1000 training and 500 testing images. It is a multi-orientated and street-viewed dataset collected for the arbitrary-oriented text detection task. The annotations are word level with four vertices.

B. Implementation Details

The pretrained ResNet-50 [47] is adopted as the backbone of our network. We adopt Adam [56] for optimizing our method and the initial learning rate is 0.0001 and will be decayed by 0.9 per 100 epochs. Our network is first pretrained on the large dataset MLT 2017 [57] and randomly crop images into 640×640, 832×832, and 1024×1024, respectively. Then, we will fine-tune our network on the target benchmark dataset with 832×832 crop images. Following DRRG [8], we also adopt the general augmentation tricks, including crops, rotations, color variations, and partial flipping.

In the training, we utilize two ways to get the key points for training kernel proposals: 1) we randomly sample one key point in each text instance. The sampling probability is computed on a Gaussian heat map, as described in Section III-C. 2) We pick top-k ($k = 50$) points in Gaussian heat map for text instance, as in Fig. 5. In testing, we select only one point of the highest score as kernel proposal for a corresponding text instance (CC), as shown in Fig. 5. Our KPN contains two hyperparameters: the threshold filtering center heat map thresh_c, and the threshold for filtering final text instances thresh_i. All experiments are performed on CPU (Intel Xeon E5-2620 v4 @ 2.10 GHz), GPU (GTX 1080Ti 11G), and PyTorch 1.2.0.

C. Ablation Study

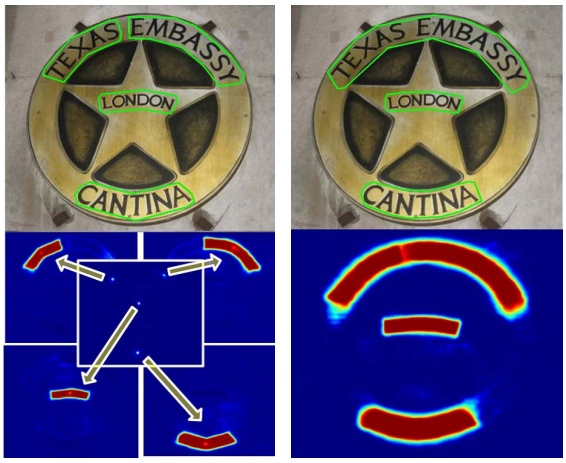
To verify our method, we conduct experiments on images with different scales. All the images will be resized into the range of [**short**, **large**]. In this section, we only train our models on Total-Text with 300 epochs and adopt Adam [56] as optimizer.

1) KPN vs. FCN: To verify the effectiveness of our KPN, we compare our KPN with an FCN-based method, which is similar to our feature extraction subnetwork in Fig. 3. The FCN-based method only utilizes the center branch to learn the masks of texts without the embedding branch. For fair comparisons, we train and evaluate our KPN and the FCN-based method both on Total-Text with 300 epochs. As listed in Table I, our KPN outperforms FCN on Total-Text nearly without sacrificing efficiency [frames per second (FPS)]. The representative detection results are shown in Fig. 7.

2) Center Point Versus Center Region: The shrinking center region of text is a popular strategy in segmentation-based methods [11], [12]. Here, we conduct an ablation study to compare our center point strategy with the center region strategy. As listed in Table II, we can find that the center point strategy outperforms the center region strategy with a significant margin. For example, at 1024 resolution, our

TABLE I
ABLATION STUDY OF KPN AND FCN ON TOTAL-TEXT

Method	Recall	Precision	H-means	FPS
FCN-640	62.65	66.58	64.55	30.36
KPN-640	**65.48**	**83.19**	**73.28**	23.15
FCN-832	66.56	65.40	65.98	21.43
KPN-832	**71.26**	**84.58**	**77.35**	15.17
FCN-1024	64.16	67.99	66.02	15.27
KPN-1024	**74.55**	**85.00**	**79.43**	10.54



(a) (b)

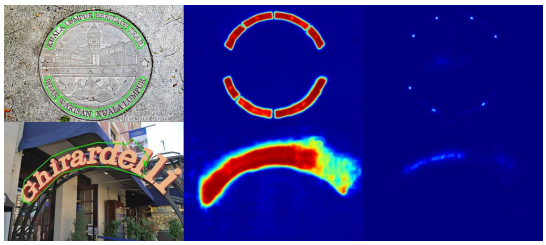
Fig. 7. Representative detecting results of KPN and FCN. (a) KPN. (b) FCN.

TABLE II
ABLATION STUDY OF CENTER REGION (C_r), CENTER POINT (C_p), POSITION EMBEDDING (P_s), AND TEST IMAGE SCALE ON TOTAL-TEXT (ONLY TRAINED ON TOTAL-TEXT WITH 300 EPOCHS). "R," "P," AND "H" REPRESENT "RECALL," "PRECISION," AND "H-MEAN," RESPECTIVELY

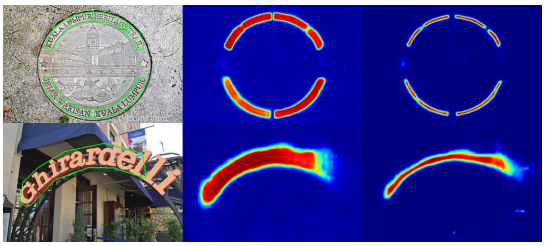
Scale	C_r	C_p	P_e	R	P	H	FPS
KPN-640	✓	×	✓	59.36	71.53	64.88	22.15
KPN-640	×	✓	✓	**65.48**	**83.19**	**73.28**	**23.15**
KPN-832	✓	×	✓	66.22	71.88	68.93	14.39
KPN-832	×	✓	×	69.64	82.13	75.53	**15.84**
KPN-832	×	✓	✓	**71.26**	**84.58**	**77.35**	15.17
KPN-1024	✓	×	✓	68.27	71.76	69.97	9.76
KPN-1024	×	✓	×	73.35	83.54	78.11	**10.88**
KPN-1024	×	✓	✓	**74.55**	**85.00**	**79.43**	10.54

method outperforms the FCN by 13.41% in terms of H-means (KPN-1024 79.43% versus FCN-1024 66.02%). As shown in Fig. 8, we can find that our center point strategy can accurately separate adjacent text instances. But center region strategy tends to suffer from adjacent and overlapping of candidate text regions.

3) Influence of Position Embedding: We conduct ablation studies on Total-Text to verify the effectiveness of position embedding (P_s). As listed in Table II, without position embedding (P_s), the performance of the proposed KPN will decrease by about 1% to 2% in terms of H-means. Generally speaking, location information is a fundamental basis for separating neighboring text, especially when they have a similar scale and appearance. Equipped with position embedding, the kernel



(a)



(b)

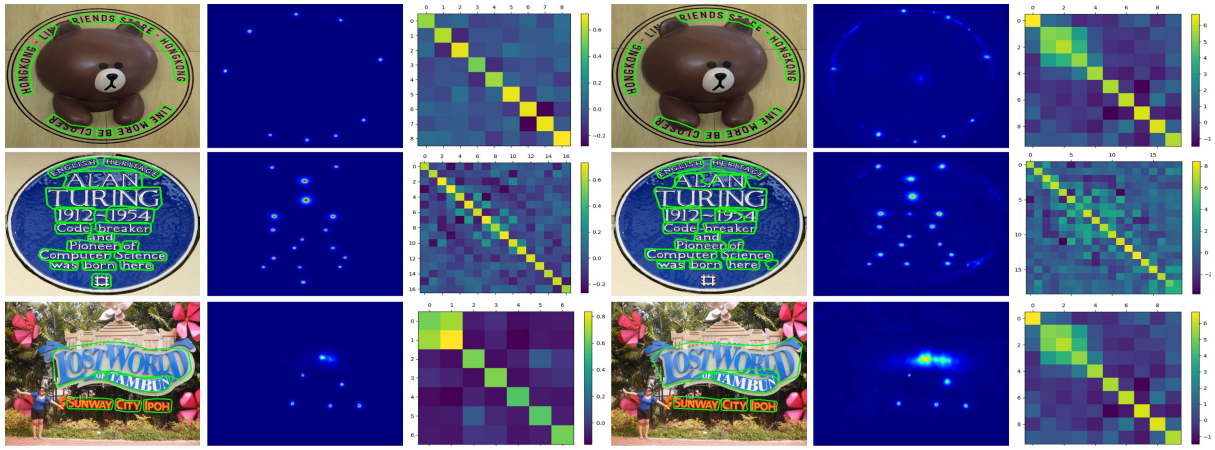
Fig. 8. Left column: detection results; middle column: sum of all the masks in KPN; right column: center points/regions. (a) Center point. (b) Center region.

TABLE III
ABLATION STUDY OF OLL

Scale	OLL	Recall	Precision	H-means
KPN-640	×	**67.86**	73.97	70.78
KPN-640	✓	65.48	**83.19**	**73.28**
KPN-832	×	**71.62**	75.20	73.36
KPN-832	✓	71.26	**84.58**	**77.35**

proposal and dynamic convolution kernel generated in our KPN will contain the central position information of the corresponding text instance.

4) Influence of Orthogonal Learning Loss: We conduct ablation studies on Total-Text to verify the influence of OLL. As listed in Table III, the OLL greatly improve the detection performance (by 2.5% in KPN-640 and 3.99% in KPN-832 in terms of H-means). In Fig. 9, we also show some visualized results of similarity matrix (S) with or without OLL in model training. From Fig. 9, we can find that similarity matrix (S) approaches an identity matrix (I) with OLL for training, and the detection contours are more accurate. In contrast, the false detection and overlap detection apparently increase without the help of OLL, as shown in the second row of Fig. 9(d). This is why the recall (R) of detection is high without OLL, as listed in Table III. In a few cases, there will be one text with multiple kernel proposals. As shown in the third row of Fig. 9(b) and (e), there are two kernel proposals for text instance "LOSTWORLD" because two centers are predicted for it. In this case, these two kernel proposals will have high similarity, as shown in the third row of Fig. 9(c) and (f). Therefore, the same text instance will be predicted if we use these two kernel proposals to convolve embedding feature maps. With the independence restriction via orthogonal constraints, the kernel proposals can only contain important self-information and position information of its text, guaranteeing the effectiveness of classifying different texts into instance-independent maps.



(a) (b) (c) (d) (e) (f)

Fig. 9. Representative visual results on Total-Text dataset with or without OLL. * means OLL is not adopted in training. (a) Detected contour. (b) Gaussian center map. (c) S. (d) Detected contour*. (e) Gaussian center map*. (f) S*.

TABLE IV

EXPERIMENTAL RESULTS ON TOTAL-TEXT. * AND "MS" REPRESENTS MULTISCALE TEST AND † DENOTES THE TEXTSPOTTER-BASED METHODS

Methods	Paper	Ext	R	P	H	FPS
TextSnake [44]	ECCV'18	Syn	74.5	82.7	78.4	-
FTSN [17]	ICPR'18	Syn	78.0	84.7	81.3	-
MSR [58]	IJCAI'19	Syn	73.0	85.2	78.6	4.3
TextField [14]	TIP'19	Syn	79.9	81.2	80.6	6
SegLink++ [28]	PR'19	Syn	80.9	82.1	81.5	
ATTR [31]	CVPR'19	-	76.2	80.9	78.5	-
CSE [59]	CVPR'19	MLT	79.1	81.4	80.2	0.42
PSENet-1s [13]	CVPR'19	MLT	77.96	84.02	80.87	3.9
LOMO [18]	CVPR'19	MLT+	75.7	88.66	81.6	4.4
LOMO* [18]	CVPR'19	MLT+	79.3	87.6	83.3	
CRAFT [29]	CVPR'19	Syn	79.9	87.6	83.6	-
TextDragon† [30]	ICCV'19	MLT+	75.7	85.6	80.3	-
PAN-640 [10]	ICCV'19	Syn	81.0	**89.3**	85.0	**39.6**
DB [12]	AAAI'20	Syn	82.5	87.1	84.7	32
ContourNet [60]	CVPR'20	-	83.9	86.9	85.4	3.8
DRRG [8]	CVPR'20	MLT	84.93	86.54	85.73	-
ABCNet† [33]	CVPR'20	MLT+	81.3	87.9	84.5	-
TextRay[35]	MM'20	-	77.9	83.5	80.06	-
KPN-640	-	MLT	82.33	88.00	85.07	22.73
KPN-832	-	MLT	85.6	88.66	87.11	15.03
KPN MS	-	MLT	**87.04**	88.17	**87.60**	-

TABLE V

EXPERIMENTAL RESULTS ON CTW-1500

Methods	Paper	Ext	R	P	H	FPS
TextSnake [44]	ECCV'18	Syn	85.3	67.9	75.6	-
CTD [61]	PR'19	Syn	65.2	74.3	69.5	-
SegLink++ [28]	PR'19	Syn	79.8	82.8	81.3	-
TextField* [14]	TIP'19	Syn	79.8	83.0	81.4	6
MSR[58]	IJCAI'19	Syn	79.0	84.1	81.5	4.3
CSE [59]	CVPR'19	MLT	76.0	81.1	78.4	0.38
LOMO* [18]	CVPR'19	MLT+	89.2	69.6	78.4	4.4
LSAE [11]	CVPR'19	Syn	77.8	82.7	80.1	3
ATRR [31]	CVPR'19	-	80.2	80.1	80.1	-
PSENet-1s [13]	CVPR'19	MLT	79.7	84.8	82.2	3.9
CRAFT [29]	CVPR'19	Syn	81.1	86.0	83.5	-
TextDragon†[30]	ICCV'19	MLT+	82.8	84.5	83.6	-
PAN-640 [10]	ICCV'19	Syn	81.2	86.4	83.7	**39.8**
DB [12]	AAAI'20	Syn	80.2	**86.9**	83.4	22
ContourNet [60]	CVPR'20	-	84.1	83.7	83.9	4.5
ABCNet† [33]	CVPR'20	MLT+	83.4	84.4	81.4	-
DRRG [8]	CVPR'20	MLT	83.02	85.93	84.45	-
KPN-640	-	MLT	82.86	84.03	83.44	24.25
KPN-832	-	MLT	84.19	84.36	84.27	16.30
KPN MS	-	MLT	**86.44**	84.04	**85.22**	-

D. Comparisons With the State-of-the-Arts

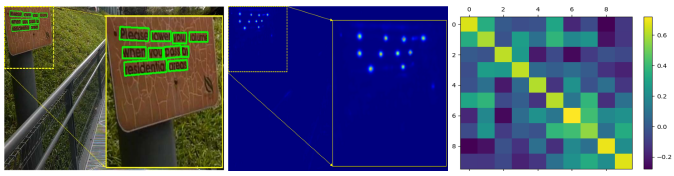
Comparisons of the proposed KPN with the state-of-the-art (SOTA) methods on Total-Text [53], CTW-1500 [54], and ICDAR 2015 [55] are listed in Tables IV–VI. Total-Text and CTW-1500 consists of images with curve texts, and ICDAR 2015 consists of images with quadrilateral texts. Notably, in Tables IV–VI, "Ext" denotes extra training data for pretraining, "Syn" represents SynText dataset for pretraining, "MLT" represents ICDAR2017-MLT dataset for pretraining, and "MLT+" represents ICDAR 2017-MLT dataset and additional datasets for pretraining.

1) Curve Text: **Total-Text** mainly contains images with word-level annotated curve texts. We set the threshold $T_c = 0.2$ and $T_i = 0.6$, and we test two pairs of size $[512, 640]$ and $[512, 832]$, i.e., KPN-640 and KPN-832. The representative visual results are shown in Fig. 10(a). As shown in Fig. 10(a), we can see that the predicted results of text instances are well separated. We utilize the official evaluating script in Total-Text [53], the results are listed in Table IV. Our KPN outperforms DRRG [8] by 1.38% in terms of H-mean, meanwhile outperforms ContourNet by 11.23 FPS in speed. Pixel aggregation network (PAN) proposes a new network that is faster than ResNet-50, while DB-640 only contains an individual FCN branch. However, our KPN, respectively, outperforms PAN-640 and DB by 2.11% and 2.41% in terms of H-means. In addition, KPN achieves comparable efficiency with PAN and DB-640. Overall, our KPN achieves SOTA performance on Total-Text.

CTW-1500 mainly contains images with curve texts with line-level annotations. We set the threshold $T_c = 0.2$, $T_i = 0.625$, and test in the size $[512, 640]$ and $[512, 832]$, i.e., KPN-640 and KPN-832. Since CTW-1500 is annotated

(a) (b) (c)

Fig. 10. Detected results on Total-Text, CTW-1500, and ICDAR 2015. The second row shows the predicted center point maps (in center red boxes), and the individual text instances in different channels. (a) Total-Text. (b) CTW-1500. (c) ICDAR 2015.



(a) (b) (c)

Fig. 11. Visualizing the similarity matrix S for Fig. 10(c). text instances are tiny and dense in this image. (a) Detected contour. (b) Gaussian center map. (c) S.

with line level, it may be difficult to predict the center point of the text line. Thus, our KPN may fail in detecting very long text. The detailed experimental results are listed in Table V. According to Table V, our KPN outperforms ContourNet [60] by 1.32% in terms of H-means, meanwhile surpassing it on efficiency with a significant margin (KPN 16.30 FPS versus ContourNet 4.5 FPS). And our KPN is comparable with DRRG [8] (84.27% versus 84.45%). Apparently, our KPN achieves promising performance on CTW-1500.

According to the experimental results on Total-Text and CTW-1500, KPN achieves promising efficiency and superior performance. The efficiency lies in our postprocessing-free segmentation-based framework that cleverly avoids computational-cost postprocessing. In addition, as shown in Fig. 10(a) and (b), KPN effectively separates neighboring text instances by classifying different texts into instance-independent feature maps. The effectiveness lies in the orthogonality between kernel proposals. When the kernel proposals are orthogonal between each other, they mainly have important self-information learned by the network and position information by position embedding instead of the shared information of other texts.

TABLE VI
EXPERIMENTAL RESULTS ON ICDAR 2015

Methods	Paper	Ext	R	P	H	FPS
DDR* [22]	ICCV'17	-	80.0	88.0	83.8	1.1
MCN [62]	CVPR'18	Syn	80	72	76	-
RRPN* [7]	TMM'18	-	77	84	80	3.3
TextSnake [44]	ECCV'18	Syn	84.90	80.40	82.6	1.1
Textboxes++*†[21]	TIP'18	Syn	78.50	87.80	82.90	2.3
PixelLink [9]	AAAI'18	Syn	82.0	85.5	83.70	3.
FTSN [17]	ICPR'18	Syn	80.0	88.6	84.1	2.5
IncepText [38]	IJCAI'18	-	80.6	90.5	85.3	
FOTS† [63]	CVPR'18	MLT+	82.04	88.84	85.31	7.8
SegLink++ [28]	PR'19	Syn	80.3	83.7	82.0	7.1
TextField* [14]	TIP'19	Syn	83.9	84.3	84.1	1.8
PAN [10]	ICCV'19	Syn	81.9	84.0	82.9	**26.1**
TextDragon†[30]	ICCV'19	MLT+	84.82	81.82	83.05	-
PSENet-1s [13]	CVPR'19	MLT	86.92	84.50	85.69	1.6
LSAE [11]	CVPR'19	Syn	85.0	88.3	86.6	3.0
CRAFT [29]	CVPR'19	Syn	84.3	89.8	86.9	-
LOMO [18]	CVPR'19	MLT+	83.5	**91.3**	87.2	3.4
ATRR [31]	CVPR'19	-	86.0	89.2	**87.6**	-
DRRG [8]	CVPR'20	MLT	84.69	88.53	86.56	-
ContourNet [60]	CVPR'20	-	86.1	87.6	86.9	3.5
DB [12]	AAAI'20	Syn	83.2	91.8	87.3	12
KPN-1280	-	MLT	83.15	84.08	83.61	12.2
KPN-1920	-	MLT	84.83	88.28	86.52	6.28
KPN MS	-	MLT	**86.96**	87.84	87.40	-

(a) (b) (c)

Fig. 12. Some visual comparison results with PSENet [13] and DB [12], where the results of PSENet and DB are reproduced by their official open-source code and model, where DB use the ResNet-50 with deformable convolution as Bockone and PSENet use the ResNet-50 as Bockone. (a) PSENet. (b) KPN. (c) DB.

2) Quadrilateral Text: **ICDAR 2015** mainly contains quadrilateral texts annotated with word level. Let $T_c = 0.24$ and $T_i = 0.8$, and we resize the longest side to 1280 and 1920 (i.e., KPN-1280 and KPN-1920). ICDAR 2015 contains a lot of small texts, but it may be hard to precisely detect the masks of small texts, especially utilizing 1/2 scale of feature maps of the input image. The detailed experimental results are listed in Table VI. According to Table VI, our KPN outperforms PAN [10] by 3.62% in terms of H-means, outperforms TextDragon [30] by 3.47% in terms of H-means. In addition, representative visual results are shown in Fig. 10(c). According to the experimental results on quadrilateral texts of ICDAR 2015, we can find that the predictions of independent text masks are not very satisfactory on tiny texts, as shown in Fig. 10(c). Hence, we further analyze the similarity matrix S of Fig. 10(c) in Fig. 11. From Fig. 11(c), we can find that the similarity matrix S is not orthogonal enough, inducing some neighboring text instances with small response values of center points. Fortunately, these small response values can be easily filtered by a threshold ($T_i = 0.8$). Hence, our method can still accurately separate adjacent texts in this case.

3) Visual Comparison: As shown in Fig. 12, we give a more intuitive comparison between the presented method and the SOTA methods (including PSENet [13] and DB [12]) by intermediate visual comparison. In Fig. 12, we visualize the final detections, the Gaussian center in KPN, the min text kernel in PSENet, and the probability map in DB. PSENet uses the min text kernel to separate neighboring text and then adopts the scale expansion algorithm to reconstruct text instances, which is not efficient and also fails frequently in many cases. Therefore, the detection speed and performance of PSENet are not so satisfactory. DB uses the probability map which is obtained by shrinking the annotations with the Vatti

(a) (b)

Fig. 13. Some visual examples of failure cases. Objects included in green contours are results of KPN; objects included in red contours are ground truths. (a) Failure cases caused by large character spacing and miss annotations. (b) Failure cases cause by inaccurate annotations and extreme size of texts.

clipping algorithm [15] to separate neighboring text. Then, it also uses the inverse transformation of the Vatti clipping algorithm [15] to get the final detection boundaries. Therefore, the detection speed of DB is quite fast. However, due to the manual scaling rules, the final detection boundaries of DB cannot cover the text well in many cases, either too large or too small, as shown in Fig. 12(c). Although it does not affect the performance of evaluation, inaccurate detection boundaries will lead to various problems in application, such as the recognition model cannot correctly recognize characters in the optical character recognition (OCR) system. In comparison, our method can not only separate neighboring texts well, but also ensure accurate boundary detection and fast detection speed, as shown in Fig. 12(b) and listed in Table IV and V.

E. Weakness

As demonstrated in the previous experiments, KPN achieves superior performance in detecting texts of arbitrary shapes. But there is still a chance of failure cases due to the complexity of scene images, such as object occlusion, large character spacing. Some failure examples are shown in Fig. 13. As shown in Fig. 13, KPN has some false detections on some text-like areas and miss detections for some extremely large or small complex text. These cases are still very challenging and nontrivial in the paradigm of text detection. However, KPN can successfully detect some text instances with missing annotations, as shown in the bottom image of Fig. 13(a).

V. Conclusion

In this article, we propose an innovative KPN for arbitrary shape text detection. The proposed KPN is the first to introduce the dynamic convolution kernel strategy to efficiently and effectively separate neighboring text instances by classifying different texts into instance-independent feature maps. Our KPN is efficient and does not rely on complex postprocessing. In addition, we also propose a novel OLL that directly enforces the independence between kernel proposals via orthogonal constraints. Specifically, our kernel proposals contain important self-information and position information, resulting in the effectiveness of separating neighboring texts and improving the robustness against tiny intervals or unclear boundaries. Extensive experiments on challenging datasets verify the impressive performance and efficiency of our method.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[2] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, Jul. 2017, pp. 936–944.

[3] Q. He, X. Sun, Z. Yan, and K. Fu, "DABNet: Deformable contextual and boundary-weighted network for cloud detection in remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–16, 2022.

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, Jun. 2015, pp. 3431–3440.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, vol. 9351, N. Navab, J. Hornegger, W. Wells, and A. F. Frangi, Eds., 2015, pp. 234–241.

[6] X. Zhou *et al.*, "EAST: An efficient and accurate scene text detector," in *Proc. CVPR*, Jul. 2017, pp. 2642–2651.

[7] J. Ma *et al.*, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Trans. Multimedia*, vol. 20, no. 11, pp. 3111–3122, Nov. 2018.

[8] S.-X. Zhang *et al.*, "Deep relational reasoning graph network for arbitrary shape text detection," in *Proc. CVPR*, Jun. 2020, pp. 9696–9705.

[9] D. Deng, H. Liu, X. Li, and D. Cai, "PixelLink: Detecting scene text via instance segmentation," in *Proc. AAAI*, 2018, pp. 6773–6780.

[10] W. Wang *et al.*, "Efficient and accurate arbitrary-shaped text detection with pixel aggregation network," in *Proc. CVPR*, Oct. 2019, pp. 8439–8448.

[11] Z. Tian *et al.*, "Learning shape-aware embedding for scene text detection," in *Proc. CVPR*, Jun. 2019, pp. 4234–4243.

[12] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization," in *Proc. AAAI*, 2020, pp. 11474–11481.

[13] W. Wang *et al.*, "Shape robust text detection with progressive scale expansion network," in *Proc. CVPR*, Jun. 2019, pp. 9336–9345.

[14] Y. Xu, Y. Wang, W. Zhou, Y. Wang, Z. Yang, and X. Bai, "TextField: Learning a deep direction field for irregular scene text detection," *IEEE Trans. Image Process.*, vol. 28, no. 11, pp. 5566–5579, Nov. 2019.

[15] B. R. Vatti, "A generic solution to polygon clipping," *Commun. ACM*, vol. 35, no. 7, pp. 56–63, 1992.

[16] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *Proc. ICCV*, 2017, pp. 2980–2988.

[17] Y. Dai *et al.*, "Fused text segmentation networks for multi-oriented scene text detection," in *Proc. ICPR*, Aug. 2018, pp. 3604–3609.

[18] C. Zhang *et al.*, "Look more than once: An accurate detector for text of arbitrary shapes," in *Proc. CVPR*, Jun. 2019, pp. 10552–10561.

[19] M. Liao, G. Pang, J. Huang, T. Hassner, and X. Bai, "Mask TextSpotter v3: Segmentation proposal network for robust scene text spotting," *CoRR*, vol. abs/2007.09482, pp. 1–20, Jul. 2020.

[20] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "TextBoxes: A fast text detector with a single deep neural network," in *Proc. AAAI*, 2017, pp. 4161–4167.

[21] M. Liao, B. Shi, and X. Bai, "TextBoxes++: A single-shot oriented scene text detector," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3676–3690, Aug. 2018.

[22] W. He, X.-Y. Zhang, F. Yin, and C.-L. Liu, "Deep direct regression for multi-oriented scene text detection," in *Proc. ICCV*, Oct. 2017, pp. 745–753.

[23] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang, "UnitBox: An advanced object detection network," in *Proc. ACM MM*, Oct. 2016, pp. 516–520.

[24] J.-B. Hou *et al.*, "HAM: Hidden anchor mechanism for scene text detection," *IEEE Trans. Image Process.*, vol. 29, pp. 7904–7916, 2020.

[25] X. Sun, P. Wang, C. Wang, Y. Liu, and K. Fu, "PBNet: Part-based convolutional neural network for complex composite object detection in remote sensing imagery," *ISPRS J. Photogramm. Remote Sens.*, vol. 173, pp. 50–65, Mar. 2021.

[26] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network," in *Proc. ECCV*, 2016, pp. 56–72.

[27] B. Shi, X. Bai, and S. Belongie, "Detecting oriented text in natural images by linking segments," in *Proc. CVPR*, Jul. 2017, pp. 3482–3490.

[28] J. Tang, Z. Yang, Y. Wang, Q. Zheng, Y. Xu, and X. Bai, "SegLink++: Detecting dense and arbitrary-shaped scene text by instance-aware component grouping," *Pattern Recognit.*, vol. 96, Dec. 2019, Art. no. 106954.

[29] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in *Proc. CVPR*, Jun. 2019, pp. 9365–9374.

[30] W. Feng, W. He, F. Yin, X.-Y. Zhang, and C.-L. Liu, "TextDragon: An end-to-end framework for arbitrary shaped text spotting," in *Proc. ICCV*, Oct. 2019, pp. 9075–9084.

[31] X. Wang, Y. Jiang, Z. Luo, C.-L. Liu, H. Choi, and S. Kim, "Arbitrary shape scene text detection with adaptive text region representation," in *Proc. ICCV*, Jun. 2019, pp. 6449–6458.

[32] Y. Wang, H. Xie, Z.-J. Zha, M. Xing, Z. Fu, and Y. Zhang, "ContourNet: Taking a further step toward accurate arbitrary-shaped scene text detection," in *Proc. CVPR*, Jun. 2020, pp. 11753–11762.

[33] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "ABCNet: Real-time scene text spotting with adaptive Bezier-curve network," in *Proc. CVPR*, Jun. 2020, pp. 9806–9815.

[34] Y. Zhu, J. Chen, L. Liang, Z. Kuang, L. Jin, and W. Zhang, "Fourier contour embedding for arbitrary-shaped text detection," in *Proc. CVPR*, Jun. 2021, pp. 3123–3131.

[35] F. Wang, Y. Chen, F. Wu, and X. Li, "TextRay: Contour-based geometric modeling for arbitrary-shaped scene text detection," in *Proc. 28th ACM Int. Conf. Multimedia*, C. W. Chen *et al.*, Eds., Oct. 2020, pp. 111–119.

[36] P. Dai, S. Zhang, H. Zhang, and X. Cao, "Progressive contour regression for arbitrary-shape scene text detection," in *Proc. CVPR*, Jun. 2021, pp. 7393–7402.
[37] S.-X. Zhang, X. Zhu, C. Yang, H. Wang, and X.-C. Yin, "Adaptive boundary proposal network for arbitrary shape text detection," in *Proc. ICCV*, Oct. 2021, pp. 1305–1314.
[38] Q. Yang, M. Cheng, W. Zhou, Y. Chen, M. Qiu, and W. Lin, "IncepText: A new inception-text module with deformable PSROI pooling for multi-oriented scene text detection," in *Proc. IJCAI*, Jul. 2018, pp. 1071–1077.
[39] K. Sofiiuk, K. Sofiyuk, O. Barinova, A. Konushin, and O. Barinova, "AdaptIS: Adaptive instance selection network," in *Proc. ICCV*, Oct. 2019, pp. 7354–7362.
[40] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. CVPR*, Jun. 2019, pp. 4401–4410.
[41] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9626–9635.
[42] Z. Tian, C. Shen, and H. Chen, "Conditional convolutions for instance segmentation," in *Proc. ECCV*, in Lecture Notes in Computer Science, vol. 12346, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., 2020, pp. 282–298.
[43] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "SOLOv2: Dynamic and fast instance segmentation," 2020, *arXiv:2003.10152*.
[44] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, "TextSnake: A flexible representation for detecting text of arbitrary shapes," in *Proc. ECCV*, 2018, pp. 19–35.
[45] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *Proc. ECCV*, vol. 11218, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., 2018, pp. 765–781.
[46] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *Proc. ICCV*, Oct. 2019, pp. 6568–6577.
[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.
[48] D. Novotný, S. Albanie, D. Larlus, and A. Vedaldi, "Semi-convolutional operators for instance segmentation," in *Proc. ECCV*, in Lecture Notes in Computer Science, vol. 11205, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., 2018, pp. 89–105.
[49] R. Liu *et al.*, "An intriguing failing of convolutional neural networks and the CoordConv solution," in *Proc. NeurIPS*, 2018, pp. 9628–9639.
[50] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proc. 3DV*, Oct. 2016, pp. 565–571.
[51] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. ICCV*, Oct. 2017, pp. 2999–3007.
[52] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. CVPR*, Jun. 2016, pp. 761–769.
[53] C. K. Ch'ng and C. S. Chan, "Total-Text: A comprehensive dataset for scene text detection and recognition," in *Proc. ICDAR*, Nov. 2017, pp. 935–942.
[54] Y. Liu, L. Jin, S. Zhang, and S. Zhang, "Detecting curve text in the wild: New dataset and new solution," *CoRR*, vol. abs/1712.02170, pp. 1–9, Dec. 2017.
[55] D. Karatzas *et al.*, "ICDAR 2015 competition on robust reading," in *Proc. ICDAR*, Aug. 2015, pp. 1156–1160.
[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.
[57] N. Nayef *et al.*, "ICDAR2017 robust reading challenge on multi-lingual scene text detection and script Identification—RRC-MLT," in *Proc. ICDAR*, Nov. 2017, pp. 1454–1459.
[58] C. Xue, S. Lu, and W. Zhang, "MSR: Multi-scale shape regression for scene text detection," in *Proc. IJCAI*, Aug. 2019, pp. 989–995.
[59] Z. Liu, G. Lin, S. Yang, F. Liu, W. Lin, and W. L. Goh, "Towards robust curve text detection with conditional spatial expansion," in *Proc. CVPR*, Jun. 2019, pp. 7269–7278.
[60] Y. Wang, H. Xie, Z.-J. Zha, M. Xing, Z. Fu, and Y. Zhang, "ContourNet: Taking a further step toward accurate arbitrary-shaped scene text detection," in *Proc. CVPR*, Jun. 2020, pp. 11750–11759.
[61] Y. Liu, L. Jin, S. Zhang, C. Luo, and S. Zhang, "Curved scene text detection via transverse and longitudinal sequence connection," *Pattern Recognit.*, vol. 90, pp. 337–345, Jun. 2019.
[62] Z. Liu, G. Lin, S. Yang, J. Feng, W. Lin, and W. L. Goh, "Learning Markov clustering networks for scene text detection," in *Proc. CVPR*, Jun. 2018, pp. 6936–6944.
[63] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, "FOTS: Fast oriented text spotting with a unified network," in *Proc. CVPR*, Jun. 2018, pp. 5676–5685.

Shi-Xue Zhang received the M.S. degree in computer science from the University of Science and Technology Beijing, Beijing, China, in 2021, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology.

His research interests include text detection, pattern recognition, and deep learning.

Xiaobin Zhu received the M.E. degree from Beijing Normal University, Beijing, China, in 2006, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2013.

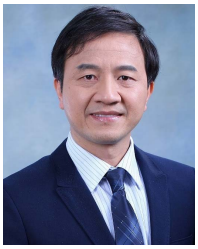
He is currently an Associate Professor with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing. His research interests include machine learning, image content analysis and classification, and multimedia information indexing and retrieval.

Jie-Bo Hou received the B.Sc. and Ph.D. degrees in computer science from the University of Science and Technology Beijing, Beijing, China, in 2014 and 2021, respectively.

His research interests include text detection, pattern recognition, and deep learning.

Chun Yang received the B.Sc. and Ph.D. degrees in computer science from the University of Science and Technology Beijing, China, in 2011 and 2018, respectively.

He is currently a Faculty Member with the School of Computer and Communication Engineering, University of Science and Technology Beijing. His current research interests include pattern recognition, classifier ensemble, and document analysis and recognition.



Xu-Cheng Yin (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from the University of Science and Technology Beijing, Beijing, China, in 1999 and 2002, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2006.

He was a Visiting Professor with the College of Information and Computer Sciences, University of Massachusetts Amherst, USA, three times in 2014, 2014, and 2016, respectively. He is currently a Full Professor and the Director of Pattern Recognition and Information Retrieval Laboratory, Department of Computer Science and Technology, University of Science and Technology Beijing.