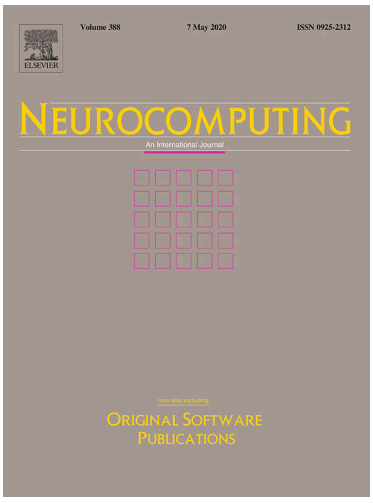
Journal Pre-proofs



RFRN: A Recurrent Feature Refinement Network for Accurate and Efficient Scene Text Detection

Guanyu Deng, Yue Ming, Jing-Hao Xue

PII: S0925-2312(20)31712-4
DOI: https://doi.org/10.1016/j.neucom.2020.10.099
Reference: NEUCOM 23010

To appear in: *Neurocomputing*

Received Date: 24 June 2020
Accepted Date: 28 October 2020

Please cite this article as: G. Deng, Y. Ming, J-H. Xue, RFRN: A Recurrent Feature Refinement Network for Accurate and Efficient Scene Text Detection, *Neurocomputing* (2020), doi: https://doi.org/10.1016/j.neucom.2020.10.099

RFRN: A Recurrent Feature Refinement Network for Accurate and Efficient Scene Text Detection

Guanyu Deng[a], Yue Ming[a,*] and Jing-Hao Xue[b]

[a]Beijing Key Laboratory of Work Safety and Intelligent Monitoring, School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

[b]Department of Statistical Science, University College London, London WC1E 6BT, U.K.

ARTICLE INFO

Keywords:
Scene text detection
Recurrent segmentation
Feature pyramid network
Feature refinement

ABSTRACT

Scene text detection plays a vital role for scene text understanding, but arbitrary-shaped text detection remains a significant challenge. To extract discriminative features, most recent state-of-the-art methods adopt heavy networks, resulting in parameter redundancy and inference inefficiency. For accurate and efficient scene text detection, in this paper we propose a novel recurrent feature refinement network (RFRN). RFRN, as a recurrent segmentation framework, contains a recurrent path augmentation that refines the previous feature maps as inner states, which not only helps improve the segmentation quality, but also fully facilitates the reuse of parameters and low computational cost. During testing, RFRN discards redundant prediction procedures for efficient inference, and achieves a good balance between speed and accuracy of inference. We conduct experiments on four challenging scene text benchmarks, CTW1500, Total-Text, ICDAR2015 and ICDAR2017-MLT, which include curved texts and multi-oriented texts with complex background. The results show that the proposed RFRN achieves competitive performance on detection accuracy while maintaining computational efficiency.

1. Introduction

Owing to the rapid development of convolutional neural network (CNN), the performance of scene text detection has obtained a significant improvement [1–7]. Different architectures have been proposed to treat scene texts as one specific target and use feature pyramid network (FPN) [8] to extract multi-scale features. However, two remaining challenges constrain the wide application of these deep networks. The first challenge is the visual diversity of scene texts, in shape, scale and appearance. In order to improve the performance of arbitrary-shaped text detection, many efforts have been made and achieved impressive results, such as TextSnake [9], MSR [10] and PSENet [11]. But the results of these methods are still unsatisfactory in detection accuracy, due to the insufficient learning of discriminative features. The second challenge is the demand for a good balance between the inference efficiency and accuracy. Recent state-of-the-art methods show good detection accuracy by adopting heavy networks, at the expense of suffering from redundant parameters and high computational cost.

In order to solve the above-mentioned challenges, we propose a recurrent feature refinement network (RFRN), as illustrated in Figure 1, focusing on learning discriminative features and simultaneously reducing parameter redundancy, thus striking a good balance between the efficiency and accuracy of inference.

FPN is widely applied in recent text detection methods, however, the multi-scale information may not be adequately fused by these methods due to the limited receptive field. Therefore, different from them, the proposed RFRN leverages FPN under a recurrent segmentation architecture, which enables iterative refinement of internal feature representation as inner states. That is, the key component of our RFRN is a recurrent feature refinement module (Figure 1(b)), which embeds in CNN some hidden and refinable states representing internal features, that store the convolutional values and recurrently refine them with the multi-scale semantic information from FPN. In the inference phase, RFRN recurrently refines its internal features by leveraging both the previous multi-scale feature maps (green blocks in Figure 1(b)) and the previous inner hidden states. In this way, the semantic information at different scale is refined through a recurrent route, more than the top-down feature augmentation in FPN, and thus the network can obtain enriched semantic features to handle arbitrary-shaped texts better. Moreover, instead of stacking parameters to achieve a deep network, RFRN reuses its parameters for recurrent segmentation. Each parameter hence can be learned by the back-propagated errors of different predictions. This helps enhance the descriptive ability of each parameter and provides low parameter redundancy. In addition, we simplify the inference procedure in RFRN to reduce its computational cost of recurrent prediction.

In summary, the main contributions of our work are threefold: 1) To improve the detection accuracy, we propose RFRN to promote the feature learning in FPN-based text detection networks, by recurrently updating feature maps as inner states and consequently refining previous predictions. 2) To reduce parameter redundancy, we embed in RFRN a parameter-reuse strategy, which can act as a regularizer on the shared parameters. This helps RFRN converge to an accurate network, better than the one with stacked parameters. 3) To illustrate the generality and effectiveness of our approach, we conduct extensive experiments on four challenging scene text benchmarks, demonstrating that RFRN is an

*Corresponding author: Yue Ming
yming@bupt.edu.cn (Y. Ming)
ORCID(s):

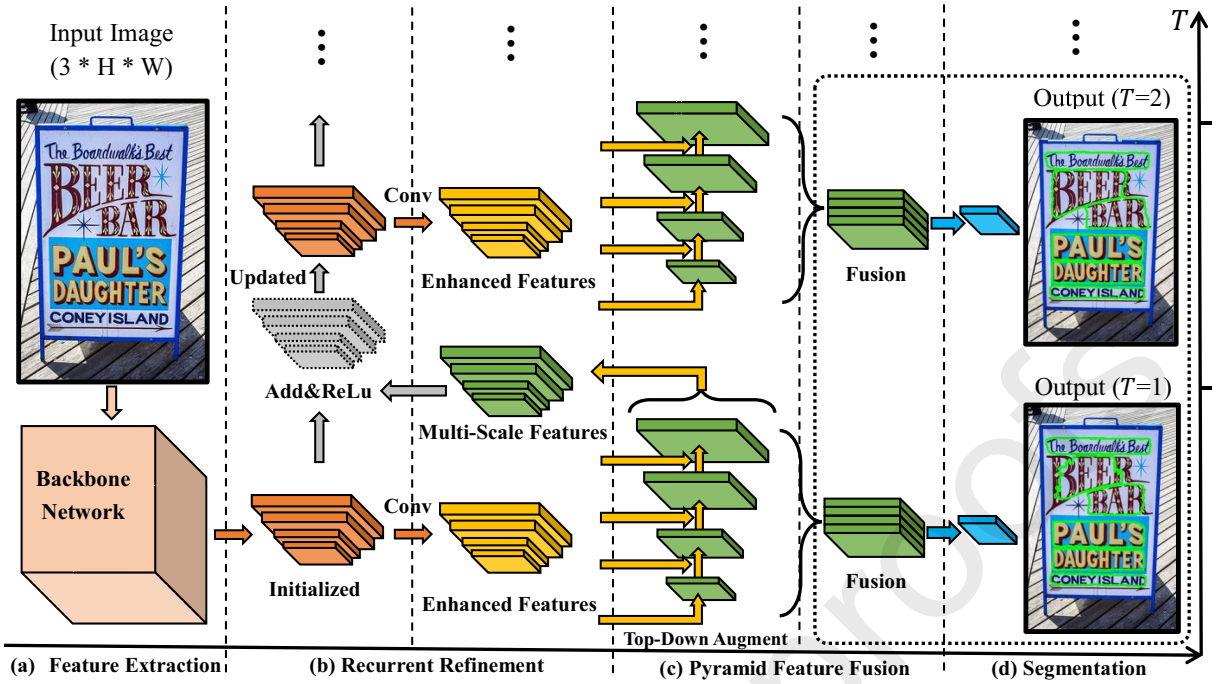


Figure 1: Pipeline of the proposed RFRN. (a) Feature Extraction. (b) Recurrent Refinement. (c) Pyramid Feature Fusion. (d) Segmentation. Given an image, the RFRN recurrently refines its inner states of multi-scale features.

accurate and fast arbitrary-shaped text detector.

The rest of this paper is organized as follows. In Section 2, we shortly review some related work on scene text detection. The proposed method is detailed in Section 3. The experimental results are given in Section 4, and we provide the conclusion in Section 5.

2. Related Work

In recent years, deep learning based scene text detection methods have been extensively exploited. Following the object detection frameworks [12, 13] or semantic segmentation frameworks [8], many regression-based or segmentation-based scene text detection methods have been proposed. Based on the target to predict, recent research have two trends: *anchor-based* and *anchor-free methods*. Some of these methods also focus on the inference speed, as *efficient scene text detectors*.

Anchor-based methods are usually adapted from various generic object detection techniques, such as SSD (Single Shot multibox Detector) [12] and Faster R-CNN [13]. Due to the aspect ratio of long text, TextBoxes [14] deployed SSD with well designed long-scale anchor boxes and convolution receptive field. For multi-oriented text detection, TextBoxes++ [15] defined two representation approaches for multi-oriented text: quadrilateral box and rotated rectangles. Then they trained SSD to regress such box. Base on Faster-RCNN, RRPN [16] regressed the angle of text boxes and proposed Rotated RoI Pooling to align multi-oriented proposals. RRD [17] designed rotation-sensitive and rotation-invariant features extraction to better deal with text detection. SegLink++ [18] regressed text local-multi-oriented boxes, and learned the attractive-and-repulsive relationship between text local components to generate arbitrary-shaped region. Wang et al. [19] proposed recurrent neural network based adaptive text region representation to refine text proposals into arbitrary shape by an adaptive number of boundary points. Liu et al. [20] proposed a two-stage tightness prior instance segmentation framework and used the border information to adjust arbitrary-shaped text. Most of these methods require fine prior anchor setting to handle the texts in various shapes. Moreover, due to the limitation of bounding boxes regression that predicts the rectangle text region, some of anchor-based methods are not able to detect curved texts. Our approach adapts to arbitrary-shaped texts with a segmentation architecture which is simpler for curved text detection.

Anchor-free methods [9, 11, 21–25] focus on how to formulate the text region and perform well on curved texts. They usually deploy fully convolutional network (FCN) to produce pixel level prediction and then apply post-processing to obtain text boxes. Some of these methods tend to achieve accurate results by pixel-level regression. EAST [21] first segmented shrunk text regions, then predicted per-pixel offsets and generated pixel level regression. For long text detection, Lyu et al. [26] predicted four corner points of text bounding boxes and segmented relative text regions in such positions. For arbitrary-shaped text detection, TextSnake [9] described text instance as an ordered sequence of overlapping disks at the text center line. SAE [24] mapped each pixel into embedding vector while the pixels

of same instance have closer embedding distance. Wang et al. [27] also predicted embedding vectors for each pixel and proposed a pixel aggregation algorithm to locate each instance. R-Net [7] introduced a bi-directional convolutional framework to handle large scale variety for regression-based method. Similarly, segmentation-based methods produce per-pixel classification to distinguish text areas from background. TextMountain [6] used text border-center information to describe arbitrary-shaped text instances and designed a parallel post-processing. PSENet [11] directly segmented the shrunk text instance in different scales, and then generated text region with progressive scale expansion algorithm. CRAFT [25] reconstructed text instance by segmenting characters region and corresponding affinity saliency map. DB [2] proposed a differentiable binarization processing to better guide a segmentation network training and applied the deformable convolutional network [28], which greatly improved the accuracy of detecting text instances of extreme aspect ratios. Although pixel-level prediction methods accurately detect curved texts, many of these methods suffer from time-consuming inference and post-processing steps. Heavy convolutional neural networks, such as ResNet-50 [29] and VGG-16 [30], were adopted to extract expressive features [6, 7, 11, 19, 24], at the expense of more redundant parameters and heavier computation. Moreover, a large network is more vulnerable to overfitting due to insufficient training data. Most text detectors utilize FPN-based network to produce predictions. The low resolution feature maps, which have higher level semantic information, are up-sampled and fused with high resolution feature maps. Their predictions are delivered in a top-down path, leading to a unidirectional route. The Path Aggregation Network (PANet) [31] and the Feature Pyramid Enhancement Module (FPEM) [27] enriched the multi-scale information, by applying a double-path enhancement that included a top-down path and a down-top path. In contrast, our network has a recurrent path to refine features. It recursively predicts segmentation results, resulting in a coarse-to-fine approach.

Efficient scene text detectors aim to achieve both fast inference and accurate detection. EAST [21] applied light-weight PVANet [32] to reduce inference time. They also provided a multi-oriented text detection head to better train the light-weight network. Other methods [15, 17, 33, 34] followed the SSD [12] architecture to obtain fast inference. They designed oriented adaptive anchors to detect text lines. Wang et al. [27] used a light-weight backbone and a pixel embedding branch to improve the detection accuracy on small texts, thus the network can work for smaller input resolution with faster inference. DB [2] obtained fast inference with light-weight ResNet18 and simple post-processing which directly expanded the shrunk text region. Nevertheless, some fast detectors are designed to detect quadrangular texts, so they are not able to deal with arbitrary-shaped text. The limited parameters also harm the representation ability of extracted features, leading to unsatisfactory feature learning, while our method aims to handle such an issue.

3. Proposed Method

3.1. Overview

In this paper, we focus on developing a network architecture to achieve efficient and accurate text detection. We propose the recurrent feature refinement network (RFRN) to learn robust representation, through recursively refining its internal features and reusing parameters within the network. The overall design of RFRN is illustrated in Figure 1, which includes four steps: feature extraction, recurrent refinement, pyramid feature fusion and segmentation. The RFRN enhances the feature representation with two mechanisms: recurrent refinement, and pyramid feature fusion.

We implement recurrent refinement as iterative updating of convolutional features. RFRN firstly extracts basic feature maps from the input images using backbone network (Figure 1(a)). These feature maps are then taken as initial values of the internal features, from which the recurrent refinement produces enhanced features (Figure 1(b)). The hyper-parameter T controls the times of refinement. After that, the feature fusion is applied to aggregate multi-scale information from different scale feature maps (Figure 1(c)). Then the segment-head predicts probability maps from the fused feature maps. Finally, we use the progressive scale expansion algorithm [11] as the post-processing algorithm to generate the whole text instance (Figure 1(d)).

3.2. Recurrent feature refinement module

We propose the recurrent feature refinement module (RFRM) to refine multi-scale semantic features. RFRM is a recurrent unit that stores and updates convolutional features iteratively. It is applied on each hierarchy level of feature pyramid to enhance semantic information multiple times. The times of enhancement is determined by hyper-parameter T, that is, RFRM refines feature maps $T-1$ times. The proposed RFRM contains two operation stages: updating and enhancing. The updating stage updates the internal features by fusing the previous internal values with the multi-scale features from FPN. Thus it refines the semantic information in the internal feature maps. Then the enhancing stage organizes output feature maps from the current internal features by a 3×3 convolutional layer. Figure 2 illustrates the details of RFRM.

When $t=1$, we initialize the internal features. In both the training and testing, the first internal features $\mathbf{h}_i^t$ is initialized using the value of backbone feature maps $\mathbf{x}_i^t$:

$$\mathbf{h}_i^t = \mathbf{x}_i^t, \;\; t=1, \; i=2,3,4,5, \tag{1}$$

where i denotes the hierarchy level of the feature maps in the network, indicating a certain resolution of the feature maps containing multi-level context information.

At the next prediction step t ($t=2,\dots,T$), current internal features $\mathbf{h}_i^t$ are updated by leveraging both the previous inner state $\mathbf{h}_i^{t-1}$ and the previous multi-scale features $\mathbf{y}_i^{t-1}$:

$$\mathbf{h}_i^t = \widetilde{\mathscr{F}}(\mathbf{h}_i^{t-1} + \mathbf{y}_i^{t-1}), \;\; t=2,\dots,T, \; i=2,3,4,5, \tag{2}$$

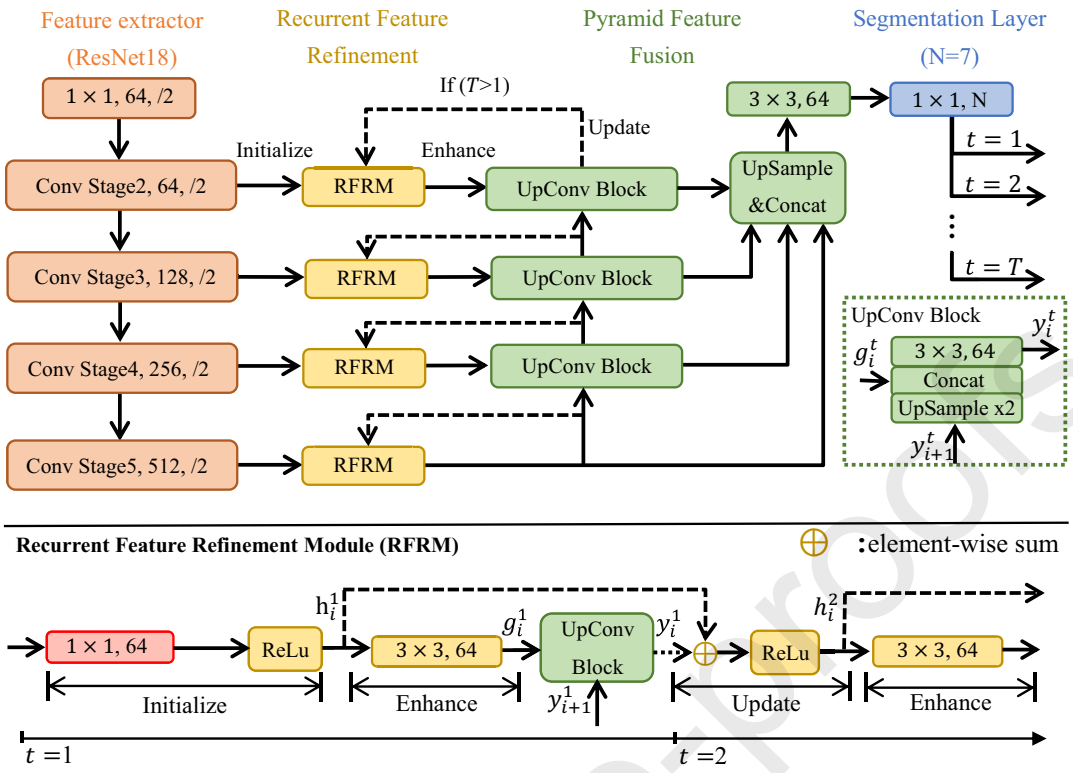


Figure 2: Upper panel: Network architecture of RFRN. "T" denotes the times of predictions. "N" denotes the numbers of output branches of different scales at each inference time (in both training and testing). We follow [11] to produce shrunk segment masks at 7 different scales ($N = 7$) and set the scale ratio to 0.4. Lower panel: Detailed structure of RFRM.

where $\widetilde{\mathcal{F}}$ is the activation function, $\mathbf{y}_i^{t-1}$ is the previous feature maps produced by pyramid fusion module at the ith hierarchy level. We use the element-wise sum operation to combine the context information within $\mathbf{h}_i^{t-1}$ and $\mathbf{y}_i^{t-1}$, and then calculate the activation value of the combined feature maps and update internal features $\mathbf{h}_i^t$. Since it is not usual to constrain the convolutional values to a certain range like $[0, 1]$, we use ReLu rather than sigmoid or tanh as activation for preserving consistency of convolutional values. This also prevents the network from gradient vanishing.

The final step of RFRM is the enhancing stage. In this stage, we apply a 3×3 convolution block to enlarge the receptive area within the inner feature maps:

$$\mathbf{g}_i^t = ReLu(Conv_i(\mathbf{h}_i^t)), \quad i = 2, 3, 4, 5, \tag{3}$$

where $Conv_i$ indicates the convolution operation at the ith level. The $Conv_i$ layer selects every 3×3 area around each element in the internal feature maps, to learn saliency features. In short, the updating stage combines the context information across different times and scales, and the enhancing stage gives spatial refinement.

Note that the parameters used for refinement are shared over different steps t, which means that the same RFRM is applied to produce $\mathbf{g}_i^1, \ldots, \mathbf{g}_i^T$. Thus the connection of different steps makes the feature refinement path a recurrent route.

In summary, the advantages of RFRM are mainly twofold. Firstly, RFRM further extends the information propagation path to a deeper network. With recurrent segmentation, the internal features, namely the inner state, are refined to have larger receptive field and more details. Secondly, it has parameter efficiency. Thanks to the parameter-reuse strategy, the network can well promote the description of text features, while requiring no extra parameters.

3.3. Pyramid feature fusion module

The pyramid feature fusion module aggregates the multi-scale information from the outputs of RFRMs. It produces the fused feature maps for predicting the final output masks. We inherit the feature pyramid structure that efficiently refines the context information with skip connection. It consists of two processing stages: top-down augmentation and feature fusion.

The feature maps produced by RFRM at four different scales, $\mathbf{g}_2^t, \mathbf{g}_3^t, \mathbf{g}_4^t, \mathbf{g}_5^t$, are taken as the input of top-down augmentation. These feature maps are fused from low to high resolution, to combine the low level texture information with high level semantic information. The top-down augmentation at the inference step t is expressed as

$$\mathbf{y}_i^t = Conv_i(upSample(\mathbf{y}_{i+1}^t) + \mathbf{g}_i^t), \quad i = 2, 3, 4, \tag{4}$$

where the $upSample$ operation fills the input feature maps to double its scale by bilinear interpolation; $Conv_i$ is a 3×3 convolution layer to smooth the output; and $\mathbf{y}_i^t$ is the augmentation result. After this, we obtain four scale features,

$\mathbf{y}_2^t, \mathbf{y}_3^t, \mathbf{y}_4^t, \mathbf{y}_5^t$, which are not only the inputs of feature fusion but also the inputs of RFRM. Thus the multi-scale features are feedback to RFRM to produce more expressive ones.

Then, for further enhancement, we apply the feature fusion stage to leverage all the feature maps. The four pyramid feature maps $\mathbf{y}_2^t, \mathbf{y}_3^t, \mathbf{y}_4^t, \mathbf{y}_5^t$ are up-sampled to have the same size and concatenated together:

$$\mathbf{f}^t = Concat(\mathbf{y}_2^t, upSp(\mathbf{y}_3^t), upSp(\mathbf{y}_4^t), upSp(\mathbf{y}_5^t)), \quad (5)$$

where the $upSp$ is $upSample$ in short; all four feature maps are up-sampled to have the same size with $\mathbf{y}_2^t$; and $\mathbf{f}^t$ is the fusion of these feature maps, which are concatenated on the channel dimension while keeping the spatial dimension. From the concatenated feature maps $\mathbf{f}^t$, in the segmentation module, which consists of one 1×1 convolution and sigmoid function, we generate per-pixel prediction.

As with RFRM, we shared the parameters of pyramid fusion modules at different inference steps $t = 1, \ldots, T$. Thus the network is more compact since requiring fewer parameters. For comparison, we also try a cascaded segmentation network with stacked parameters. Without parameter-sharing, it requires learning more parameters and delivers poorer test accuracy in our experiments. The sharing of parameters can be regarded as a regularizer constraining on all segment-heads, to force their weights to be equal, while the cascaded parameters do not have such regularization (detailed more in the ablation study in section 4). Moreover, in the testing, we discard the segmentation procedures except the last one for faster inference. Meanwhile, thanks to the feature refinement, the backbone features only need to be produced once in the entire forward pass. This also reduces the non-essential computational cost.

3.4. Optimization

As detailed in the previous sections, the proposed RFRN predicts T segmentation masks iteratively. We follow PSENet [11] to segment text region and text kernels. The text kernel is a shrunk area of the original text region. In the testing, only the last segmentation mask is considered as the final result. In the training, we keep all the segmentation masks to calculate the error for better convergence.

The loss function of each mask is formulated as

$$L^t = \alpha L_r^t + (1-\alpha) L_k^t, \quad (6)$$

where L_r^t and L_k^t denote the losses for the complete text region and the text kernels respectively; and α controls the balance between above two losses. Then, the final loss function can be formulated as the weighted sum of all predictions:

$$L_{all} = \frac{1}{T}\sum_{t=1}^{T} \beta_t L^t, \quad (7)$$

where β_t represents the weight of L^t and we discuss the influence of weights in section 4.

To compute the error, we adopt dice coefficient:

$$Dice(A, B) = \frac{2\sum_{x,y}(A_{x,y} \times B_{x,y})}{\sum_{x,y} A_{x,y}^2 + \sum_{x,y} B_{x,y}^2}, \quad (8)$$

where A and B are two segmentation masks, with $A_{x,y}$ represents the pixel value of position (x, y) ($A_{x,y} \in [0, 1]$). Thus, L_r^t and L_k^t can be calculate as

$$L_r^t = 1 - Dice(P^t, G^t), \quad (9)$$

$$L_k^t = 1 - \frac{\sum_i^{N-1} Dice(P_i^t, G_i^t)}{N-1}, \quad (10)$$

where N is the number of output branches which include one text region mask and $N-1$ text kernel masks, and P^t and G^t refer to the tth prediction and ground truth. The dice coefficient represents the similarity between two contours and it is scale-insensitive. Thus the dice coefficient can better guide the network training in a scale-invariant fashion.

4. Experimental Results and Analysis

In section 4.1, we first introduce four scene text benchmark datasets, CTW1500, Total-Text, ICDAR2015 and ICDAR2017 MLT, which contains curved and multi-oriented scene text instances, and then details the implementation of our approach. In section 4.2, we conduct the ablation study on the CTW1500 dataset to investigate the effectiveness of the proposed model. In section 4.3 and section 4.4, we compare our methods with other state-of-the-art methods on four benchmark datasets for curved and multi-oriented texts, respectively. In section 4.5, we compare computational costs of recent methods.

4.1. Datasets and implementation details

4.1.1. Datasets

SCUT-CTW1500 [35]: CTW1500 is a challenging dataset containing a large amount of arbitrary-shaped texts, where each image has more than one curved text instance. This dataset consists of 1000 scene images for training and 500 images for testing. All the text instances are annotated in the text-line level by using 14-vertices polygon. In addition, the texts are different in colors, sizes and fonts.

Total-Text [36]: Total-Text is also a challenging dataset of multi-oriented and curved texts. It consists of 1255 scene images for training and 300 images for testing. All the text instances are labeled by a word level polygon.

ICDAR2015 Incidental Scene Text [37]: ICDAR2015 is a commonly used dataset with multi-oriented text instances. Moreover, most texts are blurred and of various scales. The dataset contains 1000 training images and 500 testing images in natural scene. The text instances in ICDAR2015 have different orientations and are annotated by using 4-vertices quadrangle.

ICDAR2017 Multi-Lingual Text [38]: ICDAR2017-MLT includes multi-oriented scene text images of 9 languages representing 6 different scripts. Some languages are labeled in word-level and others are labeled in line-level. It contains 7200 training images, 1800 validation images and 9000 testing image. We use both training and validation images to train our models.

SynthText in the Wild [39] is a large scale dataset for pretraining the model [9, 26, 33]. All the images are generated with augments of random colors, fonts, scales, and orientations in natural scene. Thus it is useful for warming up the model. The dataset contains 800k images and provides both character-level and word-level annotation. We only use the word-level annotation to pretrain the model.

4.1.2. Implementation details

For maintaining both flexibility and efficiency, we adopt the ResNet-18 network as the backbone feature extractor. ResNet-18 generates feature maps of four scales with strides of 4, 8, 16, 32 to the input image, respectively. The feature maps have convolutional channels of 64, 128, 256, 512. For computational efficiency, we also use 1×1 convolution layer to squeeze the feature maps produced by ResNet-18 to have 64 channels. For training label design, we follow PSENet [11] to produce text region mask and text kernel masks in multi-scale, which does not rely on complicated handcraft annotation. We obtain 7 text masks $K_1, K_2, \ldots, K_7$, by shrinking the complete text region with offset distance $d_i, i \in (1, 7]$. The offset d_i is calculated according to [11]. For post-processing, we follow their Progressive Scale Expansion (PSE) algorithm to generate the text instance maps from the segmentation masks.

For better demonstrating the detection capacity, we provide two pre-train strategies: 1) pre-training on SynthText (SYN) [39] for 50k iterations, as with most recent methods [2, 9, 33]; 2) pre-training on ICDAR2017-MLT (IC17) [39] for 20k iterations, as with [11, 40]. Then, we finetune RFRN on evaluation datasets for 36k-74k iterations. For fair comparison, we also train our model from scratch on these datasets. The initial learning rate is set to 0.001 and we adopt poly learning rate with power 0.9. We use a weight decay of 0.0005 and momentum of 0.99. For overcoming the imbalance of positives and negatives, we adopt Online Hard Example Mining (OHEM), where the negatives rate is set to 3. For data augmentation, we apply random scale, random horizontal flip, random rotation and random crop on training images. The training images are randomly scaled as d_x times of its original size, where $d_x = 0.5, 1, 2, 3$. Then we randomly rotate input images in the range of -10 to 10 degree. After that, the images are cropped to 640×640 at a random area. In all experiments, we set batch size to 16 and take the prediction at the second inference step ($T = 2$) as the final output. In the testing, we resize the input images by setting a suitable height while keeping the aspect ratio.

The inference speed is tested with a batch size of 1, with a single TITAN X GPU in a single thread. We take the last output map as the final result and discard the others so as to achieve faster inference. We evaluate the accuracy and the speed of proposed method on the above datasets. In addition, to further indicate the advantages, we compare our network with previous state-of-the-art text detectors on model size and computational cost.

4.2. Ablation study on CTW1500

In the ablation study, we evaluate the proposed RFRN on the CTW1500 dataset to demonstrate the effectiveness of our framework. Table 1 lists the different settings in the ablation study. We first compare RFRN with its baseline and variants (Table 2) to demonstrate the effectiveness of the components of RFRN (section 4.2.1). Then we discuss the influence of three factors of RFRN (Table 4): 1) the times of predictions T (section 4.2.2); 2) the weighting in the training loss (section 4.2.3); and 3) the parameter sharing (section 4.2.4).

The **Baseline** model is a variant of PSENet, in which we lighten the backbone of PSENet from ResNet-50 to ResNet-18. For further reducing parameters, we apply 1×1 convolutional layers to squeeze the inner feature maps to have 64 channels instead 256. Our RFRN is built on the baseline, with the following variants compared:

RFRN ($T = 1$). This RFRN model only makes one prediction. Compared with the baseline, it contains extra four 3×3 convolutional layers.

RFRN ($T > 1$). This is the proposed RFRN model. We provide two strategies to train this model:

- **RFRN ($T > 1$, Shared, Last).** The loss of this model is based on the last output only.
- **RFRN ($T > 1$, Shared, All).** This model is trained with the supervision on all predictions.

RFRN ($T > 1$, Cascaded, All). This model is a variant of RFRN, which cascades different segment-heads to make multiple predictions in a unidirectional (non-recurrent) manner. It is trained with the supervision on all predictions.

Baseline + FPEM. This is the baseline model but with stacks of feature pyramid enhancement modules (FPEM) [27].

Baseline + PANet. This is the baseline model but replacing top-down path by double-path augmentation proposed in PANet [31].

In the ablation study on CTW1500, we train all the models for 37k iterations without extra data. In the testing, we scale the long side of test images to 1280.

4.2.1. Effectiveness of components of RFRN

Table 2 illustrates the effectiveness of components of our RFRN by showing the improvement over its baseline and variants. The baseline model (PSENet with ResNet-18 as the backbone) has less computational cost than the original PSENet (with ResNet-50 as the backbone), but it gets lower F-score (76.5% vs. 78.0%). Compared with the baseline model, RFRN ($T = 1$) has induced only 0.15M extra parameters, which are relatively negligible with the 11.5M parameters of the baseline, and it is unsurprising that RFRN ($T = 1$) has only minor improvement in the F-score (from 76.5% to 76.6%). Then, with the introduction of recurrent

Table 1
Experimental settings of the ablation study. "T" is the times of predictions in the training and testing. "β_t" are the loss weights: "All" denotes the weight of each β_t is 1; "Last" denotes the weight of the last output is set to T, and the others are set to 0. "SegHead" represents the model structure of recurrent segmentation: "Single" means that the model applies single segment-head for detection, which is also represented as "Shared"; "Cascaded" means that the model has multiple segment-heads which are cascaded.

Method	RFRM	T	β	SegHead
PSENet (ResNet-50)	-	1	-	Single
Baseline (PSENet, ResNet-18)	-	1	-	Single
RFRN ($T = 1$)	✓	1	-	Single
RFRN ($T > 1$, Shared, Last)	✓	1-4	Last	Single
RFRN ($T > 1$, Shared, All)	✓	1-4	All	Single
RFRN ($T > 1$, Cascaded, All)	✓	1-2	All	Cascaded
Baseline + FPEM	-	1	-	Single
Baseline + PANet	-	1	-	Single

refinement, RFRN ($T = 2$, Shared, Last/All) produces segmentation masks for two times and we take the final mask for evaluation. It shows that RFRN ($T = 2$, Shared, All) improves the F-score by 2 percent to 78.5%. With another prediction, RFRN ($T = 3$, Shared, All) improves the F-score to 79.8%. This indicates the effectiveness of the proposed recurrent refinement with parameter sharing. That is, when $T = 1$, the RFRN is still a unidirectional network, in the sense that the semantic information cannot be transmitted back to the previous neural nodes. In contrast, when $T > 1$, the network has the ability to refine its previous representation, and thus the learned features can be well enhanced.

We then compare RFRN with two similar work: the Path Aggregation Network (PANet) [31] and the Feature Pyramid Enhancement Module (FPEM) [27]. Both the FPEM and Path Aggregation enhance the multi-scale information by a top-to-down and a down-to-top augmentation. The results show the effectiveness of these double-path propagation enhancement, which improve the F-score of baseline by about 1%. However, their information propagation are still unidirectional and cannot refine the previous features by same parameters. Note that their F-score are also close to RFRN ($T = 2$, Shared, Last), which is 77.8%. With the supervision on all the predictions and the regularization on parameters, RFRN ($T = 2$, Shared, All) can improve the baseline by 2%. Even though RFRN has higher training cost (FLOPs), it can achieve lower inference cost by discarding non-essential prediction procedures. This will be detailed in Section 4.5.

We also apply our framework to other method. We combine RFRN ($T = 2$, Shared, All) with Differentiable Binarization Network (DBNet) [2], a real-time text detector. Table 3 shows the experimental results on ICDAR2015 dataset. All the models are trained under the same settings. With recurrent refinement, the F-measure of DBNet is improved by 2%, which indicates both the effectiveness and generality of our approach.

Note that the RFRN has a fixed amount of parameters no matter how many times it predicts, which shows the conciseness of RFRN. The F-score of RFRN ($T = 3$, Shared, All) is also higher than the original PSENet (with ResNet-50). Nonetheless, due to fewer channels in FPN (64 in RFRN, while 256 in PSENet), RFRN may discard some multi-scale information thus have lower recall than the original PSENet.

4.2.2. *Influence of times of predictions T*

The RFRN can leverage multiple predictions. However, the times of predictions need to be decided before training or testing. More predictions will slow down the inference and increase the network complexity. We prefer to find an appropriate setting of prediction times to achieve a good balance between accuracy and efficiency. Table 4 and Figure 3 show the ablation study on prediction times. In the training and testing phase, we set the maximum times of predictions to 3 and 4, respectively. Note that in this section, the times of predictions T_{test} in the testing phase may be different from the times T_{train} in training. From Table 4, we can find that the best F-score (79.8%) is achieved by RFRN (Shared, $T_{train} = 3$, $T_{test} = 3$, All). We notice that the FLOPs increase with the times of predictions, doubling the basic FLOPs when $T_{test} = 4$ (160.94G). For the balance of the speed and accuracy, we set both T_{train} and T_{test} to 2 when comparing RFRN with other state-of-the-art methods in the following sections.

We also notice that, although RFRN (Shared, $T_{train} = 3$, $T_{test} = 3$, All) outperforms the others, RFRN (Shared, $T_{train} = 3$, $T_{test} = 1$, All) performs (72.5%) worse than RFRN (Shared, $T_{train} = 1$, $T_{test} = 1$) (76.6%). We think the recurrent refinement may bring larger margin on feature learning. When using such a framework (e.g. $T_{train} = 3$, All), the network needs to produce optimal results on every output. However, the element-wise sum operation in RFRM (Figure 2) makes the network harder to achieve the global optimal. Thus the features of RFRN ($T_{train} = 3$, $T_{test} = 1$) is not as expressive as the one in the next prediction ($T_{test} = 2$). Therefore, the unidirectional network RFRN (Shared, $T_{train} = 1$, $T_{test} = 1$) outperforms the others when $T_{test} = 1$. But with extra refinement path of information, the weak features can be enhanced, so the F-score of RFRN (Shared, $T_{train} = 3$, $T_{test} = 3$, All) is boosted to 79.8%.

The last finding is that, when T_{train} is fixed, most of the F-scores rise with the T_{test} when $T_{test} \leq T_{train}$. When $T_{test} > T_{train}$, the F-score drops in most cases.

4.2.3. *Influence of weighting in the training loss*

The training loss of RFRN is calculated as the weighted sum of predictions. We compare the performance of RFRN under two weighting schemes: one is to calculate the training error from the final prediction only, denoted by "Last"; and the other is to calculate the average error of all predictions, denoted by "All", as shown in Table 4.

We observe that using all predictions can achieve better F-score than using the last prediction only, such as $T_{train} = 2$ (78.5% vs. 77.8%) and $T_{train} = 3$ (79.8% vs. 77.6%). The supervision on all the predictions guides the RFRM to produce more representative features in the early inference steps and

Table 2
Ablation study of model effectiveness on CTW1500. "P", "R" and "F" represent the precision, recall and F-score, respectively. "T" denotes the times of predictions in both training and testing. "Last" or "All" denotes that the training error is from the last output or every output. "Shared" or "Cascaded" represents that the segment-head for each prediction is the same or not. The FLOPS are calculated for the input of $1280 \times 1280 \times 3$.

Method	T	#Param (M)	GFLOPs	P	R	F
PSENet (ResNet-50)	1	28.628	469.02	80.6	**75.6**	78.0
Baseline (PSENet, ResNet-18)	1	11.498	80.55	81.4	72.2	76.5
RFRN ($T = 1$): Baseline + RFRM	1	11.646	85.38	82.0	71.9	76.6
RFRN ($T > 1$, Shared, Last): Baseline + RFRM + Shared + Last	2	11.646	110.56	82.5	73.6	77.8
RFRN ($T > 1$, Shared, All): Baseline + RFRM + Shared + All	2	11.646	110.56	84.3	73.5	78.5
RFRN ($T > 1$, Shared, All): Baseline + RFRM + Shared + All	3	11.646	135.75	**85.3**	74.9	**79.8**
RFRN ($T > 1$, Cascaded, All): Baseline + RFRM + Cascaded + All	2	12.054	110.56	83.5	71.3	76.9
Baseline + 2 * FPEM	1	11.881	89.422	82.1	72.6	77.0
Baseline + 3 * FPEM	1	12.374	96.357	83.8	71.9	77.4
Baseline + PANet	1	11.621	81.936	83.4	72.5	77.5

Table 3
Combination of RFRN and DBNet on ICDAR2015. "P", "R" and "F" represent the precision, recall and F-score, respectively. "T" denotes the times of predictions in both training and testing.

Method	T	P	R	F
DBNet (re-implement)	1	85.5	77.7	81.4
DBNet (re-implement) + RFRM (Shared, All)	2	**87.2**	**79.6**	**83.3**

thus improves the final accuracy. But it is also computationally most costly when training the "ALL" models.

4.2.4. Influence of parameter sharing

The idea of recurrent predictions was firstly proposed in AutoContext [41] and has inspired many recent approaches. The proposed RFRN reuses parameters for recurrent predictions, leading to a single segment-head for predictions. To investigate the advantages of parameter sharing, we implement a cascaded-version of the proposed RFRN, by stacking two different segment-heads. Each cascaded-head contains four RFRMs followed by the top-down augmentation layers to produce probability maps. The parameter-sharing of segment-head makes RFRN a recurrent segmentation network, while the cascaded segment-heads is still a unidirec-

Table 4
Ablation study of three factors on CTW1500: 1) times of predictions (T); 2) training supervision; 3) network structure (Shared vs. Cascaded). "P", "R" and "F" represent the precision, recall and F-score, respectively. The FLOPS are calculated for the input of $1280 \times 1280 \times 3$.

Method	Train-T	Test-T	Supervision	#Param(M)	GFLOPs	P	R	F
RFRN (Shared)	1	1	-	11.646	85.38	82.0	71.9	76.6
	2	1	Last	11.646	85.38	79.3	61.0	72.6
	2	2	Last	11.646	110.56	82.5	73.6	77.8
	2	3	Last	11.646	135.75	82.5	71.5	76.6
	3	1	Last	11.646	85.38	78.5	61.3	68.8
	3	2	Last	11.646	110.56	79.9	71.2	75.3
	3	3	Last	11.646	135.75	82.4	73.3	77.6
	3	4	Last	11.646	160.94	82.8	74.2	78.3
	2	1	All	11.646	85.38	82.3	64.6	72.4
	2	2	All	11.646	110.56	84.3	73.5	78.5
	2	3	All	11.646	135.75	79.7	73.9	76.7
	3	1	All	11.646	85.38	**89.6**	32.2	72.5
	3	2	All	11.646	110.56	87.9	63.4	73.7
	3	3	All	11.646	135.75	85.3	74.9	**79.8**
	3	4	All	11.646	160.94	80.9	**76.9**	78.8
RFRN (Cascaded)	2	1	All	11.646	85.38	82.6	70.5	76.1
	2	2	All	12.054	110.56	83.5	71.3	76.9

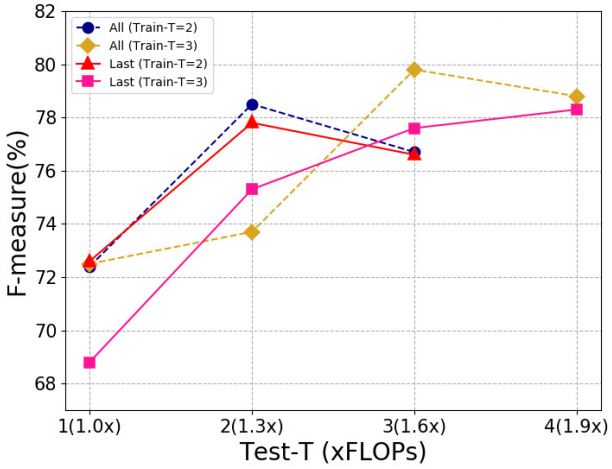


Figure 3: Influence of the times of predictions in training and testing. The figure shows the F-score of four models, which are trained with different T_{train} (Train-T) and loss functions (Last vs. All). T_{test} (Test-T) is the times of predictions during testing. "1.0x, 1.3x, 1.6x, 1.9x" denote the normalized FLOPs.

tional model. Table 4 shows the comparison results between these two schemes: parameter sharing of RFRN vs. parameter stacking of cascaded-RFRN. The cascaded-RFRN delivers poorer F-score on the CTW1500 dataset. When $T > 1$, the shared-RFRN ($T_{train} = 2$, $T_{test} = 2$, All) with 78.5% F-score is more accurate than the cascaded-RFRN ($T_{train} = 2$, $T_{test} = 2$, All) with 76.9% F-score, while having less parameters (11.646M vs. 12.054M).

The shared-RFRN can be regarded as a cascaded-RFRN adequately trained with a regularization enforcing the sharing of parameters. As all segment-heads should be very correlated when predicting segmentation masks, the parameters of each head can be more suitable to be shared than diverse. In our experiments, sharing these correlated parameters indeed helps the network converge better.

Moreover, as each head of the cascaded-RFRN is mainly trained by its own segmentation results, its F-score (76.9%) is close to that of the baseline model (76.5%) in Table 2.

4.3. Results on curved texts

To demonstrate the ability of RFRN in detecting curved texts, we conduct experiments on CTW1500 and Total-Text. In the testing, we fix the long side of input images to 1280 or 960 while keeping the aspect ratio. As we adopt the progressive scale expansion algorithm proposed in PSENet for post-processing, we follow their settings to post-process the image on smaller resolution in order to achieve higher speed. This setting is denoted as "4s", which means that the output size of our RFRN is 1/4 of the original image.

SCUT-CTW1500: Table 5 lists the experimental results. RFRN (1s-1280, $T = 2$) achieves 82.4% F-score with external pre-training (IC17), outperforming most previous state-of-the-art methods. RFRN (4s-960, $T = 2$) achieves 81.0% F-score, better than some recent methods such as SAE [24] (80.1%) and Wang et al. [19] (80.1%), while our model can run at 31 FPS, which is 10x and 3x faster than the latter two. DB (ResNet18) [2] obtains the highest speed due to their simple post-processing, but it has lower F-score (81.0%) even though they apply deformable convolution [28]. The best performer in F-score, CRAFT (83.5%), is 1.1% better than ours, but it has higher computational cost. More comparison on computational cost will be given in Section 4.5. In short, thanks to recurrent refinement framework, our method obtains competitive F-score with high inference speed.

Some qualitative results of the proposed method on CTW1500 are given in Figure 4. Note that the long side of these images is resized to 1280 while keeping the aspect ratio. RFRN ($T = 1$) fails to detect some tiny-scale cases (Figure 4(c)), but with a second prediction added, RFRN ($T = 2$) successfully detects these tiny-scale texts. Although RFRN has a smaller model size, it obtains more texts with recurrent refinement, while CRAFT and PSENet cannot handle such challenging cases. In fact, the CTW1500 dataset also contains many multi-oriented texts and has complex background. The proposed RFRN can handle these text instances, which indicates the robustness of the our method to text appearance and shape. More qualitative results on CTW1500 are shown in Figure 6(a).

Total-Text: Tabel 6 lists the results on Total-Text. The F-score of the finetuned model RFRN (1s-1280, $T = 2$, IC17) is 2% higher than the original PSENet (82.8% vs. 80.9%) and runs faster (10.8 vs. 3.9 in FPS). The recurrent segmentation method RFRN (1s-1280, $T = 2$) also outperforms most previous methods. With the well trained pixel embedding branch, Wang et al. [27] can run at the resolution of 640×640 and achieves the highest F-score and the second best FPS. Some detection results on Total-Text are given in Figure 6(b), where the word-level irregular texts can be correctly detected by RFRN.

4.4. Results on multi-oriented texts

For validating the generalization ability of RFRN in text detection, we conduct experiments on ICDAR2015 and ICDAR2017-MLT, both of which contain complex multi-oriented texts in scenes.

ICDAR2015: Table 7 lists the experimental results. In the testing, we scale the short side of input images to 1152 and 960. RFRN (4s-1152, $T = 2$) achieves 80.8% F-score when training from scratch. Pre-training on external data ICDAR2017-MLT can further improve the F-score to 85.3%, which is competitive with recent state-of-the-art methods. However, the post-processing algorithm is slow due to the large size of input images. With small input resolution (768), the RFRN achieves 20 FPS but its F-score drops 5%. To keep F-score on a competitive level, the input images are not resized to smaller resolution. Thus, our method on ICDAR2015 is not as fast as on CTW1500. The RFRN (4s-1152, $T = 2$) achieves 9.8 FPS and 85.3% F-score, which is still faster and more accurate than most of the previous methods. Compared with DB (ResNet18) [2], even though they

Table 5
The single-scale results on CTW1500. "Ext." means external data; * indicates the results from [35]; "1s" and "4s" mean that the output size is 1/1 and 1/4 of the input image.

Method	Ext.	CTW1500			
		Precision	Recall	F-score	FPS
SegLink* [33]	-	42.3	40.0	40.8	10.7
EAST* [21]	-	78.7	49.1	60.4	21.2
CTD+TLOC [35]	-	77.4	69.8	73.4	13.3
Liu et al. [20]	-	79.7	**79**	79.4	1.6
Wang et al. [27]	-	**84.6**	77.7	**81.0**	**39.8**
PSENet-1s [11]	-	80.6	75.6	78.0	3.9
RFRN-1s (1280, $T = 2$)	-	84.3	73.5	78.5	10.8
RFRN-4s (1280, $T = 2$)	-	83.0	74.1	78.0	18.7
RFRN-4s (960, $T = 2$)	-	81.2	72.0	76.3	31.5
TextSnake [9]	✓	67.9	**85.3**	75.6	-
LOMO [42]	✓	**89.2**	69.6	78.4	-
CSE [40]	✓	81.1	76.0	78.4	2.6
Wang et al. [19]	✓	80.1	80.2	80.1	10.0
SAE [24]	✓	82.7	77.8	80.1	3
MSR [10]	✓	85.0	78.3	81.5	-
CRAFT [25]	✓	86.0	81.1	83.5	-
ICG [18]	✓	82.8	79.8	81.3	-
Wang et al. [27]	✓	86.4	81.2	**83.7**	39.8
TextMountain [6]	✓	81.3	82.4	81.9	-
TextField [6]	✓	83.0	79.8	81.4	-
DB (ResNet18) [2]	✓	84.8	77.5	81.0	**55**
R-Net [7]	✓	74.6	71.0	72.8	-
PSENet-1s [11]	✓	84.8	79.7	82.2	3.9
RFRN-1s (1280, $T = 2$)	✓(SYN)	85.4	77.5	81.2	10.8
RFRN-1s (1280, $T = 2$)	✓(IC17)	84.9	80.0	82.4	10.8
RFRN-4s (1280, $T = 2$)	✓(IC17)	84.5	79.5	81.9	18.7
RFRN-4s (960, $T = 2$)	✓(IC17)	85.0	77.3	81.0	31.5

Table 6
The single-scale results on Total-Text. The rest captions are as in Table 5.

Method	Ext.	Total-Text			
		Precision	Recall	F-score	FPS
PSENet-1s [11]	-	82.7	74.5	78.4	3.9
Liu et al. [20]	-	79.1	74.5	76.7	-
Wang et al. [27]	-	**88.0**	**79.4**	**83.5**	**39.6**
RFRN-1s (1280, $T = 2$)	-	80.9	77.2	79.0	10.8
RFRN-4s (1280, $T = 2$)	-	80.3	76.4	78.3	18.7
RFRN-4s (960, $T = 2$)	-	79.9	70.2	74.7	31.5
TextSnake [9]	✓	82.7	74.5	78.4	-
CSE [40]	✓	81.4	79.1	80.2	2.4
LOMO [42]	✓	87.6	79.3	83.3	-
Wang et al. [19]	✓	80.9	76.2	78.5	-
MSR [10]	✓	83.8	74.8	79.0	-
CRAFT [25]	✓	87.6	79.9	83.6	-
ICG [18]	✓	82.1	80.9	81.5	-
TextField [6]	✓	81.2	79.9	80.6	-
DB (ResNet18) [2]	✓	88.3	77.9	82.8	**50**
Wang et al. [27]	✓	**89.3**	81.0	**85.0**	39.6
PSENet-1s [11]	✓	84.0	78.0	80.9	3.9
RFRN-1s (1280, $T = 2$)	✓(SYN)	82.9	80.9	81.9	10.8
RFRN-1s (1280, $T = 2$)	✓(IC17)	84.3	**81.4**	82.8	10.8
RFRN-4s (1280, $T = 2$)	✓(IC17)	84.6	79.9	82.2	18.7
RFRN-4s (960, $T = 2$)	✓(IC17)	84.9	74.5	79.3	31.5

Table 7
The single-scale results on ICDAR2015. The rest captions are as in Table 5.

Method	Ext.	ICDAR2015			
		Precision	Recall	F-score	FPS
EAST [21]	-	83.6	73.5	78.2	13.2
RRPN [16]	-	82.0	73.0	77.0	-
Wang et al. [27]	-	82.9	77.8	80.3	**26.1**
PSENet-1s [11]	-	81.5	**79.7**	80.6	1.6
RFRN-4s (1152, $T = 2$)	-	**84.5**	77.3	**80.8**	9.8
RFRN-4s (960, $T = 2$)	-	82.5	75.7	79.0	13.3
RFRN-4s (768, $T = 2$)	-	81.7	72.2	76.6	20.5
SegLink [33]	✓	73.1	76.8	75.0	5.5
RRD [17]	✓	85.6	79.0	82.2	6.5
Lyu et al. [26]	✓	**94.1**	70.7	80.7	3.6
TextSnake [9]	✓	84.9	80.4	82.6	1.1
MSR [10]	✓	86.6	78.4	82.3	4.3
SAE [24]	✓	88.3	85.0	86.6	-
CRAFT [25]	✓	89.8	84.3	86.9	-
Liu et al. [20]	✓	86.6	**87.6**	**87.1**	-
ICG [18]	✓	83.7	80.3	82.0	7.1
CSE [40]	✓	92.3	79.9	85.7	-
Wang [27]	✓	84.0	81.9	82.9	26.1
TextMountain [6]	✓	89.5	83.1	86.2	9.7
DB [2]	✓	86.8	78.4	82.3	**48**
R-Net [7]	✓	88.7	82.8	85.6	21.4
PSENet-1s [11]	✓	86.9	84.5	85.7	1.6
RFRN-4s (1152, $T = 2$)	✓(SYN)	84.8	80.2	82.4	9.8
RFRN-4s (1152, $T = 2$)	✓(IC17)	88.1	82.6	85.3	9.8
RFRN-4s (960, $T = 2$)	✓(IC17)	85.8	79.8	82.7	13.3
RFRN-4s (768, $T = 2$)	✓(IC17)	85.0	65.6	79.9	20.5

Table 8
The single-scale results on ICDAR2017-MLT. The rest captions are as in Table 5.

Method	Ext.	IC17-MLT			
		Precision	Recall	F-score	FPS
SARI_FDU_RRPN_V1 [26]	-	71.2	55.6	62.4	-
Sensetime_OCR [26]	-	56.9	**69.4**	62.6	-
SCUTDLVlab1 [26]	-	80.3	54.5	65.0	-
e2e_ctc01_multi_scale [26]	-	79.8	61.2	69.3	-
Lyu et al. [26]	✓	**83.8**	55.6	66.8	-
CRAFT [25]	✓	80.6	68.2	73.9	-
LOMO[42]	✓	80.2	67.2	73.1	-
TextMountain [6]	✓	82.8	68.1	**74.7**	-
DB (ResNet18) [2]	✓	81.9	63.8	71.7	**41**
PSENet-1s [11]	-	73.8	68.2	70.9	-
RFRN-1s ($T = 2$)	-	77.5	64.9	70.6	-

are the fastest method, the RFRN (4s-1152, $T = 2$) is 3% higher in F-score. Liu et al. [20] achieves both the highest recall and F-score with the tightness prior framework, but the two-stage methods can be more time-consuming due to its processing of proposals.

Some detection results on ICDAR2015 of the proposed method are shown in Figure 5, along with the results of recent models. The basic model RFRN ($T = 1$) fails to detect large scale texts (Figure 5(c)). When predicting a second time, the RFRN ($T = 2$) captures the missing texts. Because of the complex background (such as the most left and right samples in Figure 5) and the limited receptive field of unidirectional network, these large scale texts cannot be well detected by segmentation-based methods such as CRAFT and PSENet. We also notice that the RFRN can handle non-Latin text line. The results indicate that RFRN enlarges the receptive field of the network, and has the advantages in dealing with various scales and background complexity. The results on multi-oriented texts also indicate the good generalization ability of recurrent refinement. More qualitative results on ICDAR2015 are given in Figure 6(c).

ICDAR2017-MLT: Table 8 shows the results on

Table 9
The network processing cost of the proposed RFRN with/without discarding the non-essential prediction procedures. The FLOPS are calculated for the input of $1280 \times 1280 \times 3$.

Method	T_{test}	GFLOPs	GFLOPs after discard
RFRN	1	85.38 (1.0x)	85.38 (1.0x)
	2	110.56 (1.3x)	95.34 (1.1x)
	3	135.75 (1.6x)	105.42 (1.2x)
	4	160.94 (1.9x)	115.44 (1.3x)

ICDAR2017-MLT dataset. We only use the ICDAR2017-MLT data to train our model. In the testing, we scale up the short side of input images to double their original sizes. This dataset has the largest test set of scene texts, covering many challenging cases such as diverse scales and blurred texts. Although a larger backbone is more flexible to handle these difficult images, the RFRN (1s, $T = 2$) still achieves 70.6% F-score, which is closed to PSENet, while our RFRN uses a lighter network. Qualitative results on ICDAR2017-MLT are provided in Figure 6(d), where the proposed RFRN successfully detects multilingual texts of arbitrary orientations and variant scales.

4.5. Computational cost

As illustrated in Table 5, the total inference time of RFRN (1s-1280, $T = 2$, ResNet18) is about 90 ms (10.8 FPS), which includes the time cost of network and post-processing. We achieve that speed by: 1) simplifying the network processing; 2) optimizing the post-processing algorithm. Firstly, in the network processing, we discard the non-essential prediction procedures and only maintain the last one. The network processing time can be reduced by about 10 ms. In Table 9, we list more information about the speed boost obtained from such a discarding for different T_{test}. Secondly, for faster post-processing, we optimize the code of PSE algorithm [11], which generates text bounding boxes from the output segmentation masks. The original PSE algorithm is inefficient due to its excessive computation caused by: 1) traversing non-essential area in the output masks; 2) converting a 2D list to a NumPy array. We first filter the output masks with a 3×3 cross kernel, thus it only needs to traverse the text border. Then we rewrite the c++ code of PSE to directly return a NumPy array. The post-processing time is reduced by about 55 ms. Note that the original post-processing time takes about 100 ms,

To further demonstrate the advantages of our method, we provide a comprehensive comparison of efficiency with previous state-of-the-art methods. The inference time of a text detection method is decided by three factors: 1) input resolution; 2) network cost; 3) post-processing. As our study mainly focuses on the network architecture, the total FLOPs and model size of compared methods are given in Table 10, some obtained by using the source code provided by their authors. Note that the FLOPs is calculated at the same input resolution. We also report the backbone, as some methods do not provide code or computational cost measures.

Most recent methods adopt a heavy backbone, such as ResNet-50 and VGG-16, to achieve high F-score. Instead, the proposed RFRN uses ResNet18 as the backbone and reuses the parameters to become a deeper network with less parameters. With the parameter-reuse strategy, RFRN (1s, $T = 2$) achieves competitive F-score (82.4%) with smaller size. Although CRAFT has higher F-score (83.5%), our FLOPs and model size is 5x faster and nearly 2x smaller than CRAFT. Although the same post-processing algorithm in PSENet, our RFRN produces better F-score than PSENet (82.2%), with 2x less in parameters and 4x faster in FLOPS than PSENet. This indicates that the proposed RFRN achieves a good balance between accuracy and speed, among the recent methods.

Although making additional predictions, the proposed RFRN induces no extra parameters and only takes additional 0.1x computation time. This indicates the parameter and computational efficiency of RFRN. Moreover, for mobile applications, it merits investigation of the combination of RFRN and lightweight networks like ShuffleNet [5] and MobileNet [43]. We adopt MobileNetV3 as the backbone for comparative purposes (denoted by "MobileV3"). RFRN (1s, $T = 2$, MobileV3) only has 0.83M parameters and achieves 76.3% F-score, which is 0.7% higher than TextSnake.

4.6. Limitation

Although RFRN performs well in the previous experiments, it still can fail to detect some difficult images, as illustrated in Figure 7 on font style, text-like textures and large character space, which are nonetheless also challenging to other state-of-the-art methods.

5. Conclusions and Future Work

This paper proposes a new scene text detection framework called the recurrent feature refinement network (RFRN), which can enhance multi-scale feature maps iteratively and has low computational cost. The proposed RFRN can handle both arbitrary-shaped and multi-orientation scene text. Benefiting from recurrent predictions, RFRN extends the information propagating path and provides a solution to reduce computational cost of the network while maintaining high accuracy. On four benchmark datasets, CTW1500, Total-Text, ICDAR2015 and ICDAR2017-MLT, our approach produces competitive detection accuracy to the recent state-of-the-art methods, even though it uses only a lightweight backbone (ResNet18). On CTW1500, our approach is faster than most text detectors and achieves a good balance between accuracy and speed. The future work is to further improve the flexibility of recurrent segmentation like with deformable convolutional network [28] and investigate the combination of RFRN with faster post-processing techniques like the differentiable binarization explored in [2].

References

[1] S. Zhang, Y. Liu, L. Jin, Z. Wei, C. Shen, OPMP: An omni-directional pyramid mask proposal network for arbitrary-shape scene text detection, IEEE Transactions on Multimedia (2020).

Table 10
Comparison of efficiency on CTW1500. The FLOPS are calculated for the input of $1280 \times 1280 \times 3$.

Method	Backbone	Params(M)	GFLOPS	F-score	FPS
TextSnake [9]	VGG16	19.12	583.70	75.6	-
CRAFT [25]	VGG16	20.77	584.40	**83.5**	-
Wang et al. [19]	VGG16	-	-	80.1	10
TextMountain [6]	VGG16	-	-	81.9	-
R-Net [7]	VGG16	-	-	72.8	-
CSE [40]	ResNet34	-	-	78.4	2.6
LOMO [42]	ResNet50	-	-	78.4	4.4
SAE [24]	ResNet50	-	-	80.1	3
MSR [10]	ResNet50	-	-	81.5	-
PSENet-1s (1280) [11]	ResNet50	28.63	469.02	82.2	3.9
PSENet-4s (1280) [11]	ResNet50	28.63	469.02	79.9	8.4
PSENet-4s (960) [11]	ResNet50	28.63	469.02	78.3	21.7
RFRN-1s (1280, $T = 1$)	ResNet18	11.65	85.38	80.1	12.7
RFRN-1s (1280, $T = 2$)	ResNet18	11.65	95.34	82.4	10.8
RFRN-4s (1280, $T = 2$)	ResNet18	11.65	95.34	81.9	18.7
RFRN-4s (960, $T = 2$)	ResNet18	11.65	95.34	81.0	**31.5**
RFRN-1s (1280, $T = 2$)	MobileV3	**0.83**	**50.70**	76.3	14.1

[2] M. Liao, Z. Wan, C. Yao, K. Chen, X. Bai, Real-time scene text detection with differentiable binarization, in: AAAI Conference on Artificial Intelligence (AAAI), 2020, pp. 11474–11481.
[3] J. Liu, Q. Zhong, Y. Yuan, H. Su, B. Du, SemiText: Scene text detection with semi-supervised learning, Neurocomputing (2020).
[4] Y. Wang, H. Xie, Z.-J. Zha, M. Xing, Z. Fu, Y. Zhang, ContourNet: Taking a further step toward accurate arbitrary-shaped scene text detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11753–11762.
[5] W. He, X.-Y. Zhang, F. Yin, Z. Luo, J.-M. Ogier, C.-L. Liu, Realtime multi-scale scene text detection with scale-based region proposal network, Pattern Recognition 98 (2020) 107026.
[6] Y. Zhu, J. Du, TextMountain: Accurate scene text detection via instance segmentation, Pattern Recognition (2020) 107336.
[7] Y. Wang, H. Xie, Z.-J. Zha, Y. Tian, Z. Fu, Y. Zhang, R-Net: A relationship network for efficient and accurate scene text detection, IEEE Transactions on Multimedia (2020).
[8] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2117–2125.
[9] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, C. Yao, TextSnake: A flexible representation for detecting text of arbitrary shapes, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 20–36.
[10] C. Xue, S. Lu, W. Zhang, MSR: multi-scale shape regression for scene text detection, arXiv preprint arXiv:1901.02596 (2019).
[11] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, S. Shao, Shape robust text detection with progressive scale expansion network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 9336–9345.
[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, SSD: Single shot multibox detector, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2016, pp. 21–37.
[13] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (6) (2017) 1137–1149.
[14] M. Liao, B. Shi, X. Bai, X. Wang, W. Liu, TextBoxes: A fast text detector with a single deep neural network, in: AAAI Conference on Artificial Intelligence (AAAI), 2017, pp. 4161–4167.
[15] M. Liao, B. Shi, X. Bai, TextBoxes++: A single-shot oriented scene text detector, IEEE Transactions on Image Processing 27 (8) (2018) 3676–3690.
[16] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, X. Xue, Arbitrary-oriented scene text detection via rotation proposals, IEEE Transactions on Multimedia 20 (11) (2018) 3111–3122.
[17] M. Liao, Z. Zhu, B. Shi, G.-s. Xia, X. Bai, Rotation-sensitive regression for oriented scene text detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 5909–5918.
[18] J. Tang, Z. Yang, Y. Wang, Q. Zheng, Y. Xu, X. Bai, SegLink++: Detecting dense and arbitrary-shaped scene text by instance-aware component grouping, Pattern Recognition 96 (2019) 106954.
[19] X. Wang, Y. Jiang, Z. Luo, C.-L. Liu, H. Choi, S. Kim, Arbitrary shape scene text detection with adaptive text region representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 6449–6458.
[20] Y. Liu, L. Jin, C. Fang, Arbitrarily shaped scene text detection with a mask tightness text detector, IEEE Transactions on Image Processing 29 (2019) 2918–2930.
[21] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, J. Liang, EAST: an efficient and accurate scene text detector, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5551–5560.
[22] W. He, X.-Y. Zhang, F. Yin, C.-L. Liu, Multi-oriented and multi-lingual scene text detection with direct regression, IEEE Transactions on Image Processing 27 (11) (2018) 5406–5419.
[23] Y. Xu, Y. Wang, W. Zhou, Y. Wang, Z. Yang, X. Bai, TextField: Learning a deep direction field for irregular scene text detection, IEEE Transactions on Image Processing 28 (11) (2019) 5566–5579.
[24] Z. Tian, M. Shu, P. Lyu, R. Li, C. Zhou, X. Shen, J. Jia, Learning shape-aware embedding for scene text detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4234–4243.
[25] Y. Baek, B. Lee, D. Han, S. Yun, H. Lee, Character region awareness for text detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, pp. 9357–9366.
[26] P. Lyu, C. Yao, W. Wu, S. Yan, X. Bai, Multi-oriented scene text detection via corner localization and region segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7553–7563.
[27] W. Wang, E. Xie, X. Song, Y. Zang, W. Wang, T. Lu, G. Yu, C. Shen,

Efficient and accurate arbitrary-shaped text detection with pixel aggregation network, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8440–8449.

[28] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 764–773.

[29] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2016, pp. 630–645.

[30] S. Karen, Z. Andrew, Very deep convolutional networks for large-scale image recognition, in: Proceedings of International Conference on Learning Representations, 2015.

[31] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path aggregation network for instance segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8759–8768.

[32] K.-H. Kim, S. Hong, B. Roh, Y. Cheon, M. Park, PVANet: Deep but lightweight neural networks for real-time object detection, arXiv preprint arXiv:1608.08021 (2016).

[33] B. Shi, X. Bai, S. Belongie, Detecting oriented text in natural images by linking segments, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2550–2558.

[34] F. Sheng, Z. Chen, W. Zhang, B. Xu, PyrBoxes: An efficient multi-scale scene text detector with feature pyramids, Pattern Recognition Letters 125 (2019) 228–234.

[35] Y. Liu, L. Jin, S. Zhang, S. Zhang, Detecting curve text in the wild: New dataset and new solution, arXiv preprint arXiv:1712.02170 (2017).

[36] C.-K. Ch'ng, C. S. Chan, C.-L. Liu, Total-Text: toward orientation robustness in scene text detection, International Journal on Document Analysis and Recognition (IJDAR) 23 (1) (2020) 31–52.

[37] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al., ICDAR 2015 competition on robust reading, in: Proceedings of International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2015, pp. 1156–1160.

[38] Icdar 2017 robust reading competition., https://rrc.cvc.uab.es/?ch=8 (2017).

[39] A. Gupta, A. Vedaldi, A. Zisserman, Synthetic data for text localisation in natural images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2315–2324.

[40] Z. Liu, G. Lin, S. Yang, F. Liu, W. Lin, W. L. Goh, Towards robust curve text detection with conditional spatial expansion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 7269–7278.

[41] Z. Tu, X. Bai, Auto-context and its application to high-level vision tasks and 3d brain image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (10) (2009) 1744–1757.

[42] C. Zhang, B. Liang, Z. Huang, M. En, J. Han, E. Ding, X. Ding, Look more than once: An accurate detector for text of arbitrary shapes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, pp. 10544–10553.

[43] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al., Searching for MobileNetV3, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019, pp. 1314–1324.

(a) CRAFT (VGG16) [25]

(b) PSENet (ResNet50) [11]

(c) RFRN (ResNet18, $T = 1$)

(d) RFRN (ResNet18, $T = 2$)

(e) Ground Truth

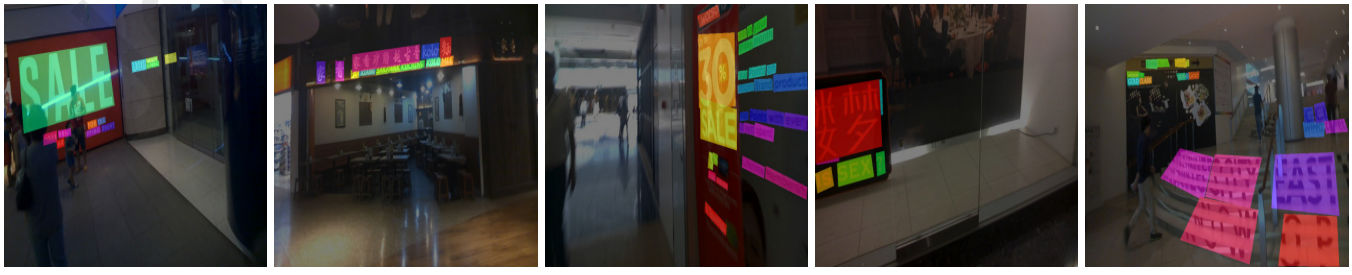
Figure 4: Comparative curve text detection results of CRAFT, PSENet and RFRN on the CTW1500 dataset. The detection results of each method are given in green polygon. Text instances in Ground Truth are represented in different colors.

(a) CRAFT (VGG16) [25]

(b) PSENet (ResNet50) [11]

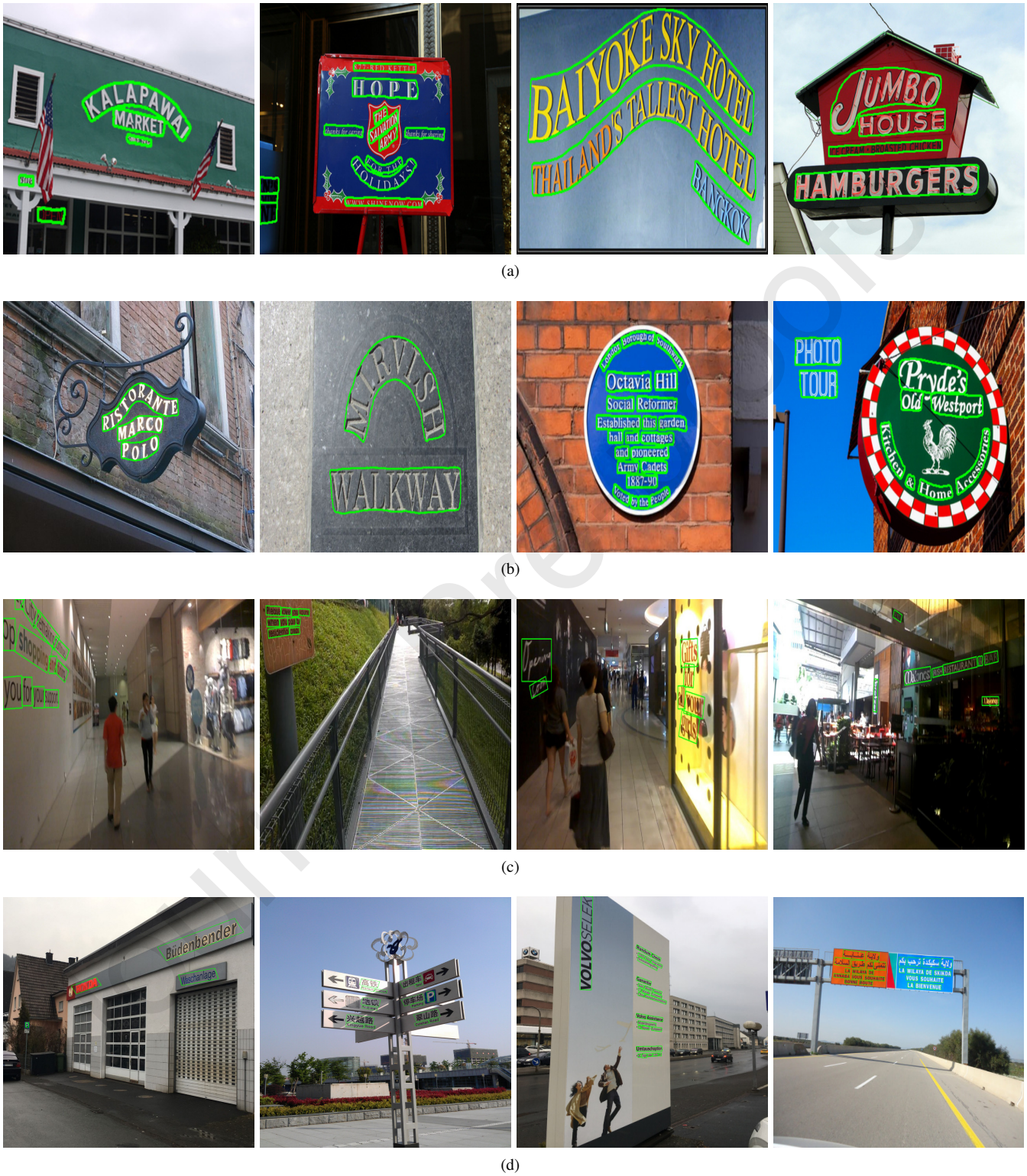
(c) RFRN (ResNet18,$T = 1$)

(d) RFRN (ResNet18, $T = 2$)



(e) Ground Truth

Figure 5: Text detection results of CRAFT, PSENet and RFRN on the ICDAR2015 dataset. The detection results of each method are given in green polygon. Text instances in Ground Truth are represented in different colors.



(a)

(b)

(c)

(d)

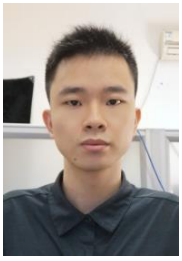
Figure 6: Qualitative results by the proposed method on (a) SCUT-CTW500, (b) Total-Text, (c) ICDAR2015, and (d) ICDAR2017-MLT.

(a) Font style

(b) Text-like textures

(c) Large space

Figure 7: Failure examples of (a) font style, (b) text-like textures, and (c) large space.



Guanyu Deng received the B.S degree from Beijing University of Posts and Telecommunications, in 2018. He is currently pursuing the master's degree at Beijing University of Posts and Telecommunications. His current research interests include scene text detection and recognition.

Yue Ming received the B.S. degree in Communication Engineering, and the M.Sc degree in Human-Computer Interaction Engineering, and Ph.D. degree in Signal and Information Processing from Beijing Jiaotong University, China, in 2006, 2008, and 2013.

She worked as a visiting scholar in Carnegie Mellon University, U.S., between 2010 and 2011. Since 2013, she has been working as a faculty member at Beijing University of Posts and Telecommunications. Her research interests are in the areas of biometrics, computer vision, computer graphics, information retrieval, pattern recognition, etc.

Jing-Hao Xue received the Dr.Eng. degree in signal and information processing from Tsinghua University in 1998 and the Ph.D. degree in statistics from the University of Glasgow in 2008. He is an Associate Professor in the Department of Statistical Science at University College London. His research interests include statistical classification, high-dimensional data analysis, computer vision and