TextTriangle: An end-to-end textspotter with piecewise linear alignment

Hui Xu (xuhui@cigit.ac.cn)
Chongqing Institute of Green and Intelligent Technology
Qiu-Feng Wang
Xi'an Jiaotong-Liverpool University
Zhenghao Li
Chongqing Institute of Green and Intelligent Technology
Yu Shi
Chongqing Institute of Green and Intelligent Technology
Xiang-Dong Zhou
Chongqing Institute of Green and Intelligent Technology

Research Article

Keywords: scene text spotting, scene text detection, scene text recognition, text feature alignment, end-to-end training

Posted Date: June 14th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1743583/v1

TextTriangle: An end-to-end textspotter with piecewise linear alignment

Hui Xu[1,2], Qiu-Feng Wang[3], Zhenghao Li[1], Yu Shi[1] and Xiang-Dong Zhou[1]

[1]Chongqing Institute of Green and Intelligent Technology,Chinese Academy of Sciences.
[2]Chongqing School, University of Chinese Academy of Sciences (UCAS Chongqing).
[3]Department of Intelligent Science, School of Advanced Technology, Xi'an Jiaotong-Liverpool University (XJTLU), China.

Contributing authors: xuhui@cigit.ac.cn; qiufeng.wang@xjtlu.edu.cn; lizh@cigit.ac.cn; shiyu@cigit.ac.cn; zhouxiangdong@cigit.ac.cn;

Abstract

Scene text detection and recognition has attracted increasing research attention recently, especially for text of arbitrary shapes. In most of text spotting methods, text feature alignment is a key component to connect the detector and recognizer for end-to-end training. Existing alignment methods can be roughly categorized into those based on global consistent transformations and based on character-level classification. However, these methods either are unreliable for heavily deformed text or ignore contextual information in recognition. In this paper, we propose a novel textspotter named TextTriangle, which detects and recognizes text of arbitrary shapes in an end-to-end manner without character-level annotations. In TextTriangle, a text instance is described as a sequence of ordered triangles attached to each other. Based on this representation, a new PiecewiseAlign layer is designed to accurately extract features of the text instance with arbitrary shapes, which is the key to make the framework end-to-end trainable. Compared with the methods based on global consistent transformations, PiecewiseAlign adopts piecewise linear transformation for feature calculation. Experiments show that PiecewiseAlign is superior to TPS-based method in text alignment, and TextTriangle achieves competitive performance on standard scene text benchmarks.

Keywords: scene text spotting, scene text detection, scene text recognition, text feature alignment, end-to-end training.

1 Introduction

Automatic reading text in the wild has attracted increasing attention on account of its vast applications such as traffic signs recognition and automatic paper reading. For this task, designed systems generally include text detection and recognition, where text instances are located by text detection, then converted into character sequences by text recognition. Scene text spotting aims to combine both text detection and recognition for end-to-end training. In spite of colossal progress has been made recently [6, 19, 36], detecting and recognizing text in the wild are still challenging due to arbitrary shapes, aspect ratios and scales.

In recent years, scene text spotting methods tend to integrate text detection and recognition into a unified framework. Several such approaches [12, 19] are proposed to perform horizontal/orientated text detection and recognition. However,

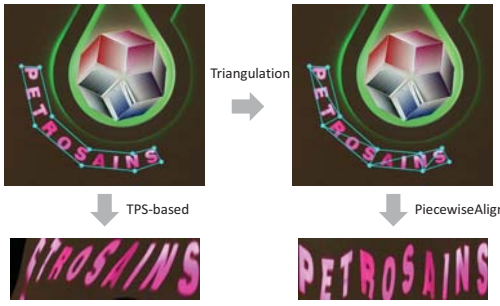


Fig. 1 The comparison of PiecewiseAlign and TPS-based alignment, which warp the original ground truth into rectangular shape.

the text description of rectangles or quadrangles is insufficient to represent arbitrary-shaped text instances in the wild. To tackle this issue, Mask TextSpotter [24] and CharNet[44] are proposed to detect and recognize the text of arbitrary shapes and achieve the state-of-the-art performance. Nevertheless, the required extra character-level annotations in the training process call for more manual efforts and higher labelling precision. Qin *et al.* [30] propose RoI masking to recognize the masked text instances directly, but the results may easily be affected by outlier pixels. Wang *et al.* [39] adopt a coarse-to-fine strategy to localize text boundary points and rectified the arbitrary-shaped text by Thin-Plate-Spline (TPS) transformation. However, the proposed method verifies that TPS-based alignment is not the best choice, especially for the text that is heavily deformed as shown in Fig. 1.

Inspired by TextSnake [22], ScRN [47] and TextPerceptron[27], we propose a novel textspotter named TextTrianlge as shown in Fig. 2. The representation in TextSnake that describes text with a series of ordered and overlapping disks [22] is not convenient to connect with the text recognition branch. In ScRN, TPS transformation is employed to rectify the arbitrary-shaped text features to axis-aligned features. Global consistent transformations such as affine transformation, perspective transformation and TPS transformation use the same transformation parameters for the whole image to be aligned. However, each part of the text may have different degrees of deformation and the text alignment based on global consistent transformations tends to aggravate the deformation, especially for the heavily deformed text as shown in Fig.1. So we propose the piecewise text alignment to learn the different linear transformation parameters of each part of the text. A linear transformation can be defined as a mapping of three points from the source image to the target image. To this end, we use triangles as the basic geometric units of irregular text. To be specific, the proposed text spotter expresses a text instance with a sequence of ordered triangles attached to each other, which is named TextTriangle. In other words, we divide the irregular text into several triangular patches and employ different transformations parameters to each patch separately as shown in Fig.4. Based on this description, PiecewiseAlign is proposed to precisely calculate convolutional features of arbitrary-shaped text instances. Compared to global consistent transformations, piecewise linear transformation causes alignment to occur within predefined triangular regions, which can achieve more accurate and stable text alignment.

In TextTriangle, a text detector first locates the text boundary points according to the predicted geometric attributes of the text, which include center line, text height, character orientation and text orientation. Then, PiecewiseAlign is adopted to extract axis-aligned text features for connecting the recognition branch. Finally, the CNN based text recognizer predicts the sequenced result using an attention-based decoder. The framework of our proposed textspotter is an end-to-end trainable system that free from character-level annotations. In summary, our contributions are threefold:

- A novel end-to-end scene textspotter named TextTriangle is proposed, which is based on the piecewise alignment for the first time and is flexible for the arbitrary-shaped text.
- A novel PiecewiseAlign layer is designed to accurately extract features of arbitrary-shaped text and unify the text detection and recognition into an overall structure, which is the first time to describe text with a series of ordered and adjacent triangles.
- PiecewiseAlign is verified to be more effective than TPS-based alignment. Meanwhile, TextTriangle has achieved competitive performance on text benchmarks.

The rest of the paper is structured as follows: Section 2 briefly introduces the related works

of scene text detection, recognition and spotting. In Section 3, we depict the proposed text spotter in details, including the text detector, PiecewiseAlign and the recognizer. The Experimental evaluation is shown in Section 4, while the conclusion is in Section 5.

2 Related Work

Currently, deep-learning-driven approaches have achieved great success in both scene text detection and recognition. We review the representative methods based on deep-learning from the following three aspects: scene text detection, recognition and spotting [20, 23].

Scene Text Detection methods can be roughly classified into regression-based [13, 41, 42] and segmentation-based methods [3, 29, 37, 45], of which the segmentation-based methods perform better in arbitrary-shaped text detection. Liao *et al.*[15] propose Differentiable Binarization (DB) module to adaptively set the thresholds for binarization in detection network. PSENet [40] gradually expands kernels at certain scale to split the close text instances. The Fourier Contour Embedding(FCE) method [46] models text instance contours in the frequency domain instead of the spatial domain via the Fourier transformation. The PCR method [26] enriches the feature representations of text contours by considering both the cyclicity in geometric topology and the contexts in semantic. TextSnake [22] describes text with a series of ordered, overlapping disks located at the center axis of text region. However, the segmented arbitrary-shaped text is not easy to be used for subsequent text recognition. The proposed method calculates the text boundary points based on the segmentation results, which is beneficial to text alignment and recognition.

Scene Text Recognition is commonly considered as a sequence-to-sequence recognition problem, where recurrent neural networks (RNNs) are taken. The recognition methods mainly contain CTC-based methods[9, 32] and encoder-decoder frameworks with attention mechanisms[28, 43]. Shi *et al.* [33] propose an end-to-end text recognition framework based on CTC. ASTER [34] employs a rectification module before recognition and predicts the character sequence with an attention-based decoder. Recently, the advanced frameworks based on various attention mechanisms are proposed [18, 38], which encode more character information explicitly or implicitly. The method [1] utilizes semantic information by proposing a multi-stage multi-scale attentional decoder that performs joint visual-semantic reasoning. Obviously, the attention-based methods performs better than CTC-base ones in arbitrary-shaped text recognition. In addition, rectification-based methods are usually involved to rectify the text shapes in the irregular scene text recognition. Among them, the geometric rectification method based on the TPS transformation is becoming the most representative method for arbitrary-shaped text rectification.

Scene Text Spotting attempts to unify text detection and recognition into an end-to-end trainable framework. The methods [14, 30, 39] based on Region Proposal Network (RPN) [31] generate proposals and extract ROI features for detection and recognition. The manually pre-designed anchors of RPN cannot easily match text instances with arbitrary shapes. FOTS [19] proposes RoIRotate to perform text alignment on horizontal/oriented texts, where the core of RoIRotate is an affine transformation. Mask TextSpotter v3 [16] and Multiplexed TextSpotter [10] perform text recognition based on arbitrary-shaped masked text features, which are strongly affected by the quality of segmentation accuracy. TextDragon [6] describes the shape of text with a series of quadrangles, and proposes RoISlide to connect text detection and recognition. ABCNet [21] adaptively fits arbitrarily-shaped text by a parameterized Bezier curve and explores a BezierAlign layer for extracting features of a text instance. SwinTextSpotter[11] and TESTR[49] proposed Transform-based methods for text detection and recognition. Methods based on text rectification [27, 39] generate fiducial points and utilize TPS transformation to rectify irregular text instances to regular ones. Compared with affine transformation that can only handle horizontal/oriented texts, TPS transformation can deal with arbitrary-shaped texts. However, it tends to cause part of the text to run out of the defined area or result in more severe deformation. The proposed method adopts PiecewiseAlign to rectify the text, which is verified to have better performance than global transformations.

3 Methodology

The proposed textspotter is depicted in Fig. 2. First, a stem network is used to extract visual features on the input image. After that, the text detector is employed to predict the attributes of the arbitrary-shaped text, based on which the boundary points are calculated and sampled. Then, the proposed PiecewiseAlign is used to output axis-aligned features of arbitrary-shaped text region, which is the key to unify text detection and recognition into an end-to-end pipeline. Finally, the text recognizer predicts the sequenced results with a lightweight network.

3.1 Text Detection

Compared to the methods based on regression [41], segmentation-based methods perform better in detection for text of arbitrary shapes. Similar to ScRN [47], we explore the geometric attributes of text lines, which contains the center line, text height, character orientation and text orientation as shown in Fig. 3. We adopt ResNet50 [8] together with a Feature Pyramid Network (FPN) [17] as the stem network. The generation of ground truth attribute maps of the text and the loss function L_{det} can refer to ScRN. After that, the boundary points of each text line can be calculated as follows.

The process of extract center point list $C = \{C_1, C_2, ..., C_n\}$ is referred to [22]. For any center point $C_p = (x_{C_p}, y_{C_p})$, the upper boundary point T_p and the lower boundary point B_p are formulated as

$$\begin{cases} x_{T_p} = x_{C_p} + 0.5 * r_p * cos\ \psi \\ y_{T_p} = y_{C_p} - 0.5 * r_p * sin\ \psi \end{cases}, \tag{1}$$

$$\begin{cases} x_{B_p} = x_{C_p} - 0.5 * r_p * cos\ \psi \\ y_{B_p} = y_{C_p} + 0.5 * r_p * sin\ \psi \end{cases}, \tag{2}$$

where r_p is the predicted text height and ψ is the predicted character orientation of C_p.

To make boundary points contain complete text instance, that is, the both ends of the text are not cropped during text alignment, we do further post-processing of the boundary points. On the basis of the center line C, two endpoints C_0 and C_{n+1} are added, where $C_0 = (x_{C_0}, y_{C_0})$ is calculated as

$$\begin{cases} x_{C_0} = x_{C_1} - \eta(x_{C_2} - x_{C_1}) \\ y_{C_0} = y_{C_1} - \eta(y_{C_2} - y_{C_1}) \end{cases}, \tag{3}$$

and C_{n+1} is calculated similarly. η is the parameter to control the extension distance, which is set as 0.7 in our experiments. And the corresponding boundary points for C_0 and C_{n+1} are calculated the same as Eq.1 and Eq.2. The positive effect of the additional endpoints will be verified in Section 4.3.2. So far, the boundary points are ready for text alignment.

Note that boundary points are generated based on ground truth attribute maps in training phase, while they are generated based on predicted geometric attribute maps in inference phase.

3.2 PiecewiseAlign

The alignment of text features, which transforms the whole image features into axis-aligned features of a text instance, is a pivotal component of the end-to-end text spotting framework. The previous methods try to perform the text feature alignment with global transformations, such as affine transformation [19], perspective transformation [36] and TPS transformation [27]. However, these global transformations are not suitable for heavily deformed text instances. Inspired by the piecewise linear mapping functions [2], we propose PiecewiseAlign for text feature alignment, which performs different linear transformations in each triangular region of the text feature map. According to the boundary points of text instances, we divide the text feature into a set of contiguous and non-overlapping triangular patches through triangulation. Then PiecewiseAlign is adopted to calculate a group of linear transformation parameters, based on which the axis-aligned features are extracted by piecewise grid sampling. It's worth noting that the linear transformations of all the patches are computed in parallel.

Triangulation can be seen as a preprocessing step for PiecewiseAlign. Boundary points on the input feature maps are first reordered, and then three sequential points are connected to form a triangle patch, as shown in Fig. 4(a). The corresponding fiducial points on the axis-aligned template are computed by the relative position of the boundary points, as shown in Fig. 4(b),

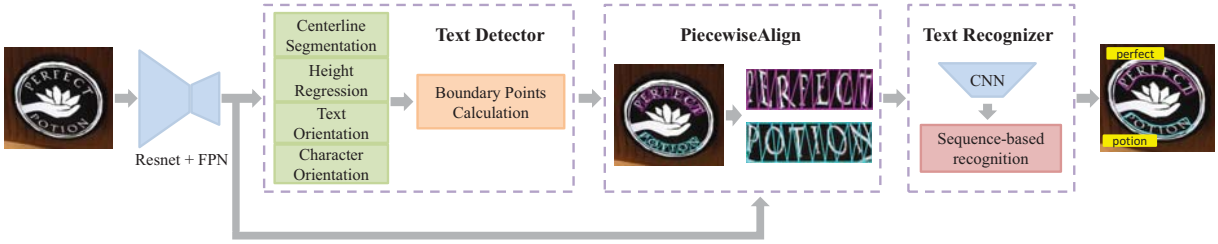


Fig. 2 The overview framework of the proposed TextTriangle, which consists of a backbone network, a text detector, an alignment layer and a text recognizer. The proposed alignment layer PiecewiseAlign calculates axis-aligned text features for connecting the detection and recognition brance, which is the first time to describe text with a series of ordered and adjacent triangles.

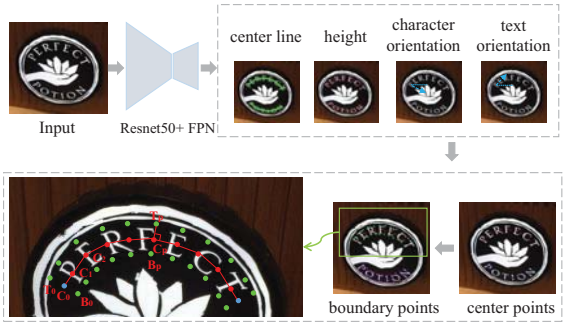


Fig. 3 The framework of the text detector. Red points are sampled from the center line. Two blue points on both ends of the text are the generated endpoints. Green points are the calculated boundary points.

where the size of the axis-aligned template is preset to $H \times W$. We set $D_{a,b}$ to be the distance between point a and b on the input feature maps in Fig.4(a). The location of a fiducial point p_i on the template in Fig.4(b) is calculated as follows,

$$p_i = \begin{cases} p_0 + W \times \dfrac{\sum_{k=0}^{\frac{i}{2}-1} D_{2k,2(k+1)}}{\sum_{k=0}^{\frac{N}{2}-2} D_{2k,2(k+1)}} & , i\%2 = 0 \cap i > 0 \\ p_1 + W \times \dfrac{\sum_{k=0}^{\frac{i-3}{2}} D_{2k+1,2k+3}}{\sum_{k=0}^{\frac{N}{2}-2} D_{2k+1,2k+3}} & , i\%2 = 1 \cap i > 1 \end{cases}, \tag{4}$$

where W is the width of the axis-aligned template. N is the number of boundary points and $N \geq 4$. $i\%2 = 0$ represents points on the top boundary while $i\%2 = 1$ indicates points on the bottom boundary. $D_{2k,2(k+1)}$ represents the distance between two adjacent points on the top boundary and $D_{2k+1,2k+3}$ represents that on the bottom boundary. The coordinates of four corners are fixed that need not to be calculated, where $p_0 = (0, 0)$ and $p_1 = (0, H-1)$. In addition, both source and target boundary points need to be normalized to [-1,1] to calculate the transformation parameters in experiments.

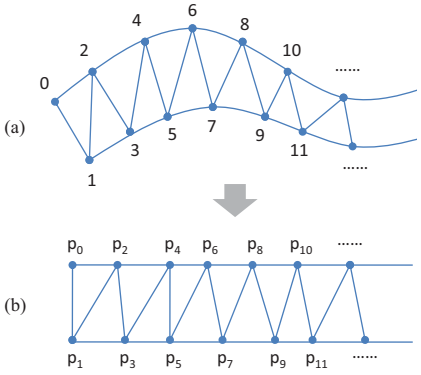


Fig. 4 The illustration of triangulation, where the numbers are reordered. (a) the triangulation based on boundary points. (b) the aligned template.

PiecewiseAlign generates axis-aligned features of a text instance by performing piecewise linear transformations and piecewise grid sampling based on the triangulation as shown in Fig. 5. To reduce the cost of computing, we first crop out the arbitrary-shaped text features with the minimum bounding box and take them as the input of PiecewiseAlign. The input feature map is warped to a group of aligned feature maps based on the triangular patches, and the corresponding masks are indispensable to retain the region of the aligned triangular patch. Then, the group of masked aligned features are added up to generate the final axis-aligned feature map.

Denoting the sequenced triangular patches of the input feature map as $R = \{R_1, R_2, ..., R_{N-2}\}$ and the corresponding aligned patches as $R' = \{R'_1, R'_2, ..., R'_{N-2}\}$. The input patch R_i =

$\{(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), (x_{i3}, y_{i3})\}$, where $\{x_{ij}, y_{ij}\}$ is the vertex coordinate of the triangular patch i. Given the input patch R_i and the aligned patch R'_i, we can get the linear transformation $T_i^{2\times3}$ as

$$T_i = \begin{bmatrix} x_{i1} & x_{i2} & x_{i3} \\ y_{i1} & y_{i2} & y_{i3} \end{bmatrix} \times \begin{bmatrix} x'_{i1} & x'_{i2} & x'_{i3} \\ y'_{i1} & y'_{i2} & y'_{i3} \\ 1 & 1 & 1 \end{bmatrix}^{-1} \quad (5)$$

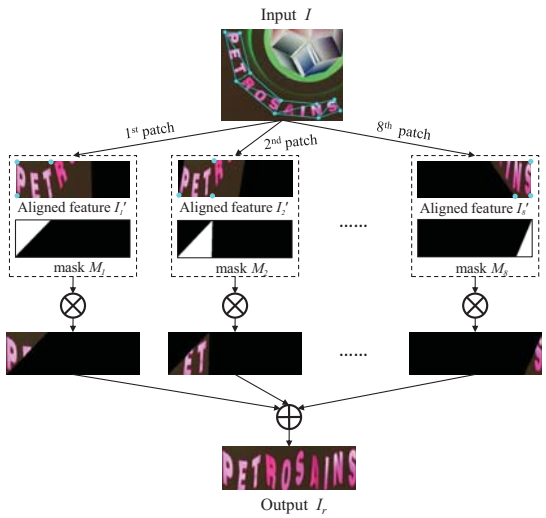


Fig. 5 The illustration of PiecewiseAlign. The blue points on the aligned feature are the fiducial points corresponding to each triangular patch.

After that, a grid generator is used to generate a sampling grid for the input feature I, and a group of aligned features $I' = \{I'_1, I'_2, ..., I'_{N-2}\}$ are generated by grid sampling. The masks with the same size of the axis-aligned template, are denoted as $M = \{M_1, M_2, ..., M_{N-2}\}$. Since two adjacent triangular patches share one edge, the pixel value on this edge of the final axis-aligned feature will be double computed. To avoid that, M can be written as:

$$M_i(q) = \begin{cases} 1 & , q \in R'_i \ and \ q \notin line(v'_{i2}, v'_{i3}) \\ 0 & , else \end{cases}, \quad (6)$$

where q is any point in the mask M_i. $line(v'_{i2}, v'_{i3})$ represents the edge between the last two vertexes of M_i. The final aligned feature map I_r is generated by adding up the group of masked I'_i, which can be written as:

$$I_r = \sum_{i=0}^{N-2} I'_i \times M_i \quad (7)$$

Moreover, the alignment for a group of triangular patches are completed in parallel, and all the operators in PiecewiseAlign are differentiable.

3.3 Text Recognition

Benefiting from the shared backbone feature and the proposed PiecewiseAlign, we adopt a light-weight network for recognition. The recognition module is an attention-based encode-decoder model. The encoder is a CNN structure as shown in Table 1., while the decoder is attention-based. With the input size of 8×32, the features are updated and fed into an attention based network.

Table 1 The network architecture of the text recognizer. Each convolutional layer is followed by a batch normalization layer and a ReLU layer.

Layer Name	Configurations	Size
Input	-	$256 \times 8 \times 32$
Convolution	c:256, k:3, s:1 $\times$ 1	$256 \times 8 \times 32$
Convolution	c:256, k:3, s:1 $\times$ 1	$256 \times 8 \times 32$
Convolution	c:256, k:3, s:1 $\times$ 1	$256 \times 8 \times 32$
Convolution	c:256, k:3, s:1 $\times$ 1	$256 \times 8 \times 32$
MaxPooling	k:2, s:2 $\times$ 1	$256 \times 4 \times 32$
Convolution	c:256, k:2, s:2 $\times$ 1	$256 \times 2 \times 32$
Convolution	c:256, k:2, s:1 $\times$ 1	$256 \times 2 \times 32$
AvgPooling	k:2, s:2 $\times$ 1	$256 \times 1 \times 32$
Channels-Permute	-	32×256
GRU	hidden:256	32×256
GRU	hidden:256	32×256

c,k,s represent channel,kernel and stride sizes respectively.

A gated recurrent unit(GRU) [5] is used to convert the features into character sequence $(o_1, o_2, ..., o_L)$, where L is the length of the predicted text. At time step t, the final predicted distribution over chararcters is computed by

$$o_t = Softmax(W_o s_t + b_o) \quad (8)$$

$$s_t = GRU(Concat(Embedding(o_{t-1}), g_t), s_{t-1}) \quad (9)$$

where s_t is the hidden state of the GRU at time step t. the glimpse vector g_t is calculated by

$$g_t = \sum_i \alpha_{t,i} h_{i,c} \tag{10}$$

where α_t denotes the attention weight at t. α_t is computed as

$$\alpha_{t,i} = \frac{exp(e_{t,i})}{\sum_i (exp(e_{t,i}))} \tag{11}$$

$$e_{t,i} = {V_a}^T tanh(H) \tag{12}$$

$$H = W_h h_i + W_s s_{t-1} + b \tag{13}$$

where W_h, W_s and b are trainable parameters. We train the model using minimum the negative log-likelihood of conditional probability as follows,

$$L_{reg} = -\sum_{n=1}^{N} \sum_{t=1}^{M} \log\, p(y_{n,t} \| y_{n,1:t-1}, I_n, \delta), \tag{14}$$

where I_n and $Y_n = \{y_{n,t}\}$ are the input image and corresponding text in the train set. δ represents the parameters of the model. M is the maximal length of the predicted text.

Since all the operators in PiecewiseAlign are differentiable, the text detector and recognizer can be unified for end-to-end training. The loss function can be written as:

$$L = L_{det} + \lambda L_{rec}, \tag{15}$$

where λ is the hyper-parameter to control the trade-off between the two losses.

4 Experiments

We evaluate both text spotting and detection performance of the proposed method on ICDAR2013, ICDAR2015, Total-Text and CTW1500. We also conduct ablation studies to verify the effectiveness of our proposed method.

4.1 Datasets

SynthText[7] is a synthetic dataset containing 800k synthetic text images by rendering synthetic text with natural images, and it is used as the pre-training dataset. It provides annotations for word/character bounding boxes and text sequences.

Total-Text[4] is one of the most important arbitrarily-shaped scene text benchmarks, which contains 1,255 training images and 300 testing images. Each text is annotated as a polygon in word-level. We use the updated Python scripts[1] provided by the official website to evaluate the detection task, and evaluate the end-to-end recognition results over two lexicons: "None" and "Full". "None" means that no lexicons are provided, and "Full" lexicon includes all words in the test set.

CTW1500 dataset contains 1,000 training images and 500 test images. Each image has at least one curved text. It also contains horizontal and multi-oriented texts. Each text is labeled as a polygon with 14 vertexes in line-level. The evaluation protocol of end-to-end recognition follows that for Total-Text.

ICDAR2015 contains 1000 training images and 500 test images, including many perspective scene text. Each text is labeled as a quadrangle with 4 vertexes in word-level.

ICDAR2013 consists of 229 training images and 233 testing images, which are mainly horizonal scene text.

4.2 Implementation Details

We train our model using 8 GTX GeForce 1080T GPUs with the image batch size of 24. We pre-train the network on SynthText and ICDAR-MLT data[25], and than fine-tune it on the training set of the target datasets. The loss weight λ is set to 1 in both pre-trained and fine-tuning stages. We adopt data augmentation strategies, e.g., randomly rotating the images with degree in range of [-15°, 15°], random brightness and contrast on input images, and random crop, where we make sure that no text has been cut. In the pre-training stage, the randomly cropped images are resized to 512×512. Adam optimizer is applied to train our model with a learning rate 0.001. In fine-tuning, we random resize the longer side of input images with length in range of [720, 1600]. The initialized learning rate is 0.001 and multiplied by

[1] https://github.com/cs-chan/Total-Text-Dataset/tree/master/Evaluation Protocol/Python scripts/Deteval.py

0.8 after each 100 epochs. Online hard example mining(OHEM) [35] strategy is also applied for balancing the foreground and background samples. In addition, because the dataset CTW1500 contains a small amount of Chinese text, we directly regard all the Chinese text as "unseen" class during training.

4.3 Ablation Studies

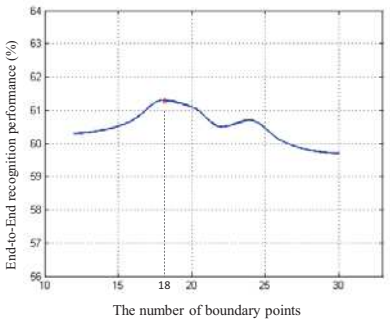
To evaluate the effectiveness of the proposed PiecewiseAlign, we conduct ablation studies from two aspects. First, we analyze the effect of the number of boundary points on text alignment and recognition performance. Second, we compare PiecewiseAlign and TPS-based alignment on recognition performance.

4.3.1 Sampling points analysis

We first conduct sensitivity analysis of how the number of the boundary points may affect the end-to-end results. The ideal text alignment result is shown in Fig.6. We verify the end-to-end recognition performance on the Total-Text dataset by comparing different number of boundary points as shown in Fig. 6(a). From the results, we find that more boundary points does not mean better recognition performance. For the same text instance with arbitrary shapes, more boundary points should make the aligned text warped smoother as shown in Fig. 6(b). But in fact, the more boundary points, the closer they are, which requires more accurate calculations for the location of points. In this case, a small deviation in the calculation of the boundary point position will lead to dislocation between the near boundary points, which will cause the misalignment of text features and the failure of text recognition. In the experiment, the end-to-end result of 18 points is better than others, so we use 18 boundary points as the final setting. Note that if the width of text is too small, reduce the number of points appropriately to ensure that the boundary points do not overlap.

4.3.2 PiecewiseAlign vs. TPS-based Alignment

The TPS-based text alignment is the most popular feature alignment operator in arbitrary-shaped text recognition and spotting. The TPS-based text alignment aims to transform features with



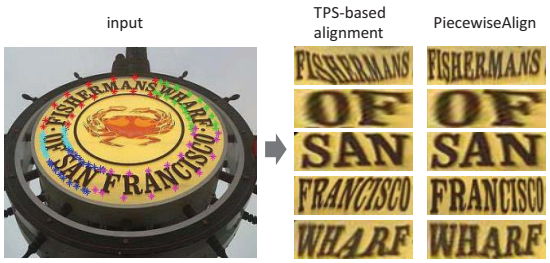
(a) The end-to-end recognition performance based on different numbers of boundary points.

(b) The ideal results of PiecewiseAlign based on different numbers of boundary points.

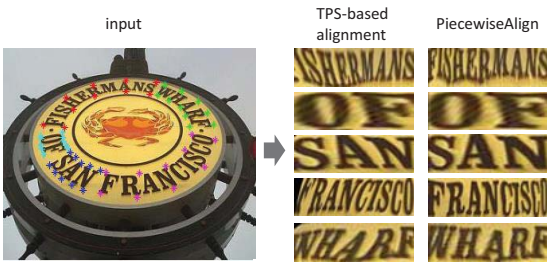
Fig. 6 Ablation studies for the number of boundary points used in PiecewiseAlign.

TPS transformation in a similar way, in which the transformation parameters are calculated from the detected boundary points. In the TPS-based method, entire text feature share this set of transformation parameters. However, the PiecewiseAlign operator divides the text feature into several regions, each of which applies different transformation parameters. Through analyzing the recognition results, we argue that TPS-based method may be unsuitable for text with large deformations and multi-scale. Because each part of the text may differ in the type and degree of deformation, globally consistent transformation cannot satisfy the internal variability of text.

Fig.7 shows a clear comparison between the two operators. We compared the alignment results with different numbers of boundary points and with or without additional endpoints. The input in Fig.7(a) is the detector output with 18 boundary points, which contain the added endpoints as described in Section 3.1. We can see that the text is clearly better aligned by PiecewiseAlign than TPS-based alignment. For text "OF" and "SAN" with small deformation, the results of two operators are similar. But for text with large deformation such as "FISHERMANS", some characters are distorted even more by TPS-based



(a) 18 boundary points after adding endpoints.



(b) 10 boundary points with no endpoints added.

Fig. 7 Visualized results of TPS-based alignment and PiecewiseAlign based on different boundary points. The addition of endpoints has been covered in Section 3.1.

alignment. Compared with TPS-based operator, PiecewiseAlign preserves the original shape of characters to the maximum extent, making the text more readable. The input in Fig.7(b) is the detector output with 10 boundary points and no endpoints added. First, in contrast to Fig.7(a), the characters at both ends of the text are at higher risk of being cropped in Fig.7(b), from which we can see the effect of the additional endpoints. Second, as the number of boundary points reduces, the alignment effect of TPS-based operator declines obviously, such as the character "F" disappearing from the text "FISHERMANS". By comparison, PiecewiseAlign is more robust regardless of the number of boundary points.

To further compare the effect of TPS-based alignment and PiecewiseAlign on text of arbitrary shapes, we evaluate a variant of our method with TPS-based alignment which is named "With TPS" in Tables 3 and 4. In the variant, we replace PiecewiseAlign with TPS-based alignment. The results show that the PiecewiseAlign is superior to the TPS-based alignment in the end-to-end recognition performance for arbitrary-shaped text.

4.4 Comparison with the State-of-the-art

We first evaluate our method on IC13 mainly consisting of horizontal texts as shown in Table 2. The results represent that our method achieve competitive performance compared to previous methods. Although the performance of detection and end-to-end recognition task are inferior to Mask TextSpotter, it is worth noting that our method dose not need any character-level annotations, which owns much more practical value.

We also evaluate our method on IC15 containing many perspective text as shown in Table 2. With the help of PiecewiseAlign layer, TextTriangle outperforms other methods in the end-to-end recognition and word spotting tasks, although the detection performance is inferior to TextDragon and Boundary.

We further compare our method with previous methods on curved benchmarks Total-Text and CTW1500. From Table 3 and Table 4, we can see the proposed method can achieve competitive performance. TextTriangle can accurately detect and recognize clear curved text and multi-scale text. Some word spotting results in Fig. 8 show that the proposed method can handle scene texts of arbitrary shapes. In addition, the performance on CTW1500 is slightly inferior to other methods. Through the analysis of the results, we find that the proposed method needs to be further improved in long text detection.

5 Conclusion

We propose TextTriangle—a novel end-to-end method to detect and recognize the scene text of arbitrary shapes. In TextTriangle, a text instance is the first time to be represented as a sequence of ordered triangles attached to each other, where a text detector is used to locate boundary points of the text instance for triangulation. With the ordered sequence of triangular patches, a new PiecewiseAlign layer is proposed to naturally connect the text detector and the recognizer. Experiments on benchmarks have demonstrated the effectiveness of our method. As for the future work, we will enhance its performance in long text detection.

Table 2 Results on IC13 and IC15. "P","R","F" represent "Precision", "Recall", "F-measure" respectively.'S','W' and 'G' represent recognition with strong, weak and generic lexicon respectively.

Dataset	Method	Detection			End-to-End			Word Spotting		
		P(%)	R(%)	F(%)	S(%)	W(%)	G(%)	S(%)	W(%)	G(%)
IC13	FOTS[19]	-	-	88.2	88.8	87.1	80.8	92.7	90.7	83.5
	TextNet[36]	93.3	89.4	91.3	89.8	88.9	83.0	94.6	**94.5**	87.0
	Mask TextSpotter[24]	**95.0**	88.6	**91.7**	**92.2**	**91.1**	**86.5**	92.5	92.0	88.2
	Boundary[39]	93.1	87.3	90.1	88.2	87.7	84.1	-	-	-
	TextPerceptron[27]	94.7	88.9	**91.7**	91.4	90.7	85.8	**94.9**	94.0	88.5
	With TPS	93.0	87.1	90.0	90.7	89.2	84.8	93.8	93.6	87.4
	TextTriangle	94.1	88.0	90.9	91.8	90.9	86.0	**94.9**	94.2	**88.6**
IC15	FOTS[19]	91.0	85.2	88.0	81.1	75.9	60.8	84.7	79.3	63.3
	TextNet[36]	89.4	85.4	87.4	78.7	74.9	60.5	82.4	78.4	62.4
	Mask TextSpotter[24]	91.6	81.0	86.0	79.3	73.0	62.4	79.3	74.5	64.2
	TextDragon[6]	**92.45**	83.75	87.88	**82.54**	**78.34**	65.15	**86.22**	**81.62**	68.03
	Boundary[39]	89.8	**87.5**	**88.6**	79.7	75.2	64.1	-	-	-
	TextPerceptron[27]	92.3	82.5	87.1	80.5	76.6	65.1	84.1	79.4	67.9
	With TPS	91.6	82.5	86.8	80.2	75.8	64.6	84.0	79.1	66.5
	TextTriangle	91.8	83.0	87.1	81.2	76.5	**65.2**	85.2	80.1	**68.1**

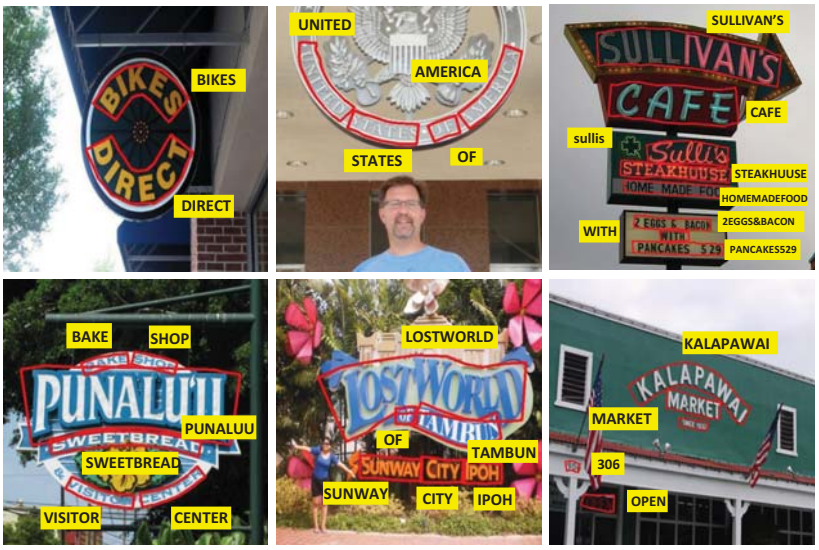


Fig. 8 Examples of word spotting results.

6 Acknowledge

This research is supported by National Natural Science Foundation of China under No.61876154.

References

[1] A.K. B, A. S, A. K, et al (2021) Joint visual semantic reasoning: Multi-stage decoder for text recognition. ICCV

Table 3 Results on Total-Text test set.

Method	Detection			End-to-End	
	P(%)	R(%)	F(%)	None(%)	Full(%)
FOTS[19]	52.3	38.0	44.0	32.2	35.5
Mask TextSpotter[24]	69.0	55.0	61.3	52.9	71.8
TextNet[36]	59.5	68.2	63.5	54.0	-
TextSnake[22]	82.7	74.5	78.4	-	-
TextDragon[6]	85.6	75.7	80.3	48.8	74.8
TextField[45]	81.2	79.9	80.6	-	-
LOMO[48]	75.7	**88.6**	81.6	-	-
TextPerceptron[27]	**88.8**	81.8	**85.2**	69.7	**78.3**
With TPS	87.9	81.0	84.6	69.0	77.6
TextTriangle	88.3	81.7	85.1	**69.9**	77.9

Table 4 Results on CTW1500 test set.

Method	Detection			End-to-End	
	P(%)	R(%)	F(%)	None(%)	Full(%)
FOTS[19]	79.5	52.0	62.8	21.1	39.7
TextSnake[22]	67.9	**85.3**	75.6	-	-
TextDragon[6]	84.5	82.8	83.6	39.7	72.4
ContourNet[42]	83.7	84.1	83.9	-	-
TextPerceptron[27]	**87.5**	81.9	84.6	**57.0**	-
ABCNet[21]	-	-	-	45.2	**74.1**
PCR[26]	87.2	82.3	**84.7**	-	-
With TPS	85.2	82.0	83.5	54.8	72.1
TextTriangle	86.7	82.2	84.3	56.8	73.9

[2] Ardeshir, Goshtasby (1986) Piecewise linear mapping functions for image registration. Pattern Recognition

[3] Chen J, Lian Z (2021) Textpolar: irregular scene text detection using polar representation. International Journal on Document Analysis and Recognition (IJDAR) 24:315–323

[4] Ch'Ng CK, Chan CS, Liu CL (2019) Total-text: toward orientation robustness in scene text detection. Document Analysis and Recognition (7)

[5] Cho K, Merrienboer BV, Gulcehre C, et al (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. Computer Science

[6] Feng W, He W, Yin F, et al (2019) Textdragon: An end-to-end framework for arbitrary shaped text spotting. ICCV

[7] Gupta A, Vedaldi A, Zisserman A (2016) Synthetic data for text localisation in natural images. In: CVPR

[8] He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. CVPR

[9] Hu W, Cai X, Hou J, et al (2020) Gtc: Guided training of ctc towards efficient and accurate scene text recognition. AAAI 34(7):11,005–11,012

[10] Huang J, Pang G, Kovvuri R, et al (2021) A multiplexed network for end-to-end, multilingual ocr. CVPR

[11] Huang M, Liu Y, Peng Z, et al (2022) Swintextspotter: Scene text spotting via better synergy between text detection and text recognition. CVPR

[12] Li H, Wang P, Shen C (2017) Towards end-to-end text spotting with convolutional recurrent neural networks. ICCV

[13] Liang M, Hou JB, Zhu X, et al (2022) Scene text detection via decoupled feature pyramid networks. International Journal on Document Analysis and Recognition (IJDAR)

[14] Liao M, Lyu P, He M, et al (2019) Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. TPAMI

[15] Liao M, Wan Z, Yao C, et al (2019) Real-time scene text detection with differentiable binarization. CVPR

[16] Liao M, Pang G, Huang J, et al (2020) Mask textspotter v3: Segmentation proposal network for robust scene text spotting. ECCV

[17] Lin T, Dollár P, Girshick R, et al (2016) Feature pyramid networks for object detection. CVPR

[18] Litman R, Anschel O, Tsiper S, et al (2020) Scatter: Selective context attentional scene text recognizer. CVPR

[19] Liu X, Liang D, Yan S, et al (2018) Fots : Fast oriented text spotting with a unified network. CVPR

[20] Liu X, Meng G, Pan C (2019) Scene text detection and recognition with advances in deep learning: a survey. International Journal on Document Analysis and Recognition(IJDAR)

[21] Liu Y, Chen H, Shen C, et al (2020) Abcnet: Real-time scene text spotting with adaptive bezier-curve network. In: CVPR

[22] Long S, Ruan J, Zhang W, et al (2018) Textsnake: A flexible representation for detecting text of arbitrary shapes. ECCV

[23] Long S, He X, Yao C (2020) Scene text detection and recognition: The deep learning era. IJCV

[24] Lyu P, Liao M, Yao C, et al (2018) Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. ECCV

[25] Nayef N, Yin F, Bizid I, et al (2017) Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification - rrc-mlt. In: ICDAR

[26] P. D, S. Z, H. Z, et al (2021) Progressive contour regression for arbitrary-shape scene text detection. CVPR

[27] Qiao L, Tang S, Cheng Z, et al (2020) Text perceptron: Towards end-to-end arbitrary-shaped text spotting. AAAI

[28] Qiao Z, Zhou Y, Yang D, et al (2020) Seed: Semantics enhanced encoder-decoder framework for scene text recognition. CVPR

[29] Qin S, Chen L (2022) Arbitrary-shaped scene text detection with keypoint-based shape representation. International Journal on Document Analysis and Recognition (IJDAR) 25(2):115–127

[30] Qin S, Bissacco A, Raptis M, et al (2019) Towards unconstrained end-to-end text spotting. ICCV

[31] Ren S, He K, Girshick R, et al (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. TPAMI 39(6)

[32] Shi B, Xiang B, Cong Y (2016) An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. TPAMI 39(11):2298–2304

[33] Shi B, Bai X, Yao C (2017) An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. TPAMI

[34] Shi B, Yang M, Wang X, et al (2018) Aster: An attentional scene text recognizer with flexible rectification. TPAMI

[35] Shrivastava A, Gupta A, Girshick R (2016) Training region-based object detectors with online hard example mining. CVPR

[36] Sun Y, Zhang C, Huang Z, et al (2018) Textnet: Irregular text reading from images with an end-to-end trainable network. ACCV

[37] Tian Z, Shu M, Lyu P, et al (2020) Learning shape-aware embedding for scene text detection. In: CVPR

[38] Wan Z, He M, Chen H, et al (2020) Textscanner: Reading characters in order for robust scene text recognition. AAAI

[39] Wang H, Lu P, Zhang H, et al (2020) All you need is boundary: Toward arbitrary-shaped text spotting. AAAI

[40] Wang W, Xie E, Li X, et al (2019) Shape robust text detection with progressive scale expansion network. CVPR

[41] Wang X, Jiang Y, Luo Z, et al (2019) Arbitrary shape scene text detection with adaptive text region representation. CVPR

[42] Wang Y, Xie H, Zha Z, et al (2020) Contournet: Taking a further step toward accurate

arbitrary-shaped scene text detection. CVPR

[43] Wu G, Zhang Z, Xiong Y (2022) Carvenet: a channel-wise attention-based network for irregular scene text recognition. International Journal on Document Analysis and Recognition (IJDAR)

[44] Xing L, Tian Z, Huang W, et al (2019) Convolutional character networks. ICCV

[45] Xu Y, Wang Y, Zhou W, et al (2019) Textfield: Learning a deep direction field for irregular scene text detection. TIP

[46] Y. Z, Chen J, Liang L, et al (2021) Fourier contour embedding for arbitrary-shaped text detection. CVPR

[47] Yang M, Guan Y, Liao M, et al (2019) Symmetry-constrained rectification network for scene text recognition. ICCV

[48] Zhang C, Liang B, Huang Z, et al (2019) Look more than once: An accurate detector for text of arbitrary shapes. CVPR

[49] Zhang X, Su Y, Tripathi S, et al (2022) Text spotting transformers. CVPR

Hui Xu received her B.Sc and M.Sc in automation from Beijing University of Posts and Telecommunications in 2009 and 2012 respectively. She is currently a Ph.D student in the Chongqing School at University of Chinese Academy of Sciences and works as an Engineer in the Chongqing Institute of Green and Intelligent Technology at Chinese Academy of Sciences. Her research interests include scene text detection and recognition.

Qiu-Feng Wang is currently an associate professor at School of Advanced Technology in Xi'an Jiaotong-Liverpool University (XJTLU). He received the B.Sc. degree in Computer Sicence from Nanjing University of Science and Technology (NJUST) in July 2006, and the Ph.D degree in Pattern Recognition and Intelligence Systems from Institute of Automation, Chinese Academy of Sciences (CASIA) in July 2012. His research interests include pattern recognition and machine learning, more specifically, the document analysis and recognition.

Zhenghao Li is an Associate Professor in the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences. He received the B.S. degree in telecommunications engineering from Chongqing University, China, in 2003 and the Ph.D. degree in instrumentation science and technology from Chongqing University in 2009. His current research interests include image processing, pattern recognition, and edge computing.

Yu Shi is a senior engineer at CIGIT, CAS. He received the B.S degree in computer science and technology and the M.E degree in software engineering from Wuhan University, Wuhan, China, in 2003 and 2007. He is currently the director of the Research Center for Intelligent Security Technology in CIGIT. He has published more than 20 patents and obtained 4 patent licenses. He is the West Light A Class awarded by Chinese academy of sciences.

Xiangdong Zhou is a professor in the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences. He received the B.S. degree in Applied Mathematics and the M.S. degree in Management Science and Engineering both from National University of Defense Technology, Changsha, China, the Ph.D. degree the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1998, 2003 and 2009, respectively. His research interests include handwriting recognition and ink document analysis.