SPTS: Single-Point Text Spotting

Dezhi Peng
South China University of Technology

Xinyu Wang
Zhejiang University

Yuliang Liu*
Huazhong University of Science and Technology

Jiaxin Zhang
South China University of Technology

Songxuan Lai
Huawei Cloud Computing Technologies

Dahua Lin
Chinese University of Hong Kong

Mingxin Huang
South China University of Technology

Jing Li
Huawei Cloud Computing Technologies

Shenggao Zhu
Huawei Cloud Computing Technologies

Chunhua Shen
Zhejiang University

Xiang Bai
Huazhong University of Science and Technology

Lianwen Jin*
South China University of Technology

ABSTRACT
Existing scene text spotting (i.e., end-to-end text detection and recognition) methods rely on costly bounding box annotations (e.g., text-line, word-level, or character-level bounding boxes). For the first time, we demonstrate that training scene text spotting models can be achieved with an extremely low-cost annotation of a single-point for each instance. We propose an end-to-end scene text spotting method that tackles scene text spotting as a sequence prediction task. Given an image as input, we formulate the desired detection and recognition results as a sequence of discrete tokens and use an auto-regressive Transformer to predict the sequence. The proposed method is simple yet effective, which can achieve state-of-the-art results on widely used benchmarks. Most significantly, we show that the performance is not very sensitive to the positions of the point annotation, meaning that it can be much easier to be annotated or even be automatically generated than the bounding box that requires precise positions. We believe that such a pioneer attempt indicates a significant opportunity for scene text spotting applications of a much larger scale than previously possible. The code is available at https://github.com/shannanyinxiang/SPTS.

CCS CONCEPTS
• Computing methodologies → Scene understanding: Computer vision.

KEYWORDS
Scene text spotting, Transformer, Vision Transformer, Single-point representation

*Corresponding authors. Part of this work was done by Yuliang Liu at the Chinese University of Hong Kong.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:
Dezhi Peng, Xinyu Wang, Yuliang Liu, et al. 2022. SPTS: Single-Point Text Spotting. In Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3503161.3547942

1 INTRODUCTION
In the last decades, it has been witnessed that modern Optical Character Recognition (OCR) algorithms are able to read textual content from pictures of complex scenes, which is an incredible development, leading to enormous interest from both academia and industry. The limitation of existing methods, and particularly their poorer performance on arbitrarily shaped scene text, have been repeatedly identified [6, 19, 23]. This can be seen in the trend of worse predictions for instances with curved shapes, varied fonts, distortions, etc.

The focus of research in the OCR community has moved on from horizontal [20, 35] and multi-oriented text [24, 42, 43] to arbitrarily shaped text [23, 29] in recent years, accompanied by the annotation format from horizontal rectangles, to quadrilaterals, and to polygons. The fact that regular bounding boxes are prone to involve noises has been well studied in previous works (see Fig. 1), which has proved that character-level and polygonal annotations can effectively lift the model performance [19, 21, 29]. Furthermore, many efforts have been made to develop more sophisticated representations to fit arbitrarily shaped text instances [8, 23, 27, 37, 44] (see Fig. 2). For example, Text Dragon [8] utilizes character-level bounding boxes to generate centerlines for enabling the prediction of local geometry attributes, ABCNet [23] converts polygon annotations to Bezier curves for representing curved text instances, and Text Snake [27] describes text instances by a series of ordered disks centered at symmetric axes. However, these novel representations are primarily and carefully designed by experts based on prior knowledge, heavily relying on highly customized network architecture (e.g., specified Region of Interest (RoI) modules) and consuming more expensive annotations (e.g., character-level annotations), limiting their generalization ability for practical applications.

To reduce the cost of data labeling, some researchers [1, 2, 12, 34] have explored training the OCR models with coarse annotations in a
The main contributions of this work are summarized as follows:

1. We propose a new Transformer-based scene text spotter, which forms the text spotting as a language modeling task. Given an input image, our method predicts a discrete token sequence that includes both detection and recognition results. Benefiting from such a concise pipeline, the complex post-processing and sampling strategies designed based on prior knowledge can be discarded, showing great potential in terms of flexibility and generality.
2. To evaluate the effectiveness of proposed methods, extensive experiments and ablations are conducted on four widely used OCR datasets, i.e., ICDAR 2013 [16], ICDAR 2015 [14], Total-Text [5], and SCUT-CTW1500 [25], involving both horizontal and arbitrarily shaped texts. The results show that the proposed SPTS can achieve state-of-the-art performance compared with existing approaches.

1.1 Related Work

In the past decades, a variety of scene text datasets using different annotation styles have been proposed, focusing on various scenarios, including horizontal text [15, 16] described by rectangles (Fig. 1a), multi-oriented text [14, 30] represented by quadrilaterals (Fig. 1b), and arbitrarily shaped text [5, 6, 25] labeled by polygons (Fig. 1d). These forms of annotations have facilitated the development of corresponding OCR algorithms. For example, earlier works [17] usually adapt the generic object detectors to scene text spotting, where feature maps are shared between detection and recognition heads via RoI modules. These approaches follow the sampling mechanism designed for generic object detection and utilize rectangles to express text instances, thus performing worse on non-horizontal targets. Later, some methods [3, 10, 22] replace the rectangular bounding boxes with quadrilaterals by modifying the regular Region Proposal Network (RPN) to generate oriented proposals, enabling better performance for multi-oriented texts. Recently, with the presentation of the curved scene text datasets [5, 6, 25], the research interest of the OCR community has shifted to more challenging arbitrarily shaped texts. Generally, there are two widely adopted solutions to solve the arbitrarily shaped text spotting task, i.e., segmentation-based [19, 32, 33, 40]
2.1 Sequence Construction

The fact that a sequence can carry information with multiple attributes naturally enables the text spotting task, where text instances are simultaneously localized and recognized. To express the target text instances by a sequence, it is required to convert the continuous descriptions (e.g., bounding boxes) to a discretized space. To this end, as shown in Fig. 4, we follow Pix2Seq [4] to build the target sequence; what distinguishes our methods is that we further simplify the bounding box to a single-point and use the variable-length transcription instead of the single-token object category.
Specifically, the continuous coordinates of the central point of the text instance are uniformly discretized into integers between \([1, \text{bins}]\), where \(\text{bins}\) controls the degree of discretization. For example, an image with a long side of 800 pixels requires only \(\text{bins} = 800\) to achieve zero quantization error. Note that the central point of the text instance is obtained by averaging the upper and lower midpoints as shown in Fig. 7a. As so far, a text instance can thereby be represented by a sequence of three parts, i.e., \((x, y, t)\), where \((x, y)\) are the discretized coordinates and \(t\) is the transcription text. Notably, the transcriptions are inherently discrete, i.e., each of the characters represents a category, thus can be easily appended to the sequence. However, different from the generic object detection that has a relatively fixed vocabulary (each \(t\) represents an object category, such as pedestrian), \(t\) can be any natural language text of any length in our task, resulting in a variable length of the target sequence, which may further cause misalignment issues and can consume more computational resources. To eliminate such problems, we pad or truncate the texts to a fixed length \(l_t\), where the \(<\text{PAD}>\) token is used to fill the vacancy for shorter text instances. In addition, like other language modeling methods, \(<\text{SOS}>\) and \(<\text{EOS}>\) tokens are inserted to the head and tail of the sequence, indicating the start and the end of a sequence, respectively. Therefore, given an image that contains \(n_t\) text instances, the constructed sequence will include \((2 + l_t) \times n_t\) discrete tokens, where the text instances would be randomly ordered, following previous works [4]. Supposing there are \(n_{cls}\) categories of characters (e.g., 97 for English characters and symbols), the vocabulary size of the dictionary used to tokenize the sequence can be calculated as \(\text{bins} + n_{cls} + 3\), where the extra three classes are for \(<\text{PAD}>\), \(<\text{SOS}>\), and \(<\text{EOS}>\) tokens. Empirically, we set the \(l_t\) and \(\text{bins}\) to 25 and 1,000, respectively, in our experiments. Moreover, the maximum value of \(n_t\) is set to 60, which means the sequence containing more than 60 text instances will be truncated.

### 2.2 Model Training

Based on the constructed sequence, the input and output sequences of the Transformer decoder are shown in Fig. 5. Since the SPTS is trained to predict tokens, it only requires to maximize the likelihood loss at training time, which can be written as:

\[
\text{maximize} \sum_{i=1}^{L} w_i \log P(s_i | I, s_{1:i-1}).
\]

where \(I\) is the input image, \(s_i\) is the output sequence, \(s\) is the input sequence, \(L\) is the length of the sequence, and \(w_i\) is the weight of the likelihood of the \(i\)-th token, which is empirically set to 1.

### 2.3 Inference

At the inference stage, SPTS auto-regressively predicts the tokens until the end of the sequence token \(<\text{EOS}>\) occurs. The predicted sequence will subsequently be divided into multiple segments, each of which contains 2 + \(l_t\) tokens. Then, the tokens can be easily translated into the point coordinates and transcriptions, yielding the text spotting results. In addition, the likelihood of all tokens in the corresponding segment is averaged and assigned as a confidence score to filter the original outputs, which effectively removes redundant and false-positive predictions.

### 3 EXPERIMENTS

We report the experimental results on four widely used benchmarks including horizontal dataset ICDAR 2015 [14], multi-oriented dataset ICDAR 2015 [14], and arbitrarily shaped datasets Total-Text [5] and SCUT-CTW1500 [25].

#### 3.1 Datasets

**Curved Synthetic Dataset 150k.** It is admitted that the performance of text spotters can be improved by pre-training on synthesized samples. Following previous work [23], we use the 150k synthetic images generated by the SynthText [9] toolbox, which contains around one-third of curved texts and two-third of horizontal instances.

**ICDAR 2013** [16] contains 229 training and 233 testing samples, while the images are primarily captured in a controlled environment, where the text contents of interest are explicitly focused in horizontal.

**ICDAR 2015** [14] consists of 1,000 training and 500 testing images that were incidentally captured, containing multi-oriented text instances presented in complicated backgrounds with strong variations in blur, distortions, etc.

**Total-Text** [5] includes 1,255 training and 300 testing images, where at least one curved sample is presented in each image and annotated with polygonal bounding box at the word-level.
SCUT-CTW1500 [25] is another widely used benchmark designed for spotting arbitrary shaped scene text, involving 1,000 and 500 images for training and testing, respectively. The text instances are labeled by polygons at text-line level.

### 3.2 Evaluation Protocol

The existing evaluation protocol of text spotting tasks consists of two steps. Firstly, the intersection over union (IoU) scores between ground-truth (GT) and detected boxes are calculated; and only if the IoU score is larger than a designated threshold (usually set to 0.5), the boxes are matched. Then, the recognized content inside each matched bounding box is compared with the GT transcription; only if the predicted text is the same as the GT will it contribute to the end-to-end accuracy. However, in the proposed method, each text instance is represented by a single-point; thus, the evaluation metric based on the IoU is not available to measure the performance. Meanwhile, comparing the localization performance between bounding-box-based methods and the proposed point-based SPTS might be unfair, e.g., directly treating points inside a bounding box as true positives may overestimate the detection performance. To this end, we propose a new evaluation metric to ensure a relatively fair comparison to existing approaches, which mainly considers the end-to-end accuracy as it reflects both detection and recognition performance (failure detections usually lead to incorrect recognition results). Specifically, as shown in Fig. 6, we modified the text instance matching rule by replacing the IoU metric with a distance metric, i.e., the predicted point that has the nearest distance to the central point of the GT box would be selected, and the recognition results will be measured by the same full-matching rules used in existing benchmarks. Only one predicted point with the highest confidence would be matched to the ground truth; others are then marked as false positives.

To explore whether the proposed evaluation protocol can genuinely represent the model accuracy, Table 1 compares the end-to-end recognition accuracy of ABCNetv1 [23] and ABCNetv2 [26] on Total-Text [5] and SCUT-CTW1500 [25] under two metrics, i.e., the commonly used bounding box metric that is based on IoU, and the proposed point-based metric. The results demonstrate that the point-based evaluation protocol can well reflect the performance, where the difference between the values evaluated by box-based and point-based metrics are no more than 0.5%. For example, the ABCNetv1 model achieves 53.5% and 53.0% scores on the SCUT-CTW1500 dataset under the two metrics, respectively. Therefore, we use the point-based metric to evaluate the proposed SPTS in the following experiments.

### 3.3 Implemented Details

The model is first pretrained on a combination dataset that includes Curved Synthetic Dataset 150k [23], MLT-2017 [31], ICDAR 2013 [16], ICDAR 2015 [14], and Total-Text [5] for 150 epochs, which is optimized by the AdamW [28] with an initial learning rate of $5 \times 10^{-4}$, while the learning rate is linearly decayed to $1 \times 10^{-5}$. After pretraining, the model is then fine-tuned on the training split of each target dataset for another 200 epochs, with a fixed learning rate of $1 \times 10^{-5}$. The entire model is distributively trained on 32 NVIDIA V100 GPUs with a batch size of 32. Note that the effective batch size is 64 because two independent augmentations are performed on each image in a mini-batch, following [4,11]. In addition, we utilize ResNet-50 as the backbone network, while both the Transformer encoder and decoder consist of 6 layers with eight heads. Regarding the architecture of the Transformer, we adopt the Pre-LN Transformer [41]. During training, short size of the input image is randomly resized to a range from 640 to 896 (intervals of 32) while keeping the longer side shorter than 1,600 pixels. Random cropping and rotating are employed for data augmentation. At the inference stage, we resize short edge to 1,000 while keeping longer side shorter than 1,824 pixels, following the previous works [23,26].

### 3.4 Ablation Study

#### 3.4.1 Ablation study of the position of the indicated point

In this paper, we propose to simplify the bounding box to a single-point. Intuitively, all points in the region enclosed by the bounding box should be able to represent the target text instance. To explore the differences, we conduct ablation studies that use three different strategies to get the indicated points (see Fig. 7), i.e., the central point obtained by averaging the upper and lower midpoints, the top-left corner, and the random point inside the box. It should be noted that, we use the corresponding ground-truth here to calculate the distance matrix for evaluating the performance, i.e., the distance to the ground-truth top-left point is used for top-left, the distance to the ground-truth central point for central, and the closest distance to the ground-truth polygon for random.
Table 2: Ablation study of the position of the indicated point.

| Position | E2E Total-Text | E2E CTW1500 |
|----------|----------------|-------------|
|          | None | Full | None | Full |
| Central  | 74.2 | 82.4 | 63.6 | 83.8 |
| Top-left | 71.6 | 79.7 | 61.4 | 82.0 |
| Random   | 73.2 | 80.8 | 62.3 | 81.1 |

Table 3: Comparison with different shapes of bounding boxes. \(N_p\) is the number of parameters required to describe the location of text instances by different representations.

| Variants     | Total-Text | SCUT-CTW1500 | \(N_p\) |
|--------------|------------|--------------|--------|
|              | None | Full | None | Full |
| SPTS-Bezier  | 60.6 | 71.6 | 52.6 | 73.9 | 16 |
| SPTS-Rect    | 71.6 | 80.4 | 62.2 | 82.3 | 4 |
| SPTS-Point   | 74.2 | 82.4 | 63.6 | 83.8 | 2 |

Table 4: End-to-end recognition results on ICDAR 2013. “S”, “W”, and “G” represent recognition with “Strong”, “Weak”, and “Generic” lexicon, respectively.

| Method                      | IC13 End-to-End |
|-----------------------------|-----------------|
|                            | S | W | G |
| Bounding Box-based methods  |   |   |   |
| Jaderberg et al. [13]       | 86.4 |   |   |
| Textboxes [20]              | 91.6 | 89.7 | 83.9 |
| Deep Text Spotter [3]       | 89.0 | 86.0 | 77.0 |
| Li et al. [17]              | 91.1 | 89.8 | 84.6 |
| MaskTextSpotter [29]        | 92.2 | 91.1 | 86.5 |
| Point-based method          |   |   |   |
| SPTS (Ours)                 | 93.3 | 91.7 | 88.3 |

Table 5: End-to-end recognition results on ICDAR 2015. “S”, “W”, and “G” represent recognition with “Strong”, “Weak”, and “Generic” lexicon, respectively.

| Method                      | IC15 End-to-End |
|-----------------------------|-----------------|
|                            | S | W | G |
| Bounding Box-based methods  |   |   |   |
| FOTS [22]                   | 81.1 | 75.9 | 60.8 |
| MaskTextSpotter [18]        | 83.0 | 77.7 | 72.5 |
| CharNet [21]                | 83.1 | 79.2 | 69.7 |
| TextDragon [8]              | 82.5 | 78.3 | 65.2 |
| MaskTextSpotter v3 [19]     | 83.3 | 78.1 | 74.2 |
| MANGO [32]                  | 81.8 | 78.9 | 67.3 |
| ABCNetV2 [26]               | 82.7 | 78.5 | 73.0 |
| PAN++ [40]                  | 82.7 | 78.2 | 69.2 |
| Point-based method          |   |   |   |
| SPTS (Ours)                 | 77.5 | 70.2 | 65.8 |

Table 6: End-to-end recognition results on Total-Text. “None” represents lexicon-free. “Full” represents that we use all the words appeared in the test set.

| Method                      | Total-Text End-to-End |
|-----------------------------|-----------------------|
|                            | None | Full |
| Bounding Box-based methods  |       |       |
| CharNet [21]                | 66.6 | -    |
| ABCNet [23]                 | 64.2 | 75.7 |
| PGNet [39]                  | 63.1 | -    |
| Mask TextSpotter [29]       | 65.3 | 77.4 |
| Qin et al. [33]             | 67.8 | -    |
| Mask TextSpotter v3 [19]    | 71.2 | 78.4 |
| MANGO [32]                  | 72.9 | 83.6 |
| PAN++ [40]                  | 68.6 | 78.6 |
| ABCNet v2 [26]              | 70.4 | 78.1 |
| Point-based method          | SPTS (Ours) | 74.2 | 82.4 |

Table 7: End-to-end recognition results on SCUT-CTW1500. “None” represents lexicon-free. “Full” represents that we use all the words appeared in the test set. ABCNet* means using the github checkpoint.

| Method                      | SCUT-CTW1500 End-to-End |
|-----------------------------|--------------------------|
|                            | None | Full |
| Bounding Box-based methods  |       |       |
| TextDragon [8]              | 39.7 | 72.4 |
| ABCNet [23]                 | 45.2 | 74.1 |
| ABCNet* [23]                | 53.2 | 76.0 |
| MANGO [32]                  | 58.9 | 78.7 |
| ABCNet v2 [26]              | 57.5 | 77.2 |
| Point-based method          | SPTS (Ours) | 63.6 | 83.8 |

The results are shown in Table 2, where the results of central, top-left, and random are close on both datasets. It suggests that the performance is not very sensitive to the positions of the point annotation.

3.4.2 Comparison between different representations. The proposed SPTS can be easily extended to produce bounding boxes by modifying the point coordinates to bounding box locations during sequence construction. Here, we conduct ablations to explore the influence by using different representations of the text instances. Specifically, three variants are explored, including the Bezier curve bounding box (SPTS-Bezier), the rectangular bounding box (SPTS-Rect), and the indicated point (SPTS-point). Since we only focus on end-to-end performance here, to minimize the impact of detection results, each method uses corresponding representations to match the GT box in the evaluation. That is to say, the single-point model (original SPTS) uses the evaluation metrics introduced in Sec. 3.2, i.e., distance between points; the predictions of SPTS-Rect are matched to the circumscribed rectangle of the polygonal annotations; the SPTS-Bezier adopts the original metric that matches polygon boxes. As shown in Table 3, the SPTS-point achieves the best performance on both the Total-Text and SCUT-CTW1500 datasets, outperforming the other two representations by a large margin. Such experimental results suggest that a low-cost annotation, i.e., the indicated point, is capable of providing supervision for the text spotting task. The reason for the low performance of SPTS-Bezier and SPTS-Rect may be that longer sequences (e.g., SPTS-Bezier with \(N_p = 16\) vs. SPTS-Point with \(N_p = 2\)) make them difficult to converge; thus, SPTS-Bezier and SPTS-Rect cannot achieve comparable accuracy under the same training schedule.

3.5 Comparison with Existing Methods on Scene Text Benchmarks

3.5.1 Horizontal-Text Dataset. Table 4 compares the proposed SPTS with existing methods on the widely used ICDAR 2013 [16] benchmark. Our method achieves state-of-the-art results with all three lexicons. It should be noted that the proposed SPTS only utilizes a single-point for training, while the other approaches are fully trained with more costly bounding boxes.

https://github.com/aim-uofa/AdelaiDet/blob/master/configs/BAText/README.md
3.5.2 Multi-Oriented Dataset. The quantitative results of the ICDAR 2015 [14] dataset are shown in Table 5. A performance gap between the proposed SPTS and state-of-the-art methods can still be found, which shows some limitations of our method for tiny texts that are often presented in the ICDAR 2015 dataset. Because the sequence is directly decoded from the feature of the entire image without dedicated RoI operations, the tiny texts are difficult to handle with our method.

3.5.3 Arbitrarily Shaped Dataset. We further compare our method with existing approaches on the benchmarks containing arbitrarily shaped texts, including Total-Text [5] and SCUT-CTW1500 [25]. As shown in Table 6, SPTS achieves state-of-the-art performance only using extremely low-cost point annotations. Additionally, Table 7 shows that our method outperforms state-of-the-art approaches by a large margin on the challenging SCUT-CTW1500 dataset, which further identifies the potentiality of our method.

3.5.4 Summary. In summary, the proposed SPTS can achieve state-of-the-art performance compared with previous text spotters on several widely used benchmarks. Especially on the two curved datasets, i.e., Total-Text [5] and SCUT-CTW1500 [25], the proposed SPTS outperforms some recently proposed methods by a large margin. The reason why our methods can achieve better accuracy on arbitrary-shaped texts might be: (1) The proposed SPTS discards the task-specific modules (e.g., RoI modules) designed based on prior knowledge; therefore, the recognition accuracy is decoupled with the detection results, i.e., SPTS can achieve acceptable recognition results even the detection position is shifted. However, the recognition heads of other methods heavily rely on the detection results, which is the main reason of their poor end-to-end accuracy. Once the text instance cannot be perfectly localized, their recognition heads fail to work. (2) Although previous models are trained in an end-to-end manner, the interactions between their detection and recognition branches are limited. Specifically, the features fed to the recognition module are sampled based on the ground-truth position while training but from detection results at the inference stage, leading to feature misalignment, which is far more severe on curved text instances. However, by tackling the spotting task in a sequence modeling manner, the proposed SPTS eliminates such issues, thus showing more robustness on arbitrarily shaped datasets. The visualization results of SPTS on four testing datasets are shown in Fig. 8.

Figure 8: Qualitative results on the scene text benchmarks. Images are selected from Total-Text (first row), SCUT-CTW1500 (second row), ICDAR 2013 (third row), and ICDAR 2015 (fourth row). Zoom in for best view.
Table 8: Comparison between the end-to-end recognition results of the SPTS and NPTS models.

| Method | Total-Text | SCUT-CTW1500 | ICDAR 2013 | ICDAR 2015 |
|--------|------------|--------------|------------|------------|
|        | None   | Full | None | Full | S | W | G | S | W | G |
| SPTS   | 74.2  | 82.4 | 63.6  | 83.8 | 93.3 | 91.7 | 88.5 | 77.5 | 70.2 | 65.8 |
| NPTS   | 64.7  | 71.9 | 55.4  | 74.3 | 90.4 | 84.9 | 80.2 | 69.4 | 60.3 | 55.6 |

3.6 Extensions of SPTS

3.6.1 No-Point Text Spotting. The experiments suggest that the detection and recognition may have been decoupled. Based on the results, we further show that SPTS can be converged even without the supervision of the single point annotations. The No-Point Text Spotting (NPTS) model is obtained by removing the coordinates of the indicated points from the constructed sequence. Fig. 9 shows the qualitative results of NPTS, which indicates the model may have learned the ability to implicitly find out the locations of the text merely based on the transcriptions. The comparison between the end-to-end recognition results of the SPTS and NPTS models is presented in Tab. 8. The evaluation metric described in Sec. 3.2 is adapted for NPTS, where the distance matrix between the predicted and GT points is replaced with an edit distance matrix between the predicted and GT transcriptions. Despite the obvious gap between SPTS and NPTS, the preliminary results achieved by NPTS are still surprising and very encouraging, which is worth studying in the future.

3.6.2 Single-Point Object Detection. To demonstrate the generality of SPTS, we conduct experiments on the Pascal VOC [7] object detection task, where the model is trained with central points and the corresponding categories. All other settings are identical to the text spotting experiment. Some preliminary qualitative results on the validation set are shown in Fig. 10. The results suggest that single-point might be viable to provide extremely low-cost annotation for general object detection.

4 LIMITATION

One limitation of the proposed framework is that the training procedure requires a large number of computing resources. For example, 150 epochs for 160k scene text pretraining and 200 epochs for fine-tuning require approximately 63 hours when the model is distributively trained using 32 NVIDIA V100 GPU cards. Moreover, owing to the auto-regressive decoding manner, the inference speed is only 1.0 fps on ICDAR 2013 using one NVIDIA V100 GPU.

Another limitation of SPTS is that it can not well handle extremely tiny text spotting (as indicated in Sec. 3.5.2), because the feature representation ability of SPTS can not reach a high resolution for extracting effective tiny text features. This issue deserves further study in the future.

5 CONCLUSION

We propose SPTS, which is, to the best of our knowledge, a pioneering method that tackles scene text spotting using only the extremely low-cost single-point annotation. The successful attempt sheds some brand-new insights that challenges the necessity of the traditional box annotations in the field. SPTS is an auto-regressive Transformer-based framework that can simply generate the results as sequential tokens, which can avoid complex post-processing or exclusive sampling stages. Based on such a concise framework, extensive experiments demonstrate state-of-the-art performance of SPTS on various datasets. We further show that our method has the potential to be extended to the no-point text spotting and generic object detection tasks.

ACKNOWLEDGEMENT

This research is supported in part by NSFC (Grant No.: 61936003), GD-NSF (no.2017A030312006, No.2021A1515011870), and the Science and Technology Foundation of Guangzhou Huangpu Development District (Grant 2020GH17).
REFERENCES

[1] Youngmin Bark, Bado Lee, Dongyoon Han, Sangdoon Yun, and Hwalsuk Lee. 2019. Character region awareness for text detection. In Proc. IEEE Conf. Comp. Vis. Patt. Recogn. 9365–9374.

[2] Christian Bartz, Haojin Yang, and Christoph Meinell. 2018. SRE: Towards semi-supervised end-to-end scene text recognition. In Proc. AAAI Conf. Artificial Intell. 6674–6681.

[3] Michal Busta, Lukas Neumann, and Jiri Matas. 2017. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In Proc. IEEE Int. Conf. Comp. Vis. 2204–2212.

[4] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. 2022. PicSeg: A language modeling framework for object detection. In Proc. Int. Conf. Learn. Represent.

[5] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. 2018. FOTS: Single-Point Text Spotting MM ’22, October 10–14, 2022, Lisboa, Portugal.

[6] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. 2017. Mask TextSpotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In Proc. EURASIP Conf. Patt. Recogn. 20:36–40.

[7] Yiya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. Proc. Int. Conf. Learn. Representations (2018).

[8] Yulong Liu, Chengju Liao, Cong Yao, Wenhao Wu, and Xiang Bai. 2018. MaskTextSpotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In Proc. Eur. Conf. Comp. Vis. 67–83.

[9] Yuliang Liu, Jinyun Lu, Shuaihua Shen, Honglei Wang, and Jiajun Liang. 2017. EAST: An efficient and accurate scene text detector. In Proc. Int. Conf. Mach. Learn. 2467–2476.

[10] Yuliang Liu, Jinyun Lu, Shuaihua Shen, Honglei Wang, and Jiajun Liang. 2017. EAST: An efficient and accurate scene text detector. In Proc. Int. Conf. Mach. Learn. 2467–2476.
Appendix

A TINY TEXT SPOTTING

As discussed in Sec. 3.5.2 and Sec. 4, the proposed SPTS can not accurately recognize tiny texts because it directly predicts the sequence based on the low-resolution high-level features without RoI operations. Especially on the ICDAR 2015 dataset, there still is a performance gap (65.8 vs. 74.2) between our method and the state-of-the-art approaches. Quantitatively, if the texts with area smaller than 3000 (after resizing) are ignored during evaluation, the F-measure with generic lexicons on ICDAR 2015 will be significantly improved from 65.8 to 73.5. Furthermore, current state-of-the-art methods on ICDAR 2015 usually adopt larger image size during testing. For example, the short sides of the testing images are resized to 1440 pixels while the long sides are shorter than 4000 pixels in Mask TextSpotterV3 [19]. As shown in Tab. 9, the performance of SPTS on ICDAR 2015 with larger testing size is much better than that with smaller testing size, indicating the tiny text spotting is an important issue worth studying in the future.

B ORDER OF TEXT INSTANCES

As described in Sec. 2.1, the text instances are randomly ordered in the constructed sequence following Pix2Seq [4]. In this section, we further investigate the impact of the order of text instances. The performances on Total-Text and SCUT-CTW1500 of different ordering strategies are presented in Tab. 10. The “Area” and “Dist2ori” mean that text instances are sorted by the area and the distance to the topleft origin in descending order, respectively. The “Topdown” indicates that text instances are arranged from top to bottom. It can be seen that the random order adopted in SPTS achieves the best performance, which may be explained by the improved robustness due to the different sequences constructed for the same image at different iterations.

Table 9: End-to-end recognition results on ICDAR 2015. “S”, “W”, and “G” represent recognition with “Strong”, “Weak”, and “Generic” lexicon, respectively. Bold indicates the state of the art and underline indicates the second best.

| Method                      | ICD15 End-to-End S | ICD15 End-to-End W | ICD15 End-to-End G |
|-----------------------------|--------------------|--------------------|--------------------|
| Bounding Box-based methods  |                    |                    |                    |
| FOTS [22]                   | 81.1               | 75.9               | 60.8               |
| Mask TextSpotter [18]       | 83.0               | 77.7               | 73.5               |
| CharNet [21]                | 83.1               | 79.2               | 69.1               |
| TextDragon [8]              | 82.5               | 78.3               | 65.2               |
| Mask TextSpotter v3 [19]    | 83.3               | 78.1               | 74.2               |
| MANGO [32]                  | 81.8               | 78.9               | 67.3               |
| ABCNetV2 [26]               | 82.7               | 78.5               | 73.0               |
| PAN++ [40]                  | 82.7               | 78.2               | 69.2               |
| Point-based method          |                    |                    |                    |
| SPTS (1000)                 | 77.5               | 70.2               | 65.8               |
| SPTS (1440)                 | 79.5               | 74.1               | 70.2               |

Table 10: Ablation study of different ordering strategies of text instances in the sequence construction.

| Order       | Total-Text | SCUT-CTW1500 |
|-------------|------------|--------------|
|             | None       | Full         |
|             | None       | Full         |
| Area        | 70.7       | 79.2         |
| Topdown     | 73.2       | 81.3         |
| Dist2ori    | 72.1       | 81.8         |
| Random      | 74.2       | 82.4         |

Table 11: Further comparison of different representations of text instances.

| Variants     | Epochs | Total-Text | SCUT-CTW1500 | N_p |
|--------------|--------|------------|--------------|-----|
| SPTS-Beizer  | 1x     | 60.6       | 71.6         | 52.6| 73.9 | 16  |
| SPTS-Beizer  | 2x     | 62.9       | 74.4         | 51.1| 74.3 | 16  |
| SPTS-Point   | 1x     | 74.2       | 82.4         | 63.6| 83.8 | 2   |

C FURTHER DISCUSSION ON THE REPRESENTATION OF TEXT INSTANCES

In Sec. 3.4.2, the ablation experiments demonstrate that the SPTS-Point with fewer parameters for describing the location of text instances outperforms SPST-Rect and SPTS-Bezier. The longer sequence required by SPTS-Rect and SPTS-Bezier may make them difficult to converge. In Tab. 3, the results of SPTS-Rect and SPTS-Bezier are obtained using the same training schedule as SPTS-Point. To further explore their potentiality, we compare the SPTS-Bezier trained for 2x epochs with SPTS-Point in Tab. 11. Normally, the SPTS-Bezier should perform better than SPTS-Point owing to the more detailed annotation. However, it can be seen that the SPTS-Bezier with 2x epochs does not significantly outperform the counterpart with 1x epochs and is still inferior to the SPTS-Point with 1x epochs. The reason may be the dramatically increased difficulty of the convergence of the Transformer with longer decoded sequences. More training data and epochs may be necessary to address this problem.