Arbitrary Reading Order Scene Text Spotter with Local Semantics Guidance

Jiahao Lyu[*1,4], **Wei Wang**[*3], **Dongbao Yang**[1], **Jinwen Zhong**[1†], **Yu Zhou**[2†]

[1]Institute of Information Engineering, Chinese Academy of Science
[2]VCIP & TMCC & DISSec, College of Computer Science, Nankai University
[3] Shanghai Artificial Intelligence Laboratory
[4]School of Cyber Security, University of Chinese Academy of Sciences
{lvjiahao, yangdongbao, zhongjinwen}@iie.ac.cn, wangwei@pjlab.org.cn, yzhou@nankai.edu.cn

Abstract

Scene text spotting has attracted the enthusiasm of relative researchers in recent years. Most existing scene text spotters follow the detection-then-recognition paradigm, where the vanilla detection module hardly determines the reading order and leads to failure recognition. After rethinking the auto-regressive scene text recognition method, we find that a well-trained recognizer can implicitly perceive the local semantics of all characters in a complete word or a sentence without a character-level detection module. Local semantic knowledge not only includes text content but also spatial information in the right reading order. Motivated by the above analysis, we propose the Local Semantics Guided scene text Spotter (LSGSpotter), which auto-regressively decodes the position and content of characters guided by the local semantics. Specifically, two effective modules are proposed in LSGSpotter. On the one hand, we design a Start Point Localization Module (SPLM) for locating text start points to determine the right reading order. On the other hand, a Multi-scale Adaptive Attention Module (MAAM) is proposed to adaptively aggregate text features in a local area. In conclusion, LSGSpotter achieves the arbitrary reading order spotting task without the limitation of sophisticated detection, while alleviating the cost of computational resources with the grid sampling strategy. Extensive experiment results show LSGSpotter achieves state-of-the-art performance on the InverseText benchmark. Moreover, our spotter demonstrates superior performance on English benchmarks for arbitrary-shaped text, achieving improvements of 0.7% and 2.5% on Total-Text and SCUT-CTW1500, respectively. These results validate our text spotter is effective for scene texts in arbitrary reading order and shape.

1 Introduction

Aiming to integrate the detection (Liao et al. 2020b; Shu et al. 2023; Qin et al. 2023) and recognition (Qiao et al. 2020b; Du et al. 2022) tasks, scene text spotting has received increasing attention recently because of its numerous applications, such as structure information exaction (Xu et al. 2020; Li et al. 2021; Shen et al. 2023), automatic driving (Guo et al. 2021; Min et al. 2022), scene understanding (Zhu et al. 2023; Zeng et al. 2023), scene text editing (Zeng et al. 2024; Li et al. 2024) etc. With the development of well-organized datasets (Liu et al. 2017; Karatzas et al. 2015; Ch'ng and Chan 2017; Ye et al. 2023a; Zhang et al. 2019) and fundamental vision models (Dosovitskiy et al. 2020; Liu et al. 2021b), scene text spotters have achieved prominent results on several public benchmarks.

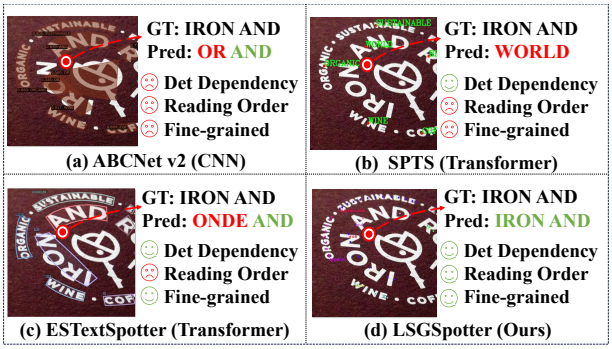

Figure 1: The comparison of arbitrary reading order text instances and analysis from Total-Text (Ch'ng and Chan 2017) of ABCNet v2 (Liu et al. 2021a), SPTS (Peng et al. 2022) and ESTextSpotter (Huang et al. 2023). Our spotter can use the locally fine-grained semantics to perceive reading order without accurate detection dependency.

Existing scene text spotting methods can mainly be divided into two categories according to the utilization of the fundamental vision model: CNN-based and Transformer-based methods. Motivated by general object detection methods (He et al. 2017; Liu et al. 2016), most of the CNN-based spotters (Lyu et al. 2018; Liao et al. 2021; Liao et al. 2020a; Liu et al. 2020; Wang et al. 2022) follow the detection-then-recognition paradigm. As the prior stage, detection performance plays a dominant role in the whole pipeline. Transformer-based spotters (Zhang et al. 2022; Huang et al. 2022; Ye et al. 2023b; Huang et al. 2023) allow the queries of detection and recognition to interact mutually. Some spotters (Peng et al. 2022; Kim et al. 2022; Liu et al. 2023; Kil et al. 2023) regard the spotting task as sequence generation and try unifying data in multiple OCR-related tasks to improve the performance on scene text spotting.

Although existing spotters achieve remarkable perfor-

[*]These authors contributed equally.
[†]Corresponding Authors.

mances, arbitrary reading order text spotting is still a challenging problem. As shown in Figure 1, while facing the text instances including the ordinary and inverse texts at the same time, there are obvious spotting failures for all the representative methods. ABCNet v2 (Liu et al. 2021a), as a representative CNN-based spotter, is powerless to detect inverse text. The detection errors are accumulated and propagated in the recognition stage with Bezier-Align. SPTS (Peng et al. 2022) is a Transformer-based method in an auto-regressive manner. Compared with ABCNet v2, it alleviates the detection dependency and locates the arbitrary reading order instances well. However, SPTS only uses a single point to represent and perceive each text instance due to lacking fine-grained semantics. That is why SPTS produces many incorrect recognition results despite correct localization. As another Transformer-based method, ESTextSpotter (Huang et al. 2023) also reduces the extreme detection dependency. However, it still fails to spot the inverse texts due to the lack of special design.

To solve such failure cases, we revisit how people spot arbitrary reading order texts accurately. People intuitively pay attention to the coarse location of texts and read them character-by-character, similar to how a well-trained auto-regressive recognizer (Du et al. 2022; Qiao et al. 2021b) works without accurate character localization (Lyu et al. 2024). Motivated by this, we turn the end-to-end text spotting into the recognition problem to alleviate the dependency on detection. However, two crucial problems still need to be solved. Firstly, the recognition model hardly identifies the correct reading order. Therefore, a specific module should be designed to determine where to start reading and in what order. Secondly, text instances are scattered in the scene image. If we recognize the features of each character by traversing the whole image, it will waste a lot of computing resources and have low spotting efficiency.

To overcome the problems mentioned above, we propose a Local-Semantics-Guided Scene Text Spotter (LSGSpotter) to handle the arbitrary reading order problem, which exploits the auto-regressive manner elegantly and facilitates the synergy of detection and recognition. Specifically, we design a Start Point Localization Module (SPLM) to separate distinct text instances elaborately. Note that different from the previous detection module, our localization module relaxes the limitation for the recognition stage. The utilization of SPLM also gets the start points and contributes to the correct reading order. To solve the second issue mentioned above, a Multi-scale Adaptive Attention Module (MAAM) is proposed to adaptively aggregate text features in a local area. In MAAM, we adopt the strategy of grid sampling to alleviate the computational resource. During the inference phase, given a scene text image, after extracting the multi-level features using ResNet50 and FPN, the SPLM predicts the start point according to the image feature. As the reference point, the start point guides the feature sampling at the first step during decoding. Then MAAM gets adaptively local feature grids from the reference position and multi-level image features. The cross-attention module of the Transformer decoder can decode the current character and capture the shift of the next character. The above procedure will be repeated until the end-of-sequence token. Therefore, our LSGSpotter can leverage fine-grained semantic information to auto-regressively decode the complete text instance step by step without sophisticated detection dependency. In conclusion, our contributions are as follows:

- We propose LSGSpotter, a local semantics-guided scene text spotter to handle the arbitrary reading order text instances without sophisticated detection. Our spotter auto-regressively decodes the position and content of characters guided by the local semantic information to alleviate the dependency on detection.
- We introduce two effective modules to solve the arbitrary reading order problem and improve the efficiency of our spotter. Specifically, the Start Point Localization Module (SPLM) aims to localize the reference point for the correct reading order. Guided by local semantics, the Multi-scale Adaptive Attention Module (MAAM) decodes the character shift and content auto-regressively, which enhances the interaction between the position and content information. The strategy of grid sampling in MAAM also releases the computational burden.
- Extensive experiments show our proposed method outperforms InverseText, a specific benchmark for arbitrary reading order. Moreover, we also validate the state-of-the-art performances of LSGSpotter on arbitrarily shaped benchmarks, including 81.5% on Total-Text, and 68.9% on SCUT-CTW1500 without the help of lexicon.

2 Related Works

2.1 CNN-based Methods

CNN-based spotters are derived from general object detectors, and can mainly be divided into top-down manners and bottom-up manners.

Top-down Methods For top-down methods, Li et al. (Li, Wang, and Shen 2017) firstly propose the end-to-end trainable scene text spotter based on CRNN (Shi, Bai, and Yao 2016). To solve the arbitrarily shaped text spotting, Mask TextSpotter series (He et al. 2017; Liao et al. 2021; Liao et al. 2020a) are proposed. Other methods (Lu et al. 2022; Liu et al. 2020, 2021a; Wang et al. 2022; Qiao et al. 2020a) explore various text representations for more accurate text boundaries. AETextspotter (Wang et al. 2020) notices the ambiguity of Chinese layout and introduces the language model to drop out the results of non-semantic character sequences. Top-down spotters adopt the RoI-Align-like methods to align the text features used between detection and recognition prevalently. However, detection-first paradigm methods depend on the accurate detection results extremely.

Bottom-up Methods Some text spotters try to introduce bottom-up manners to alleviate the detection-dependency problem. CharNet (Xing et al. 2019) and CRAFTS (Baek et al. 2020) use character-level annotations to perform character and text detection in a single pass. MANGO (Qiao et al. 2021a) develops the Mask Attention Module to extract the global features for text instances. PGNet (Wang

et al. 2021a) performs the text instances with multi-task objectives, such as the centerline, border offset, direction offset and character sequence prediction. Although bottom-up methods eliminate the dependency of detection, they still use a specially designed polygon restoration process and extra character-level annotations.

2.2 Transformer-based Methods

NAR Methods With the Transformer's successful applications of various visual tasks, recent works (Zhang et al. 2022; Huang et al. 2022; Kittenplon et al. 2022; Huang et al. 2023) firstly explore the DEtection TRansformer (DETR) (Carion et al. 2020) framework as the main architecture of scene text spotters. Compared with CNN-based methods, Transformer-based methods succeed in long-range modeling and produce a more robust performance on scene text spotting. TTS (Kittenplon et al. 2022) and TESTR (Zhang et al. 2022) firstly adopt Transformer into the text spotting task. DeepSOLO (Ye et al. 2023b) improves the initialization of queries fed into the SOLO decoder based on TESTR. ESTextSpotter achieves explicit synergy by modeling discriminative and interactive features for text detection and recognition within a single decoder.

AR Methods AR methods (Kim et al. 2022; Peng et al. 2022; Liu et al. 2023; Kil et al. 2023) model the scene text spotting as a sequence generation task and unify more document tasks. SPTS (Peng et al. 2022) first proposes to transform the text spotting into the sequence generation task, and later SPTSv2 (Liu et al. 2023) accelerates the inference speed by designing a parallel-decoding scheme. UNITS (Kil et al. 2023) tries combining with more datasets related to OCR for training a model to balance multiple tasks. Compared with NAR methods, AR methods can organize more data related to OCR tasks for training easily. However, slow inference speed is still an unresolved problem.

3 Methodology

3.1 Overview

Figure 2 shows the overall architecture of our LSGSpotter. Given a scene text image I with n text instances, an image encoder E, including the backbone and neck network, extracts the multi-level feature maps F. Next, F is flattened and fed into the Start Point Localization Module (SPLM) to get the start point $SP_i = \{(x_i, y_i, p_i)\}_{i=1}^{n}$ of each text instance, where (x_i, y_i) and p_i are coordinates of the start point, and the confidence of i-th text instance respectively. After that, the global image features F and start point SP are fed into the Multi-scale Adaptive Attention Module (MAAM) to decode the character content and shift relative to the position of the former character autoregressively. This section will describe each module of LSGSpotter in detail.

3.2 Start Points Localization Module

Generally, scene texts scatter across the whole image. Therefore, to separate different text instances and avoid calculating the global attention of the whole image, we propose a simple Start Points Localization Module (SPLM) to locate the start points.

Given the global multi-level feature F, a convolutional block maps F into three channels F_s, F_c and F_r. The convolutional block consists of 3 Conv-BN-ReLU layers. Specifically, F_s is the probability map of text start position, which helps decide the reading order of text instances. F_c is the probability map of the text centerline, which assists in separating the adjacent instances. F_r is the probability map of the text region.

$$F^*_{start} = \sqrt{F_s \cdot F_r \cdot F_c} \tag{1}$$

In the inference stage, we fuse three probability maps into a fine-grained start map F^*_{start} as Equation (1). Then the start point $SP = \{x_i, y_i, p_i\}_{i=1}^{n}$ can extracted from F^*_{start}, where (x_i, y_i) is the coordinates of the i-th start point and p_i is corresponding confidence. Given a threshold T, n connected regions can be generated by $M = \{m_i\}_{i=1}^{n} = \{\mathbf{1}|F^*_{start} > T\}$. The i-th start point (x_i, y_i) is the center of the i-th connected regions m_i, and the confidence p_i is the mean value of probability in m_i, termed as $p_i = \{\overline{F(x,y)}|(x,y) \in m_i\}$

During the training stage, polygonal annotations in publicly organized datasets can generate three corresponding ground truths GT_s, GT_c and GT_r. The detailed label generation method of GT_s is described in Figure 3, and GT_c and GT_r are mentioned by PGNet (Wang et al. 2021a). F_r and F_c are supervised by BCELoss and F_s is supervised by Smooth L1 loss. Note that to simplify the calculation of $\mathcal{L}_s$ and $\mathcal{L}_c$, we only consider the part of the text region. The loss of SPLM is termed as $\mathcal{L}_{start}$, which is the sum of optimization targets $\mathcal{L}_s$, $\mathcal{L}_c$ and $\mathcal{L}_r$ of three maps.

$$\mathcal{L}_{det} = \mathcal{L}_s + \mathcal{L}_c + \mathcal{L}_r, \tag{2}$$

3.3 Multi-scale Adaptive Attention Module

For the scene text recognizer based on Transformer, Self-Attention is responsible for obtaining the semantic information and the cross-attention perceives the visual information of the corresponding character. Assume the visual features $F_g \in R^{hw \times d_v}$, where h and w are the height and width of the feature map, and the semantic information from the self-attention is E_{s_t}. The cross-attention calculation is described as Equation (3) and Equation (4):

$$E_{c_t} = Attention(Q_t, K, V) = Softmax(\frac{Q_t K^T}{\sqrt{d}})V, \tag{3}$$

$$Q_t = W_Q E_{s_t}, \quad K = W_K F_g, \quad V = W_V F_g, \tag{4}$$

where $W_Q \in R^{d \times d_e}$, $W_K \in R^{d \times d_v}$, $W_V \in R^{d \times d_v}$. The local features need to be adaptively extracted from F_g. Therefore, the grid window size is estimated by E_{s_t} as Equation (5):

$$gs_t = \{(h_t, w_t)_l\} = Sigmoid(FC(E_{s_t})). \tag{5}$$

The grid size $gs_t \in R^{l \times 2}$ shows the height and width of the perceived range of local attention on different levels

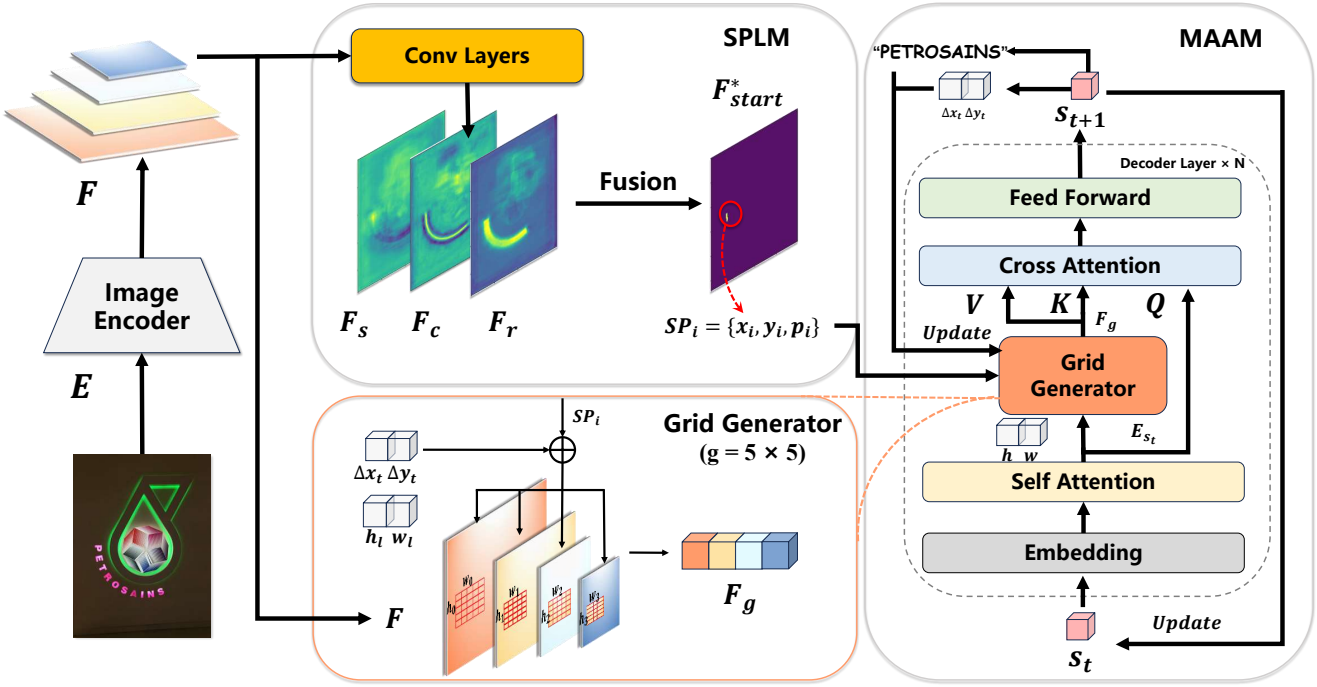

Figure 2: The architecture of LSGSpotter. Image encoder refers to the aggregation of the backbone and neck. SPLM and MAAM are abbreviations of *Start Point Localization Module* and *Multi-level Adaptive Attention Module* respectively. The start point produced by SPLM is the first reference point in the MAAM.

of feature maps. Suppose that the grid size is g. Then the coordinates of grid points can be formulated as Equation (6):

$$gp_{l_t} = \{(-\frac{w_{l_t}}{2} + \frac{w_{l_t}}{g-1}i, -\frac{h_{l_t}}{2} + \frac{h_{l_t}}{g-1}i) + p_t)\} \in R^{g \times g \times 2}, \tag{6}$$

where $i, j = 0, 1, ..., g-1$ represents the point in row i and column j of the grid. $p_t = (p_{t_x}, p_{t_y})$ is equivalent in different levels of feature maps due to the normalization of coordinates. The grid features can be obtained by bi-linear interpolation as Equation (7):

$$F_{g_{t_l}} = Sample(F, gp_{l_t}) \in R^{g^2 \times d_v}. \tag{7}$$

Then visual features can be concentrated as Equation (8), where n is the levels of feature maps and $n = 4$.

$$F_g = Concat(F_{g_{t_0}}, F_{g_{t_1}}, ..., F_{g_{t_n}}). \tag{8}$$

In conclusion, the MAAM module calculates attention on different local text areas when decoding every text instance. This design not only ensures that the decoder perceives the visual information of characters but also prevents the high calculation consumption with the whole image area. The outputs of MAAM are fed into a regression branch to predict the coordinate shift Δx_t and Δy_t of the next character and a classification branch with a Softmax layer to predict the character. As the input of the next step, the coordinate shift and the content of the character in the current step appends into decoded character sequences. The auto-regressive process will be ended until the [EOS] token. Therefore, our spotter makes the decoder decide the ending position independently without sophisticated detection.

3.4 Label Generation

To ensure that the local grid features can be extracted from the corresponding characters in each decoding step, the prediction of reference points should be as accurate as possible, so it is necessary for reference points to be supervised directly. The ideal reference point should be the center of each character. However, due to the expensive cost of annotating character positions, most datasets do not contain character-level annotations, so it is not easy to obtain an accurate character-level center position. Fortunately, the local grid designed in MAAM can cover a certain image range and the grid size is adaptive and variable, so the local grid can adaptively perceive the corresponding feature information through the attention mechanism even if the character reference point's location could be inaccurate. Therefore, we propose a simple strategy of label generation using the existing annotations in the scene text dataset.

Considering the existing datasets in scene text spotting use polygons as the representations of text boundaries, and the order of vertices follows the reading orders. Therefore, we obtain two edges for each text instance along the reading order. Given the length of the text instance m, two edges $E_{top} = \{t_i\}_{i=0}^{m}$ and $E_{bottom} = \{b_i\}_{i=0}^{m}$ are interpolated as $m+1$ points from raw polygons. The center line

(a) (b) (c)

Figure 3: The visualization of Label Generation on different language datasets. Points in different colors represent the different text instances in (a) and (b). The red arrows in (c) show the disturbance shift of center points.

$C = \{c_i\}_{i=0}^{m}$ can be calculated as Equation (9):

$$c_i = \frac{t_i + b_i}{2}. \tag{9}$$

Suppose that each character is equal in width. The t-th character ($t = 0, 1, ..., m-1$) center coordinate r_t can be calculated as Equation (10):

$$r_t = \frac{c_t + c_{t+1}}{2}. \tag{10}$$

Now r_t is regarded as the reference point of t-th character. During the auto-regressive decoding, c_0 and c_m are the [SOS] and [EOS] tokens respectively for character position. The visualization of reference points is shown in Figure 3.

During the training phase, the teacher-forcing training strategy in an auto-regressive manner alleviates the difficulty of optimization, which causes bad performances of our spotter. If the reference points r_t are fed into the decoder during the training stage, while the reference points are predicted value p_t during the inference stage, it will lead to exposure bias in the inference because of the accumulative errors of predicted reference points. Moreover, the precise annotations in the training stage make the model overlook the previous hidden states so that cannot learn the semantic knowledge. To solve these problems, we design a strategy for disturbing the reference points in the training phase. Given a set of reference points $R = \{r_t\}_{t=0}^{n}$, we use Equation (11) to describe this procedure, where η_x, η_y are the disturbing weight of x-axis and y-axis respectively, and distributed between -1 and 1 uniformly. This setting not only makes reference points have a certain disturbance but also prevents the reference point from disturbing too much. The visualization of disturbing reference points is shown in Figure 3 (c).

$$r'_t = r_t + (\frac{\eta_x}{2}|r_{t_x} - r_{(t-1)_x}|, \frac{\eta_y}{2}|r_{t_y} - r_{(t-1)_y}|). \tag{11}$$

3.5 Optimization

The overall loss function $\mathcal{L}$ of LSGSpotter includes two parts, the detection loss $\mathcal{L}_{det}$ and the decoder loss $\mathcal{L}_{dec}$. $\mathcal{L}$ can be represented as Equation (12), where λ is the weight factor for balancing between $\mathcal{L}_{det}$ and $\mathcal{L}_{dec}$:

$$\mathcal{L} = \mathcal{L}_{det} + \lambda \mathcal{L}_{dec}. \tag{12}$$

(a) Without Disturbance (b) With Disturbance

Figure 4: Impact of reference point disturbance strategy on model performance. Without disturbance during training, some characters will be omitted in the inference stage.

In addition, $\mathcal{L}_{dec}$ consists of the reference regression loss and character recognition loss. It can be defined as Equation (13), where y_t and $\hat{y}_t$ are ground truth and prediction of the transcription.

$$L_{dec} = \sum_{t}[-y_t log\hat{y}_t + SmoothL1(\hat{p}_t, p_t)]. \tag{13}$$

4 Experiments

4.1 Settings

Following the settings of previous works, we pre-train our model on SynthText-150k, MLT-2017 (Nayef et al. 2017), ICDAR2013 (Karatzas et al. 2013), ICDAR2015 (Karatzas et al. 2015), TextOCR (Singh et al. 2021) and Total-Text for 600k iterations, which AdamW optimizes with the learning rate of 2e-4 and the weight decay is 1e-4. After pretraining, the model is fine-tuned on the training split of the target benchmark for 200 epochs. The initial learning rate is 1e-4 and declined to 1e-5 on the 60th epoch. The entire model is trained on 4 NVIDIA RTX3090 GPUs with a batch size of 4 on the single GPU. In addition, we utilize the ResNet50 (He et al. 2016) with deformable convolution module (Dai et al. 2017) for the backbone and the 6-layer Transformer decoder for the auto-regressive stage. During the training, the short size of an input image is resized and padded to 960. Random cropping and rotating are employed for data augmentation. In the inference stage, we resize the short edge to 960 while keeping the long side shorter than 1600 with the fixed aspect ratio. For evaluation, we follow the point-based evaluation metrics proposed by SPTS (Peng et al. 2022), because our method only generates center points for representing the position information rather than accurate polygons. SPTSv2 (Liu et al. 2023) has proven the fairness of point-based metrics compared with the box-based.

4.2 Comparison with State-of-the-art Methods

Illustrated by Table 1, our method achieves state-of-the-art results on the InverseText, the most challenging benchmark. Specifically, it presents 73.7% performance without the help of lexicons, being 4.9% better than IAST (Zhang et al. 2024), a scene text spotter specifically designed for inverse texts. Our spotter also achieves 82.3% performance

Methods	InverseText		SCUT-CTW1500		Total-Text	
	None	Full	None	Full	None	Full
Box/Polygon-based Metric						
TextDragon (Feng et al. 2019)	-	-	39.7	72.4	44.8	74.8
ABCNet (Liu et al. 2020)	22.2	34.3	45.2	74.1	64.2	75.7
TextPerceptron (Qiao et al. 2020a)	-	-	57.0	-	69.7	78.3
MaskTextSpotterV3 (Liao et al. 2020a)	-	-	-	-	71.2	78.4
MANGO (Qiao et al. 2021a)	-	-	58.9	78.7	72.9	83.6
PAN++ (Wang et al. 2021b)	-	-	-	-	68.6	78.6
ABCNetv2 (Liu et al. 2021a)	34.5	47.4	57.2	77.2	70.4	78.1
Boundary (Lu et al. 2022)	-	-	-	-	65.0	76.1
TPSNet (Wang et al. 2022)	-	-	60.5	80.1	78.5	84.1
SwinTextSpotter (Huang et al. 2022)	55.4	67.9	51.8	77	74.3	84.1
TESTR (Zhang et al. 2022)	34.2	41.6	56.0	81.5	73.3	83.9
UNITS (Kil et al. 2023)	-	-	<u>66.4</u>	82.3	78.7	86.0
DeepSOLO (Ye et al. 2023b)	64.6	71.2	64.2	81.4	79.7	87.0
ESTextSpotter (Huang et al. 2023)	-	-	64.9	83.9	<u>80.8</u>	<u>87.1</u>
IAST (Zhang et al. 2024)	<u>68.8</u>	<u>80.6</u>	62.4	82.9	71.9	83.5
Point-based Metric						
SPTS (Peng et al. 2022)	38.3	46.2	63.6	83.8	74.2	-
SPTS v2 (Liu et al. 2023)	63.4	74.9	63.6	<u>84.3</u>	75.5	84.0
LSGSpotter	**73.7**	**82.3**	**68.9**	**84.4**	**81.5**	**87.3**

Table 1: End-to-end scene text spotting results on the InverseText, Total-Text, SCUT-CTW1500 English benchmarks. Bold indicates SOTA, and underline indicates the second best. "None" represents lexicon-free, while "Full" indicates all the words in the test set are used.

Settings	Total-Text	SCUT-CTW1500
Without disturbance	77.8	62.6
With disturbance	81.5 (+3.7)	68.9 (+6.3)

Table 2: Ablation experiments about reference point disturbance

F_r	F_c	F_s	Total-Text
✓			76.4
✓	✓		81.1 (+4.7)
✓		✓	80.4 (+4.0)
✓	✓	✓	81.5 (+5.1)

Table 3: Ablation experiments about the approach of starting point localization. F_r, F_c, F_s are the text region map, the text centerline map, and the start point map described in Section 3.2 in detail. The evaluation protocol is "None".

on evaluation in "Full" condition. The main reason is that LSGSpotter has SPLM to locate the start point. It not only learns where the text instance is located but is also implicitly aware of the reading order. The local semantics guidance also leverages fine-grained information to determine the right reading order.

Furthermore, We report experiment results on several public benchmarks, including English benchmarks InverseText, Total-Text, and SCUT-CTW1500. As shown in Table 1, our spotter outperforms with 81.5% on Total-Text and 68.9% on SCUT-CTW1500, 0.7% higher than ESTextSpotter (Huang et al. 2023) and 2.5% higher than UNITS (Kil et al. 2023). With the help of the lexicon, our spotter also significantly surpasses the previous methods on the "Full" evaluation protocol. We believe that the auto-regressive manner aids in learning effective and fine-grained semantic information to decode the scene texts accurately.

4.3 Ablations

The disturbance of reference points Table 2 shows the significant influence of the disturbance of reference points. "With disturbance" expresses that the reference points are disturbed during training. The experiment results show the disturbance of reference points surpasses the "Without disturbance" by 3.7% on Total-Text and 6.3% on SCUT-CTW1500. We explain this improvement by Figure 4. Whether the reference disturbance is used or not, the prediction of coordinates will inevitably have slight errors. The error accumulations in an auto-regressive manner could lead to missing some characters, as shown in Figure 4(a). However, when we use the strategy of reference point disturbance, the model can learn some linguistic knowledge to alleviate the influence of reference shift. Therefore, the design of reference points disturbance during training is necessary.

Ablation on the settings of SPLM In the SPLM, we use three feature maps to locate the reference position. In the ablation study on SPLM, we analyze the necessity of three different features. The experiment results are shown in Table 3. When we only use the F_r to locate the reference point, the F-measure is 76.4% on Total-Text. When we leverage the F_c with F_r, it improves by 4.7% compared with the baseline.

Grid	AS	Grid Size			CTW1500	FPS
		3×3	5×5	7×7		
					58.6	1.1
✓			✓		63.7 (+5.1)	7.6
✓	✓		✓		68.9 (+10.3)	7.4
✓	✓	✓			67.5 (+8.9)	8.4
✓	✓			✓	69.2 (+10.6)	6.9

Table 4: Ablation experiments about MAAM. AS means Adaptive Scale. The evaluation protocol is "None".

Settings	FLOPs
LSGSpotter	194.47G
- Grid Sampling	846.35G

Table 5: Ablations for computational consumption. "LSGSpotter" indicates the default setting and "- Grid Sampling" refers to the configuration of LSGSpotter without grid sampling.

After replacing F_s with F_c, it further enhances the performance of 4.0%. Eventually, we use three feature maps together and obtain 5.1% better than the baseline. The above experiment results prove the effectiveness of the fusion of three feature maps at the same time.

Ablation on the settings of MAAM Table 4 shows the ablation studies results on MAAM, where "Grid" indicates the use of the local grid, and "AS" refers to whether the grid scale is adaptively learned. Removing the local grid means all image tokens from multi-level feature maps participate in cross-attention during decoding. As shown in the 1st and 2nd lines of Table 4, dense attention cannot bring satisfying performance and efficient inference speed. The comparison of the 2nd and 3rd lines of Table 4 validates the performance and efficiency of the adaptive scale. We explain that the adaptive scale makes the decoder perceive the location of each character due to the large-scale variation range of scene texts. Additionally, we quantitatively compare the complexity reduction caused by grid sampling in Table 5. The sampling operation brings about several times increase in efficiency. Furthermore, we attempt three different grid number settings to explore the effect of grid size. The experiments claim that the grid size of 7×7 brings the most enhancement on SCUT-CTW1500. However, it is only a 0.3% improvement compared with the settings of 5×5, but it slows down the inference speed extremely. Therefore, we use 5×5 as the default setting of the grid size.

Ablation on the settings of *T* As shown in Table 6, we perform an ablation experiment on Total-Text using different values of T. Results show that when $0.3 \leq T \leq 0.7$, T slightly impacts detection and spotting performances, because it serves as a threshold for selecting starting points derived from the center of connected regions, which offers robustness to noise. However, when $T \geq 0.8$, recall significantly drops, adversely affecting end-to-end results. Conversely, $T \leq 0.3$, false positives increase significantly, but end-to-end performance remains stable. This suggests that MAAM effectively filters out false positives by directly outputting the [EOS] token upon encountering such points, demonstrating its noise resistance.

T	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
F	87.5	90.4	92.0	93.0	92.6	92.1	88.4	74.5
None	80.0	81.0	81.3	81.5	81.4	81.1	79.1	68.9

Table 6: Ablation for the threshold T of SPLM. "P", "R", "F" and "None" denote precision, recall, and F1-score for detection and end-to-end spotting performance, respectively.

Figure 5: Qualitative results on the testing set of InverseText, Total-Text, SCUT-CTW1500 from left to right in the first line. The second line is the visualization of local grids predicted in MAAM for some challenging text instances. The color from light to deep indicates the decoding order.

4.4 Visualization and Qualitative Analysis

Figure 5 shows the visualization of four public benchmarks mentioned in this paper. It can be observed that our method can accurately locate the start point and recognize the texts. Notably, our method performs well in large aspect-ratio, inverse, and curved text instances by locating the start point and predicting the next character in an auto-regressive manner, which helps determine the reading order of text instances.

5 Conclusion

In this paper, we propose LSGSpotter, a local-semantics-guided scene text spotter. To address the extreme dependency of accurate detection in the whole spotting pipeline, we revisit the recognition process and propose two effective modules, SPLM and MAAM respectively. SPLM locates the start point of the text instance to avoid our spotter paying more attention to the background noise. Moreover, SPLM learns implicit reading orders of text instances and solves the inverse text effectively. MAAM decodes the character shift and content step-by-step. The adaptive local grid attention can save the computational resources further. Extensive experiments show the outperformed performances of our spotter on three English benchmarks. It proves our spotter can handle the arbitrary reading order problem effectively. In the future, We hope our method can inspire further exploration of detection-free spotting.

Acknowledgments

Supported by the National Natural Science Foundation of China (Grant NO 62376266 and 62406318), and by the Key Research Program of Frontier Sciences, CAS (Grant NO ZDBS-LY-7024).

References

Baek, Y.; Shin, S.; Baek, J.; Park, S.; Lee, J.; Nam, D.; and Lee, H. 2020. Character region attention for text spotting. In *ECCV*, 504–521. Springer.

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-end object detection with Transformers. In *ECCV*, 213–229. Springer.

Ch'ng, C. K.; and Chan, C. S. 2017. Total-text: A comprehensive dataset for scene text detection and recognition. In *ICDAR*, volume 1, 935–942. IEEE.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *ICCV*, 764–773.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*.

Du, Y.; Chen, Z.; Jia, C.; Yin, X.; Zheng, T.; Li, C.; Du, Y.; and Jiang, Y.-G. 2022. SVTR: Scene text recognition with a single visual model. In *IJCAI*, 884–890.

Feng, W.; He, W.; Yin, F.; Zhang, X.-Y.; and Liu, C.-L. 2019. Textdragon: An end-to-end framework for arbitrary shaped text spotting. In *ICCV*, 9076–9085.

Guo, Y.; Feng, W.; Yin, F.; Xue, T.; Mei, S.; and Liu, C.-L. 2021. Learning to understand traffic signs. In *ACM MM*, 2076–2084.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask R-CNN. In *ICCV*, 2961–2969.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Huang, M.; Liu, Y.; Peng, Z.; Liu, C.; Lin, D.; Zhu, S.; Yuan, N.; Ding, K.; and Jin, L. 2022. SwinTextSpotter: Scene Text Spotting via Better Synergy between Text Detection and Text Recognition. In *CVPR*, 4593–4603.

Huang, M.; Zhang, J.; Peng, D.; Lu, H.; Huang, C.; Liu, Y.; Bai, X.; and Jin, L. 2023. Estextspotter: Towards better scene text spotting with explicit synergy in Transformer. In *ICCV*, 19495–19505.

Karatzas, D.; Gomez-Bigorda, L.; Nicolaou, A.; Ghosh, S.; Bagdanov, A.; Iwamura, M.; Matas, J.; Neumann, L.; Chandrasekhar, V. R.; Lu, S.; et al. 2015. ICDAR 2015 competition on robust reading. In *ICDAR*, 1156–1160. IEEE.

Karatzas, D.; Shafait, F.; Uchida, S.; Iwamura, M.; i Bigorda, L. G.; Mestre, S. R.; Mas, J.; Mota, D. F.; Almazan, J. A.; and De Las Heras, L. P. 2013. ICDAR 2013 robust reading competition. In *ICDAR*, 1484–1493. IEEE.

Kil, T.; Kim, S.; Seo, S.; Kim, Y.; and Kim, D. 2023. Towards unified scene text spotting based on sequence generation. In *CVPR*, 15223–15232.

Kim, S.; Shin, S.; Kim, Y.; Cho, H.-C.; Kil, T.; Surh, J.; Park, S.; Lee, B.; and Baek, Y. 2022. DEER: Detection-agnostic end-to-end recognizer for scene text spotting. *arXiv*.

Kittenplon, Y.; Lavi, I.; Fogel, S.; Bar, Y.; Manmatha, R.; and Perona, P. 2022. Towards weakly-supervised text spotting using a multi-task transformer. In *CVPR*, 4604–4613.

Li, H.; Wang, P.; and Shen, C. 2017. Towards end-to-end text spotting with convolutional recurrent neural networks. In *ICCV*, 5238–5246.

Li, Y.; Qian, Y.; Yu, Y.; Qin, X.; Zhang, C.; Liu, Y.; Yao, K.; Han, J.; Liu, J.; and Ding, E. 2021. Structext: Structured text understanding with multi-modal Transformers. In *ACM MM*, 1912–1920.

Li, Z.; Shu, Y.; Zeng, W.; Yang, D.; and Zhou, Y. 2024. First Creating Backgrounds Then Rendering Texts: A New Paradigm for Visual Text Blending. In *ECAI*.

Liao, M.; Lyu, P.; He, M.; Yao, C.; Wu, W.; and Bai, X. 2021. Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes. *IEEE TPAMI*, 43(2): 532–548.

Liao, M.; Pang, G.; Huang, J.; Hassner, T.; and Bai, X. 2020a. Mask Textspotter v3: Segmentation proposal network for robust scene text spotting. In *ECCV*, 706–722. Springer.

Liao, M.; Wan, Z.; Yao, C.; Chen, K.; and Bai, X. 2020b. Real-time scene text detection with differentiable binarization. In *AAAI*, volume 34, 11474–11481.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. s. 2016. Ssd: Single shot multibox detector. In *ECCV*, 21–37. Springer.

Liu, Y.; Chen, H.; Shen, C.; He, T.; Jin, L.; and Wang, L. 2020. ABCNet: Real-time scene text spotting with adaptive bezier-curve network. In *CVPR*, 9809–9818.

Liu, Y.; Jin, L.; Zhang, S.; and Zhang, S. 2017. Detecting curve text in the wild: New dataset and new solution. *arXiv*.

Liu, Y.; Shen, C.; Jin, L.; He, T.; Chen, P.; Liu, C.; and Chen, H. 2021a. Abcnet v2: Adaptive bezier-curve network for real-time end-to-end text spotting. *IEEE TPAMI*, 44(11): 8048–8064.

Liu, Y.; Zhang, J.; Peng, D.; Huang, M.; Wang, X.; Tang, J.; Huang, C.; Lin, D.; Shen, C.; Bai, X.; et al. 2023. SPTS v2: single-point scene text spotting. *IEEE TPAMI*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021b. Swin Transformer: Hierarchical vision Transformer using shifted windows. In *CVPR*, 10012–10022.

Lu, P.; Wang, H.; Zhu, S.; Wang, J.; Bai, X.; and Liu, W. 2022. Boundary TextSpotter: Toward Arbitrary-Shaped Scene Text Spotting. *IEEE TIP*, 31: 6200–6212.

Lyu, J.; Wei, J.; Zeng, G.; Li, Z.; Xie, E.; Wang, W.; and Zhou, Y. 2024. TextBlockV2: Towards Precise-Detection-Free Scene Text Spotting with Pre-trained Language Model. *arXiv preprint arXiv:2403.10047*.

Lyu, P.; Liao, M.; Yao, C.; Wu, W.; and Bai, X. 2018. Mask Textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In *ECCV*, 67–83.

Min, W.; Liu, R.; He, D.; Han, Q.; Wei, Q.; and Wang, Q. 2022. Traffic sign recognition based on semantic scene understanding and structural traffic sign location. *IEEE TITS*, 23(9): 15794–15807.

Nayef, N.; Yin, F.; Bizid, I.; Choi, H.; Feng, Y.; Karatzas, D.; Luo, Z.; Pal, U.; Rigaud, C.; Chazalon, J.; et al. 2017. Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-rrc-mlt. In *ICDAR*, volume 1, 1454–1459. IEEE.

Peng, D.; Wang, X.; Liu, Y.; Zhang, J.; Huang, M.; Lai, S.; Li, J.; Zhu, S.; Lin, D.; Shen, C.; et al. 2022. Spts: Single-point text spotting. In *ACM MM*, 4272–4281.

Qiao, L.; Chen, Y.; Cheng, Z.; Xu, Y.; Niu, Y.; Pu, S.; and Wu, F. 2021a. MANGO: A mask attention guided one-stage scene text spotter. In *AAAI*, volume 35, 2467–2476.

Qiao, L.; Tang, S.; Cheng, Z.; Xu, Y.; Niu, Y.; Pu, S.; and Wu, F. 2020a. Text perceptron: Towards end-to-end arbitrary-shaped text spotting. In *AAAI*, volume 34, 11899–11907.

Qiao, Z.; Zhou, Y.; Wei, J.; Wang, W.; Zhang, Y.; Jiang, N.; Wang, H.; and Wang, W. 2021b. PIMNet: a parallel, iterative and mimicking network for scene text recognition. In *ACM MM*, 2046–2055.

Qiao, Z.; Zhou, Y.; Yang, D.; Zhou, Y.; and Wang, W. 2020b. SEED: Semantics enhanced encoder-decoder framework for scene text recognition. In *CVPR*, 13528–13537.

Qin, X.; Lyu, P.; Zhang, C.; Zhou, Y.; Yao, K.; Zhang, P.; Lin, H.; and Wang, W. 2023. Towards Robust Real-Time Scene Text Detection: From Semantic to Instance Representation Learning. In *ACM MM*, 2025–2034.

Shen, H.; Gao, X.; Wei, J.; Qiao, L.; Zhou, Y.; Li, Q.; and Cheng, Z. 2023. Divide rows and conquer cells: Towards structure recognition for large tables. In *IJCAI*, 1369–1377.

Shi, B.; Bai, X.; and Yao, C. 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE TPAMI*, 39(11): 2298–2304.

Shu, Y.; Wang, W.; Zhou, Y.; Liu, S.; Zhang, A.; Yang, D.; and Wang, W. 2023. Perceiving ambiguity and semantics without recognition: an efficient and effective ambiguous scene text detector. In *ACM MM*, 1851–1862.

Singh, A.; Pang, G.; Toh, M.; Huang, J.; Galuba, W.; and Hassner, T. 2021. Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. In *CVPR*, 8802–8812.

Wang, P.; Zhang, C.; Qi, F.; Liu, S.; Zhang, X.; Lyu, P.; Han, J.; Liu, J.; Ding, E.; and Shi, G. 2021a. PGNET: Real-time arbitrarily-shaped text spotting with point gathering network. In *AAAI*, volume 35, 2782–2790.

Wang, W.; Liu, X.; Ji, X.; Xie, E.; Liang, D.; Yang, Z.; Lu, T.; Shen, C.; and Luo, P. 2020. AE Textspotter: Learning visual and linguistic representation for ambiguous text spotting. In *ECCV*, 457–473. Springer.

Wang, W.; Xie, E.; Li, X.; Liu, X.; Liang, D.; Yang, Z.; Lu, T.; and Shen, C. 2021b. PAN++: Towards efficient and accurate end-to-end spotting of arbitrarily-shaped text. *IEEE TPAMI*, 44(9): 5349–5367.

Wang, W.; Zhou, Y.; Lv, J.; Wu, D.; Zhao, G.; Jiang, N.; and Wang, W. 2022. TPSNet: Reverse thinking of thin plate splines for arbitrary shape scene text representation. In *ACM MM*, 5014–5025.

Xing, L.; Tian, Z.; Huang, W.; and Scott, M. R. 2019. Convolutional character networks. In *ICCV*, 9126–9136.

Xu, Y.; Li, M.; Cui, L.; Huang, S.; Wei, F.; and Zhou, M. 2020. LayoutLM: Pre-training of text and layout for document image understanding. In *ACM SIGKDD*, 1192–1200.

Ye, M.; Zhang, J.; Zhao, S.; Liu, J.; Du, B.; and Tao, D. 2023a. DPText-DETR: Towards better scene text detection with dynamic points in Transformer. In *AAAI*, volume 37, 3241–3249.

Ye, M.; Zhang, J.; Zhao, S.; Liu, J.; Liu, T.; Du, B.; and Tao, D. 2023b. DeepSolo: Let Transformer decoder with explicit points solo for text spotting. In *CVPR*, 19348–19357.

Zeng, G.; Zhang, Y.; Zhou, Y.; Yang, X.; Jiang, N.; Zhao, G.; Wang, W.; and Yin, X.-C. 2023. Beyond OCR+ VQA: Towards end-to-end reading and reasoning for robust and accurate TextVQA. *PR*, 138: 109337.

Zeng, W.; Shu, Y.; Li, Z.; Yang, D.; and Zhou, Y. 2024. TextCtrl: Diffusion-based scene text editing with prior guidance control. In *NeurIPS*.

Zhang, R.; Zhou, Y.; Jiang, Q.; Song, Q.; Li, N.; Zhou, K.; Wang, L.; Wang, D.; Liao, M.; Yang, M.; et al. 2019. Icdar 2019 robust reading challenge on reading Chinese text on signboard. In *ICDAR*, 1577–1581. IEEE.

Zhang, S.-X.; Yang, C.; Zhu, X.; Zhou, H.; Wang, H.; and Yin, X.-C. 2024. Inverse-like Antagonistic Scene Text Spotting via Reading-Order Estimation and Dynamic Sampling. *IEEE TIP*.

Zhang, X.; Su, Y.; Tripathi, S.; and Tu, Z. 2022. Text Spotting Transformers. In *CVPR*, 9519–9528.

Zhu, Y.; Liu, Z.; Liang, Y.; Li, X.; Liu, H.; Bao, C.; and Xu, L. 2023. Locate then generate: Bridging vision and language with bounding box for scene-text vqa. In *AAAI*, volume 37, 11479–11487.