A Scene-Text Synthesis Engine Achieved Through Learning From Decomposed Real-World Data

Zhengmi Tang, *Member, IEEE*, Tomo Miyazaki, *Member, IEEE*,
and Shinichiro Omachi, *Senior Member, IEEE*

***Abstract*— Scene-text image synthesis techniques that aim to naturally compose text instances on background scene images are very appealing for training deep neural networks due to their ability to provide accurate and comprehensive annotation information. Prior studies have explored generating synthetic text images on two-dimensional and three-dimensional surfaces using rules derived from real-world observations. Some of these studies have proposed generating scene-text images through learning; however, owing to the absence of a suitable training dataset, unsupervised frameworks have been explored to learn from existing real-world data, which might not yield reliable performance. To ease this dilemma and facilitate research on learning-based scene text synthesis, we introduce DecompST, a real-world dataset prepared from some public benchmarks, containing three types of annotations: quadrilateral-level BBoxes, stroke-level text masks, and text-erased images. Leveraging the DecompST dataset, we propose a Learning-Based Text Synthesis engine (LBTS) that includes a text location proposal network (TLPNet) and a text appearance adaptation network (TAANet). TLPNet first predicts the suitable regions for text embedding, after which TAANet adaptively adjusts the geometry and color of the text instance to match the background context. After training, those networks can be integrated and utilized to generate the synthetic dataset for scene text analysis tasks. Comprehensive experiments were conducted to validate the effectiveness of the proposed LBTS along with existing methods, and the experimental results indicate the proposed LBTS can generate better pretraining data for scene text detectors. Our dataset and code are made available at: https://github.com/iiclab/DecompST.**

***Index Terms*— Scene text synthesis, data augmentation, scene-text detection.**

I. Introduction

DEEP neural networks have demonstrated remarkable success in the field of scene text detection and recognition, yet their performance heavily depends on the quantity and quality of the labeled training data. However, manual collection and labeling of images are costly in terms of both time and resources, and automatic data generation is expected. The image synthesis technique that composes text instances on background images offers a cost-effective and scalable alternative to manual annotation, and this approach has attracted increasing interest in the computer vision community.

Various approaches have been investigated in the development of generation engines for synthetic scene-text images. Initially, based on the observation of real-world data, a set of sophisticated rules has been proposed to guide the design of generation engines. Gupta et al. [1] and Zhan et al. [2] generated synthetic text images from two-dimensional (2D) background images based on different strategies such as region selection, text warping, and text color matching. Liao et al. [3] and Long and Yao [4] further proposed rendering text on the surface of models in three-dimensional (3D) virtual worlds using Unreal Engine. Although realistic occlusions, perspectives, and illuminations can be realized in 3D engines, there is still a gap between the virtual and real worlds. To eliminate heuristic rules and complex setups, Yang et al. [5] proposed a learning-based method consisting of a location module and an appearance module. The location module employs a conditional variational auto-encoder (cVAE) [6] to learn the distribution of text locations directly from the original scene-text image and corresponding text bounding boxes (BBoxes). During training, the cVAE takes a scene text image as input, while during inference, a pure background image is used as input. The "condition" is changed during the training and inference process, which is unreasonable and may limit its performance.

In this study, we aim to address the challenge of inadequate training data and facilitate learning-based text synthesis methods. To this end, we propose the DecompST dataset, which enables the decomposition of real-world scene text images into pure background images and pure text instances. These decomposed data can be utilized to train robust neural networks to learn the complicated layout and appearance of text instances in real-world scene images. The overall concept is illustrated in Fig. 1. Building upon the DecompST dataset, we propose a Learning-Based Text Synthesis engine (LBTS) that mainly includes a text location proposal network (TLPNet) and text appearance adaptation network (TAANet). TLPNet first predicts suitable regions from the background images for text embedding. TAANet then adaptively changes the perspective and color of the synthetic text instance to match the background. Once the networks have been effectively trained, an integrated data generation pipeline can be built to produce a scalable volume of synthetic data, which can

Manuscript received 8 October 2022; revised 15 July 2023 and 14 September 2023; accepted 17 October 2023. Date of publication 27 October 2023; date of current version 1 November 2023. This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 19K12033 and Grant 22H00540. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Xu-Yao Zhang. *(Corresponding author: Zhengmi Tang.)*

The authors are with the Graduate School of Engineering, Tohoku University, Sendai 980-8579, Japan (e-mail: tzm@dc.tohoku.ac.jp; tomo@tohoku.ac.jp; machi@ecei.tohoku.ac.jp).

Digital Object Identifier 10.1109/TIP.2023.3326685

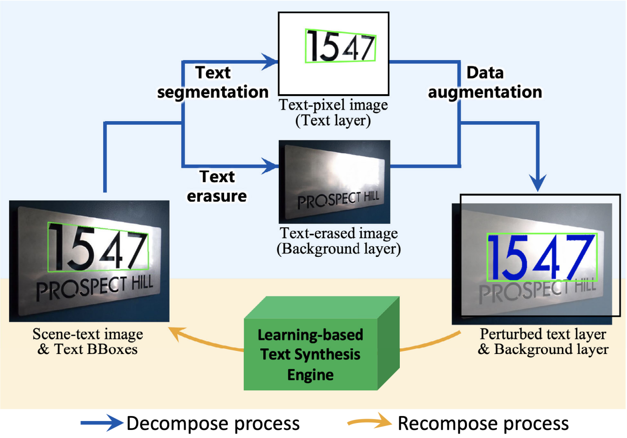

1547
Text segmentation
Data augmentation
Text-pixel image (Text layer)
Text erasure
PROSPECT HILL
Text-erased image (Background layer)
1547 PROSPECT HILL
Scene-text image & Text BBoxes
Learning-based Text Synthesis Engine
Perturbed text layer & Background layer
Decompose process
Recompose process

Fig. 1. Concept of our proposal. We first decomposed the real-world scene-text image into a text layer and a background layer. Next, we applied data augmentation to perturb the geometry and color of the text layer. Then, we proposed a Learning-based Text Synthesis Engine to recompose the two layers back to their original natural relationship, so that the engine can learn the complicated layout and appearance of text instances from real-world scene images.

subsequently be utilized as training data for various scene-text analysis tasks.

The main contributions of our study are summarized as follows:

- We introduce the DecompST dataset, which is able to decompose real-world scene-text images into separate pure background images and text instances, for the training of learning-based scene-text synthesis methods.
- We propose a learning-based scene-text image synthesis engine (LBTS) that consists of a text location proposal network and a text appearance adaptation network, to generate realistic synthetic scene-text images.
- The quality of our generated dataset and other existing synthetic datasets is evaluated by the performance of a baseline text detector. The experimental results demonstrate that our method can generate better pretraining data for scene text detectors than other state-of-the-art methods.

The structure of this paper is organized as follows. Section II reviews related studies on scene-text detection, image synthesis, and data augmentation for scene-text analysis. Section III provides details about the proposed DecompST dataset. Section IV introduces the proposed method, including the flow of data preprocessing and the structure of the two networks. In Section V, we evaluate and compare our proposed method with related synthetic datasets based on experimental results. Finally, concluding statements are presented in Section VI.

II. Related Work

A. Scene Text Detection

With the rise of deep learning, scene text detection has been dramatically reshaped and facilitated, showing promising performance compared to traditional manual feature engineering algorithms [7], [8], [9], [10], [11]. Recent learning-based scene text detection methods have been inspired by general object detection and image segmentation methods, which can be roughly categorized into regression-based and segmentation-based methods. Regression-based methods aim to predict the bounding boxes of text instances directly. TextBoxes [12] modified the anchors in the SSD [13] to handle text with various aspect ratios. CTPN [14] combines the framework of Faster R-CNN [15] with a recurrence mechanism to predict the contextual and dense fixed-width proposals of text. RRPN [16] proposes a rotation region proposal based on Faster R-CNN to bind arbitrary-oriented text with rotated rectangles. EAST [17] proposes a simplified detection pipeline that directly regresses rotated rectangles or quadrangles of text without using anchors. LOMO [18] improved the performance of EAST on the long text and arbitrarily shaped scene text by iteratively refining the preliminary proposals and considering the geometric properties of scene text.

Segmentation-based methods usually first extract text from the segmentation map and then compute the text bounding boxes by post-processing. Zhang et al. [19] integrated semantic labeling using FCN and MSER for pixel-level multi-oriented text detection. The Mask textspotter [20] was inspired by the framework of Mask R-CNN [21] and performed character-level instance segmentation for each alphabet; thus, it has the ability to detect and recognize irregular text. TextSnake [22] proposed a novel and flexible representation of arbitrarily shaped text and predicted heat maps of text centerlines, text regions, radii, and orientations to extract text instances. PSENet [23] gradually expanded small text kernels to complete shapes using multiple segmentation maps to effectively split close text instances. Liao et al. [24] proposed a differentiable binarization (DB) module in a simple segmentation network to perform binarization. CRAFT [25] exploited the affinity between characters in the form of a heat map and proposed a weakly supervised framework to estimate character-level ground truths in existing real word-level datasets. ACE [26] proposed to evolve the key points of the horizontal bounding box towards the corner points to detect arbitrarily-oriented objects or text.

B. Image Synthesis

Inserting foreground objects into a background image is one of the most common image synthesis approaches for generating a photo-realistic composite image, which may face inconsistency problems between the foreground and background in the geometry and appearance domains. To solve these inconsistency problems, many subtasks have been investigated, such as object placement, image blending, image harmonization, and shadow generation. Before the deep-learning era, many researchers explored automated image blending and harmonization. These methods transfer the color from one image to another based on the low-level statistics of the images, such as color distribution or histograms [27], [28], [29], gradient-domain information [30], [31], [32], [33], and multi-scale statistical features [34], among others.

With the emergence of neural networks, more challenging tasks have been investigated. ST-GAN [35] seeks the geometric realism of image compositing by integrating a generative adversarial network (GAN) and spatial transformer networks

(STNs) [36] to warp the foreground object in an iterative fashion. SF-GAN [37] combines an STN and CycleGAN [38] to perform geometry transformation and appearance domain translation concurrently with an end-to-end trainable network. Benefiting from the designed structure, the SF-GAN can also achieve synthesis realism in both geometry and appearance spaces without using paired training data. GCC-GAN [39] was proposed to address geometric and color consistency in composite images by integrating four subnetworks: a transformation network, a refinement network, a discriminator network, and a segmentation network. In the transformation network, not only are the parameters of the transformation matrix predicted, but the parameters of linear color transformation that control the contrast and brightness are also predicted simultaneously. Tsai et al. [40] introduced an end-to-end image harmonization network with a shared encoder and two decoders, where the learned semantic information was used to facilitate harmonization. Inspired by AdaIN [41], Ling et al. [42] treated image harmonization as a background-to-foreground style transfer problem and proposed a plug-and-play region-aware adaptive instance normalization (RAIN) module that explicitly formulates the visual style from the background and adaptively applies it to the foreground.

C. Data Augmentation for Scene Text Analysis

The text synthesis technique, which involves inserting text instances into scene background images, was initially investigated as a data augmentation approach for the training of scene text detection and recognition models. Later, synthetic datasets were utilized as important training data for other tasks such as scene text segmentation [43], [44], scene text erasing [45], [46], and scene text editing [47], [48].

Wang et al. [49] generated a character-centered synthetic image to train a character-level scene-text recognition model. Jaderberg et al. [50] generated a word-centered synthetic dataset using a set of predefined random processes, including font selection and rendering, bordering/shadowing and coloring, layer composition, projective distortion, blending, and noise addition. SF-GAN [37] was trained without paired data because of its unsupervised pipeline, which can also be applied in text synthesis tasks to generate patch-level synthetic text images. Yim et al. [51] further analyzed existing synthesis techniques [1], [50] and integrated the effective parts as a new-generation engine for scene text recognition tasks. These methods generate text-centered images, whose applications are limited.

Gupta et al. [1] first attempted to synthesize text in the wild to generate the SynthText dataset, which is beneficial for training scene-text detection tasks. The SynthText engine finds suitable text embedding regions in the background image following a set of rules that consider semantic segmentation maps and depth maps, and it renders text instances with color selection, perspective distortion, and Poisson blending [30] according to the local background information. Zhan et al. [2] exploited saliency-guided "semantic coherent" image synthesis by leveraging the annotations of semantic segmentation map and visual saliency map. They also designed an adaptive text appearance mechanism to determine the color and brightness of texts by matching a list of pairs, which includes the HoG feature of the background and LAB space statistics of text, gathered from real scene-text images. Yang et al. [5] proposed a learning-based, data-driven text synthesis engine by dividing the text synthesis into two sub-tasks:1) determining the location of text and 2) making the appearance of the inserted text more realistic. A conditional variational auto-encoder [6], [52] was utilized to learn the distribution of text locations from real-world data, and a masked Cycle-GAN [38] was proposed to translate the appearance of synthetic images to the real-data domain. In contrast to rendering text in 2D static images, Liao et al. [3], [4] renders text and the scene as integrity in 3D virtual worlds using the Unreal Engine. In this way, real-world variations, including complex yet correct perspective distortions, various lighting conditions, and occlusions, can be realized in the synthesized scene text images.

In terms of learning-based methods for synthesizing scene-text images, our method is closely related to the method proposed in [5]. Their approach samples latent vectors from the prior distribution and feeds them to a cVAE to directly output the affine transformation parameters, which are used to globally transform the location and perspective of text instances. However, owing to the direct use of scene text images and the corresponding text BBoxes for training, the "condition" of cVAE is changed during the training and inference processes, which may achieve unsatisfactory performance. Our proposed DecompST dataset can address this problem by providing a data pair of text-erased images and original text BBoxes.

Another closely related method is presented in [37], which can concurrently achieve realism in both geometry and appearance spaces without supervision by employing an innovative network structure. In addition, the method in this study can generate patch-level synthetic text images for scene-text recognition tasks. In contrast to their work, our proposed method is a fully supervised image synthesis method that leverages the DecompST dataset, aiming to train more robust networks to generate image-level synthetic scene-text images specifically for the text detection task.

III. DecompST Dataset

We introduce a dataset called DecompST, which is a quadruplet of the original scene-text images, text BBoxes, text-erased images, and stroke-level text masks. This dataset can decompose real-world scene-text images into pure background images and text instances, as shown in Fig. 2. Those components can be utilized to train a robust network to learn the complicated layout and appearance of text instances in real-world scene images. We have made this dataset publicly available and hope that it can motivate more learning-based scene text synthesis methods to generate high-quality synthetic training data for scene text detection and recognition tasks.

A. Image Collection

All the images in our dataset were collected from several public real-world scene text detection benchmarks, including

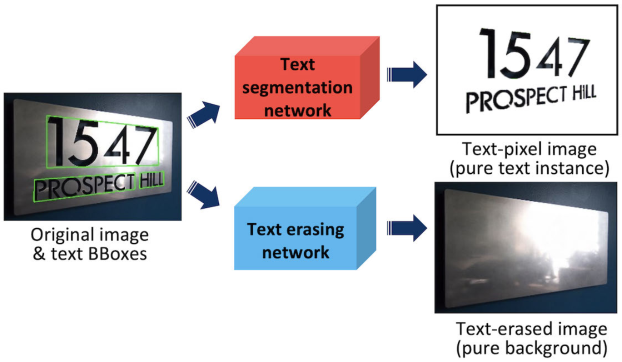

Text segmentation network
Text erasing network
1547 PROSPECT HILL
Text-pixel image (pure text instance)
Original image & text BBoxes
Text-erased image (pure background)

Fig. 2. Given the original image and corresponding text BBoxes, we decompose real-world scene-text images into pure background images and text instances using a text-erased image and stroke-level text mask.

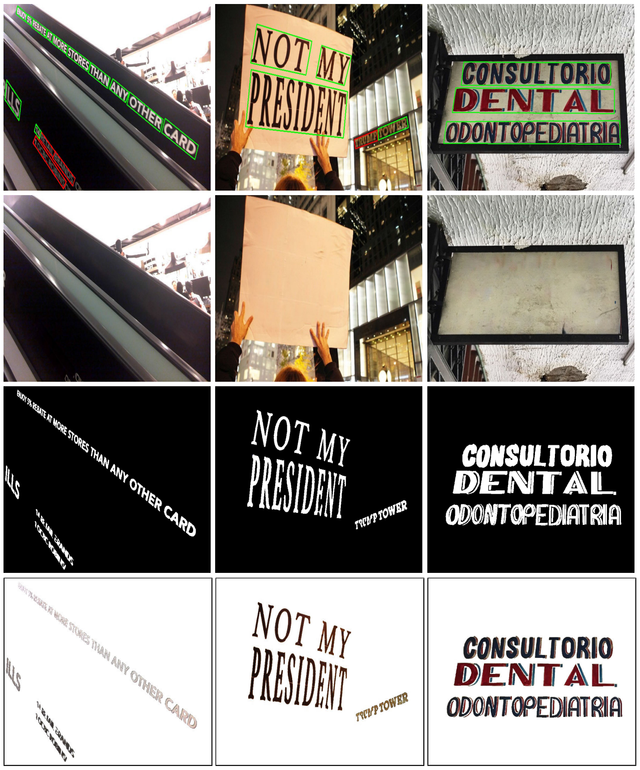
TABLE I
NUMBER OF IMAGES AND VALID TEXT INSTANCES FROM DIFFERENT SOURCE DATASETS

	Images	Text Instances
IC15 [53]	787	1848
MLT19 [54]	1681	6652
SegText [55]	2117	7517
Total	4585	16017

the ICDAR-2015 [53], MLT-2019 [54], and TextSeg [55] datasets. The ICDAR-2015 [53] and MLT-2019 [54] datasets are classic benchmarks for scene text detection. The TextSeg [55] dataset, on the other hand, specifically focuses on scene text segmentation. It provides comprehensive annotations encompassing quadrilateral BBoxes at both word and character levels, along with pixel-level text masks. We opted to use the TextSeg dataset because its manually-labeled, high-quality pixel-level text masks align with our requirements for stroke-level text masks. For each dataset, we collected both the training and validation sets, but we only selected Latin and Chinese parts of the MLT-2019 [54] dataset, and the scene-image part of the TextSeg [55] dataset.

B. Annotation Details

This section provides a detailed description of the annotation process applied to create the DecompST dataset. For each text instance in the collected images, our goal was to obtain the corresponding text-erased patch and stroke-level text mask. Since the text instances in images are already labeled by BBoxes, we utilized a word-level scene-text-erasing method [46] to erase each text instance individually and generate text-erased images. To obtain the stroke-level text mask of the ICDAR-2015 [53] and MLT-2019 [54] datasets, we employed the stroke mask prediction module (SMPM) in [46] to extract the pixel-level text mask. However, as the original SMPM was designed to predict a dilated text mask, we retrained the SMPM using the same synthetic dataset [46], but with original-size text masks as ground truth. Subsequently, this retrained SMPM was utilized to accurately predict text masks that precisely fit the text instances. Given that predictions made by neural networks can sometimes be imperfect, it is necessary to manually label the quality of predicted results.



(a) (b) (c)

Fig. 3. Some image samples from our proposed DecompST Dataset. The first row contains the original images with text BBoxes, where valid text instances are marked in green BBoxes and invalid ones are in red BBoxes. The second row is our generated text-erased images. The third row is the stroke-level text masks. The fourth row is the text-pixel images masked by stroke-level masks.

Our labeling criteria for text-pixel images focused on the readability of text and the integrity of the text mask. As for text-erased images, we assessed the quality based on the effectiveness of text erasure and the restoration of the background. During the annotation process, the annotators checked the text-pixel image and text-erased image of each text instance and labeled both their quality as 1 or 0, where 1 indicated good and 0 indicated bad. Only text instances that received 1 on both sides were considered valid data, and other data were deemed invalid. For the TextSeg dataset, because accurate pixel-level text masks were provided, all text masks were labeled as 1, and we only assessed the quality of the text-erased image, assigning a label of 1 or 0.

Finally, the DecompST dataset contains 4585 images with 16017 valid text instances with corresponding text-erased images, stroke-level text masks, and quadrilateral bounding boxes, as summarized in Table I. Visual samples of annotated instances from the DecompST dataset are presented in Fig. 3.

IV. METHODOLOGY

In this section, we present our proposed learnable text synthesis (LBTS) method, which mainly consists of two subnetworks: the text location proposal network (TLPNet) and the text appearance adaptation network (TAANet), as illustrated in Fig. 4. More concretely, during the training, given a text-erased image, TLPNet first predicts suitable regions for text embedding. Then, a perturbed text layer is added and TAANet adaptively adjusts the perspective and color of the perturbed

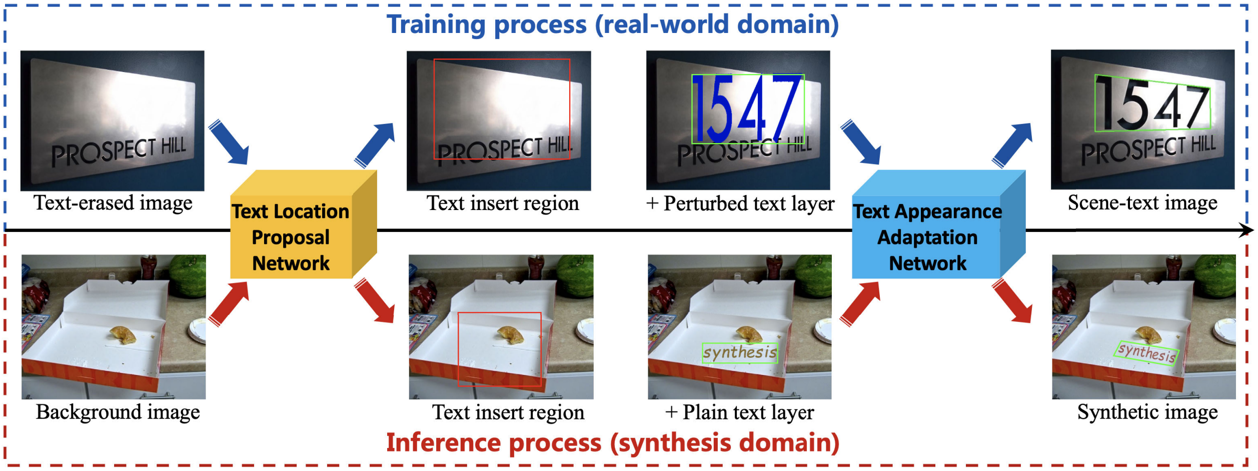


Fig. 4. Pipeline of our proposed Learning-based Text Synthesis Engine (LBTS). It mainly consists of two networks: text location proposal network (TLPNet) and text appearance adaptation network (TAANet). Given a background image, TLPNet predicts suitable regions for text embedding. Then, TAANet aligns the geometric and color relationship between the synthetic text instance and the background. We trained our proposed networks on decomposed real-world data and applied them in the synthesis domain to generate synthetic scene-text images.

text layer to restore its original natural appearance. After training, we can feed two networks with unseen background images and plain text images to generate synthetic scene-text images. Further details regarding the network structure, training process, and inference strategy are presented in the following subsections.

A. Text Location Proposal Network

1) Data Preprocessing in Training: Undoubtedly, the regions within the original BBoxes can be regarded as the ground truth of the text region for learning. Furthermore, we consider that the feasible region for text embedding could be extended if the background shares a similar pattern in a neighboring area, especially in the case of scene text that usually appears in relatively plain regions, such as billboards, walls, and signs. To identify the regions that have a similar appearance to the text-erased regions, we adopted the concept of the appearance descriptor and appearance distance from InstaBoost [56] to measure the appearance similarity between text-erased regions and all other regions within an image. The appearance descriptor $\mathcal{D}(\cdot)$ is a combination of three weighted regions $\mathcal{R}_i$ of each valid text instance in the text-erased image, which is related to the corresponding text location:

$$\mathcal{D}(p_x, p_y) = \{(\mathcal{R}_i(p_x, p_y), w_i) | i \in \{1, 2, 3\}\}, \tag{1}$$

where $\mathcal{R}_1$ denotes the region of the stroke-level mask, and $\mathcal{R}_2$ and $\mathcal{R}_3$ are the dilated contours of the stroke-level mask with different scales ($\mathcal{R}_2$ is the inner contour), given p_x, p_y as the center of the instance. w_i is the weight coefficient of $\mathcal{R}_i$, and $w_1 > w_2 > w_3$ is defined to emphasize the higher similarity around the inner neighboring areas of the original text instance. Fig. 5 (b) shows some examples of visualizations of the descriptor's region $\mathcal{R}_i$ and weight w_i.

Next, given a target text appearance descriptor $\mathcal{D}_t(p_{tx}, p_{ty})$, we assess the appearance similarity between the appearance descriptor of each pixel in the text-erased image and $\mathcal{D}_t$ using the appearance distance. The appearance distance for a given pixel (x, y), conditioned on $\mathcal{D}_t$, can be formulated as follows:

$$d^{\mathcal{D}_t}_{(x,y)} = \min_{(u,v)\in BBOX} \sum_{i=1}^{3} \sum_{\substack{(x_t,y_t)\in\mathcal{R}_{ti}(p_{tx},p_{ty}) \\ (x_s,y_s)\in\mathcal{R}_{si}(p_{tx}-u+x,\,p_{ty}-v+y)}} w_i \Delta(I(x_t, y_t), I(x_s, y_s)), \tag{2}$$

where $BBOX$ is the area inside the original text BBox. $I(x, y)$ denotes the RGB value of the text-erased image on (x, y) pixel coordinates, and Δ is the Euclidean distance. The result of Δ is counted as infinity if (x_s, y_s) is outside the boundary of the text-erased image.

By gathering the appearance distance of each pixel conditioned on the target text instance, we construct the target text appearance distance map H^t_d. $H(x, y)$ denote the value of the map H at pixel coordinates (x, y). Consequently, $H^t_d(x, y) = d^{\mathcal{D}_t}_{(x,y)}$. We generate the corresponding appearance consistency heatmap H^t_a by applying a normalization function to every pixel of the H^t_d, expressed as follows:

$$H^t_a(x, y) = \left(1 - \frac{H^t_d(x, y)}{d_{max}}\right)^3, \tag{3}$$

here, d_{max} is the maximum value in H^t_d except the infinity. During the calculation of Eq. 3, the infinity is set to d_{max}.

For each text instance in an image, we calculate the corresponding appearance consistency heatmaps and combine them into H_a:

$$H_a(x, y) = \max_{k\in W} H^k_a(x, y), \tag{4}$$

where $k \in W$ is the index of the text instance and W denotes the set of valid text instances in the text-erased image.

Up to this point, the appearance consistency heatmap H_a only takes into account the color similarity between patches of valid text instances and other patches in a text-erased image. Therefore, H_a is redundant and lacks semantic information. To address this limitation, we propose a further processing method for H_a by incorporating semantic information provided by the edge map. First, we compute the difference

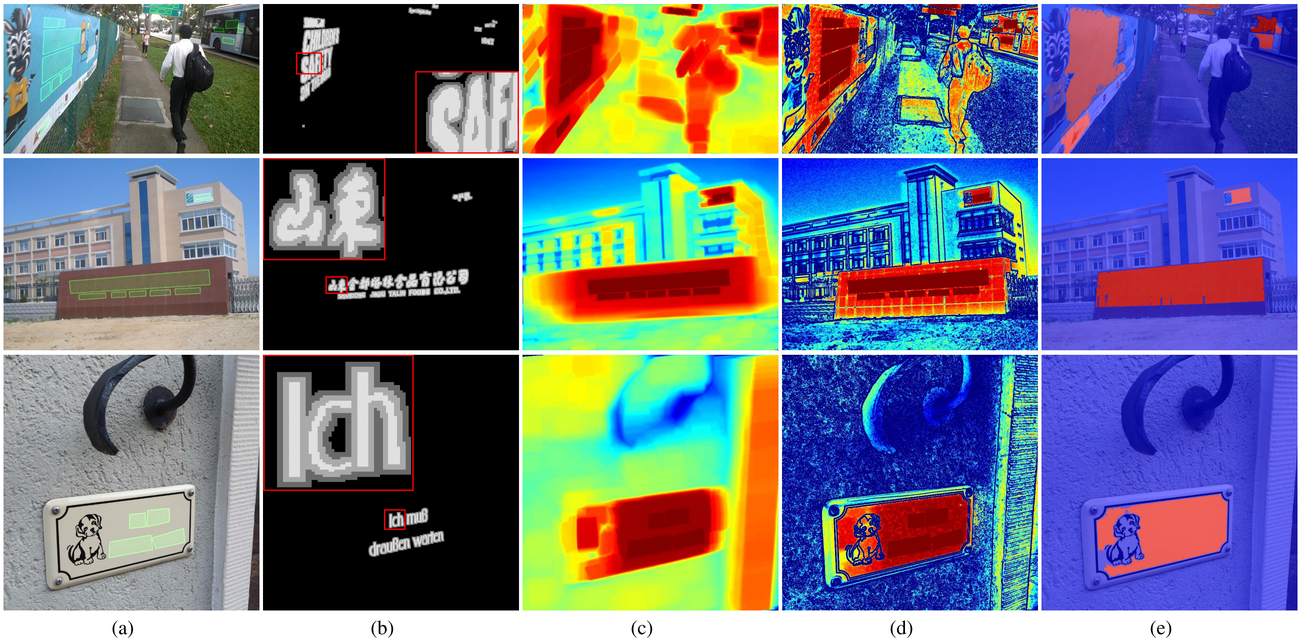


Fig. 5. Flow of data preprocessing. (a) Text-erased image I and *BBOX* regions. (b) Visualization of $\mathcal{R}_i$, w_i. The $\mathcal{R}_i$ refer to the corresponding regions of each text instance and the brighter regions in (b) of $\mathcal{R}_i$ mean higher w_i. (c) Appearance consistency heatmap H_a. (d) Edge-based segmented heatmap H_e. (e) Final generated heatmap H_f (H_f is overlaid on I for a better view). The red regions are treated as GT during the training of TLPNet. Note that the original H_a, H_e, H_f are gray-scale images; we visualized them as heatmaps in this figure.

between the heatmap H_a and the Sobel edge map. This operation can divide H_a with edge information, while it also may disrupt the original *BBOX* regions. To ensure the original *BBOX* regions are completely preserved in the result, we use the following operation:

$$H_e(x, y) = \max\left(H_a(x, y) - \lambda Sobel(I)(x, y), H_{BBOX}(x, y)\right), \tag{5}$$

where I is the text-erased image, and *Sobel* is the Sobel edge detection operation. λ is the weight required to balance the segmentation degree. H_{BBOX} is a heatmap in which pixels inside the valid text BBoxes are set to 1.0; otherwise, 0.

Then, the heatmap H_e is further segmented using thresholding and we obtain H_t:

$$H_t(x, y) = \begin{cases} H_e(x, y), & \text{if } H_e(x, y) > T \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where T denotes a constant threshold. In our implementation, T and λ were set to 0.75 and 5.0, respectively.

Next, we compute all connected components in H_t and mark them as $\mathcal{S}_j$, where j is the index of each segmented region. We filter out small regions and regions that do not contain a high appearance consistency score in $\mathcal{S}_j$ to ensure final text insert regions are the extension of the *BBOX* regions. Finally, we set the values of pixels inside remaining $\mathcal{S}_j$ to 1 and inpaint the small holes to generate the final heatmap H_f as the ground truth for the training of TLPNet. The processing flow of the appearance consistency heatmap is shown in Fig. 5. Through our preprocessing, BBox-based text regions are extended into semantic-based ones by considering the similarity of the regions' appearance.

2) Network Structure of TLPNet: Given a background image I_{bg}, TLPNet aims to segment the mask of the text region H_f, which is suitable for text embedding. We adopted the segmentation head of the DB [24] and used ResNeXt-50 [57] as the backbone for our TLPNet, which is illustrated in Fig. 6. During training, we used a binary cross-entropy (BCE) loss and a DICE loss.

$$L_{bce}(S, T) = -(T\log(S) + (1-T)\log(1-S)) \tag{7}$$

$$L_{dice}(S, T) = 1 - \frac{2\sum_i^N S_i T_i}{\sum_i^N S_i + \sum_i^N T_i} \tag{8}$$

$$L_{TLPNet} = \lambda_0 L_{bce}(\hat{H}_f, H_f) + L_{dice}(\hat{H}_f, H_f), \tag{9}$$

where S and T represent the prediction and ground truth of the mask image, respectively, and N denotes the total number of pixels in the image. $\hat{H}_f$ and H_f are the prediction and ground truth of TLPNet, respectively. λ_0 is set as 10 in our implementation.

B. Text Appearance Adaptation Network

We consider that the realism of text appearance has two aspects: proper perspective and harmonious color that align with the background context. To address this, our TAANet comprises 1) a geometry transformation module (GTM) and 2) a color harmonization module (CHM), as illustrated in Fig. 7. For the GTM, there are three inputs: a patch-level plain text image P_{pt}, a background image I_{bg}, and a reference rectangle *Rect* indicating the approximate location and scale of the text in the background image. The GTM outputs a composed image $\hat{I}_{comp}$, where P_{pt} is transformed by homography matrices to fit the local geometric context of the background based on *Rect*. P_{pt} is a fixed-size text-centered patch image in which text is placed horizontally. In the CHM, the composed image $\hat{I}_{comp}$ and its corresponding text mask I_{ttA} are taken as inputs, and the output $\hat{I}_{out}$ is an image in which the color of the text is properly transferred to harmonize with the background.

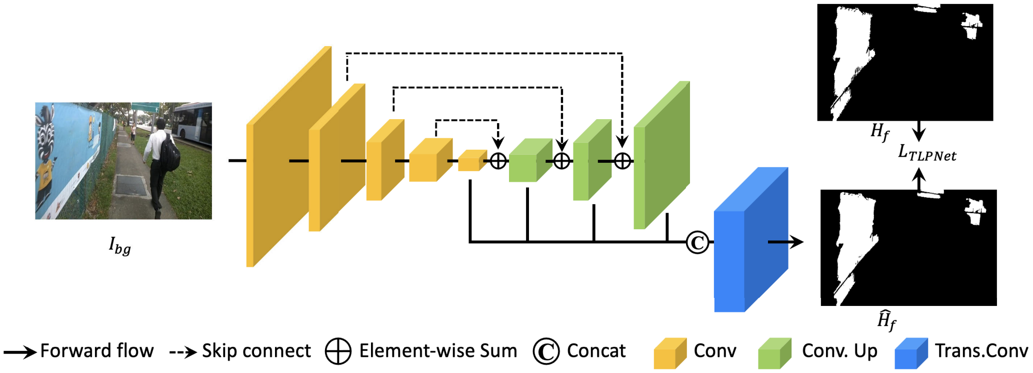


Fig. 6. Structure of the text location proposal network. Given a background image I_{bg}, TLPNet aims to segment the text region, which should be as close to the heatmap H_f.

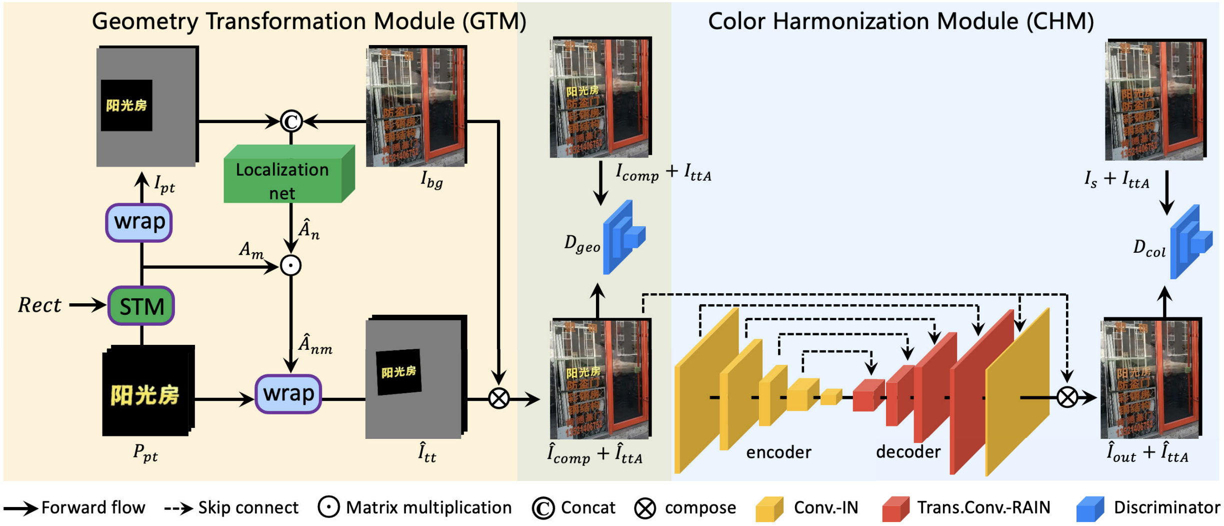


Fig. 7. Overview of our proposed text appearance adaptation network. It is composed of a geometry transformation module (GTM) (left) and a color harmonization module (CHM) (right). Given an input triplet consisting of a patch plain-text image P_{pt}, a reference rectangle $Rect$, and a background image I_{bg}, the GTM learns to place the text with a realistic perspective. Then, the CHM takes the composite image I_{comp} and text mask I_{ttA} as inputs, and outputs a synthetic text image I_{out} with a harmonious color.

1) Data Preprocessing: In a simple image-level text synthesis scenario, we are provided with a background image, a plain text patch, and a hint indicating the rough location of the text. Accordingly, given the BBox and stroke-level mask of one text instance from the source scene-text image I_s in the DecompST dataset, preprocessing aims to remove the original geometry and color information of text instances to obtain a patch-level plain text image P_{pt} and a reference rectangle $Rect$. P_{pt} is a text instance with a perturbed color and horizontal layout, while $Rect$ indicates the approximate location and scale of the text within I_s. By restoring P_{pt}, $Rect$, and the text-erased image I_{bg} back to I_s, the geometry and color relationship between text and background can be learned through TAANet.

The first step of preprocessing is to cut off the target text instance from the text-pixel image and apply a perspective transformation to warp the target text instance into a rectangular one based on its quadrilateral-BBox annotation so that we obtain a horizontal text instance without perspective. Sequentially, we augment the data by randomly altering the aspect ratio of the rectangle BBox and jittering the center of the rectangle BBox, to further perturb the geometric relationship between the target text instance and the background. Next, the text pixels of the target text instance are clustered in only two or three colors using K-means to remove color information and noise. In addition, we augment the data by jittering the color of the text in the HSL space. Finally, to reduce the interdependence between text instances within an image, other text instances are randomly erased in the background image. The entire process flow is shown in Fig. 8.

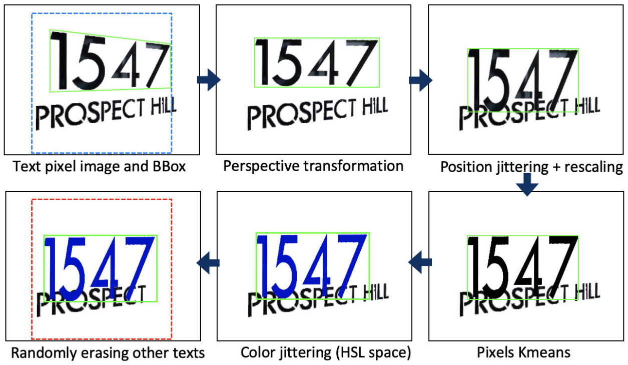

1547
PROSPECT HILL
Text pixel image and BBox
Perspective transformation
Position jittering + rescaling
Randomly erasing other texts
Color jittering (HSL space)
Pixels Kmeans

Fig. 8. Flow of the preprocessing of training data in TAANet. The blue and red dashed boxes are the same reference rectangle $Rect$ but in the images before and after the processing to show the clipping regions to obtain patch text images.

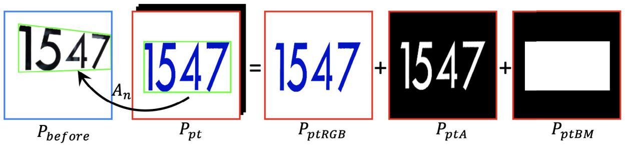


Fig. 9. Illustration of A_n and P_{pt}. The images in blue box P_{before} and red box P_{pt} are obtained by cropping from the dash boxes with the same color in Fig. 8.

Based on the aforementioned processing, we can obtain the reference rectangle $Rect$, patch-level plain text image P_{pt}, background image I_{bg}, and ground truth of the transformation matrix A_n using the following operations. $Rect$ is a square box centered on the processed target text instance. Using $Rect$, the target text instances before and after processing are cropped, resized, and padded to create P_{pt} and the text image before processing P_{before}. A_n is computed based on the transformed BBox in P_{pt} and the original BBox in P_{before}. Moreover, P_{pt} is a five-channel image with RGB channels P_{ptRGB}, alpha channel P_{ptA}, and a mask channel of the BBox-level of the text region P_{ptBM}, as shown in Fig. 9. P_{ptBM} is utilized as additional information during network training, which will be discussed in the later section. Finally, I_{bg} is generated by composing the remaining text in the processed text-pixel image and text-erased image.

2) Geometry Transformation Module (GTM): The first step of the GTM is to feed $Rect$ into a spatial transformer module (STM) [58] and generate a transformation matrix A_m parameterized by θ_m. The A_m is used to warp and pad the patch-level plain text image P_{pt} into the plain-text image I_{pt}. Then, the I_{pt} and background image I_{bg} are concatenated and fed into the localization network (ResNet-34 [59]) to regress the parameters θ_n of the homography transformation matrix A_n. Once the transformation matrices A_m and A_n are obtained, they are applied to the P_{pt} to sample the transformed text image I_{tt}. In the GTM, A_m is used to determine the coarse location and scale of the text based on the $Rect$, and A_n is used to transform the local perspective of the text instance. The transformation is expressed as follows:

$$\begin{pmatrix} x_i^{tt} \\ y_i^{tt} \end{pmatrix} = \mathcal{T}_{\theta_m}(\mathcal{T}_{\theta_n}(G_i)) = A_m A_n \begin{pmatrix} x_i^{pt} \\ y_i^{pt} \\ 1 \end{pmatrix}, \tag{10}$$

where $\mathcal{T}_\theta$ is a 2D perspective transformation and G_i is a pixel in a regular grid G, which is the same as the grid in P_{pt}. Therefore, $G_i = (x_i^{pt}, y_i^{pt})$, which are the coordinates of P_{pt}, and (x_i^{tt}, y_i^{tt}) are the corresponding coordinates in the warped grid that defines the sample points.

$$I_{tt} = \mathcal{S}(\mathcal{T}_{\theta_m}(\mathcal{T}_{\theta_n}(G)), P_{pt}), \tag{11}$$

where $\mathcal{S}$ represents the differentiable bilinear sampler [36] that computes the pixel value of I_{tt} by interpolating the corresponding neighbor pixels in P_{pt}.

After obtaining the transformed text image I_{tt} and background image I_{bg}, we can compose them to obtain I_{comp}:

$$I_{comp} = I_{ttRGB} \circ I_{ttA} + I_{bg} \circ (1 - I_{ttA}), \tag{12}$$

where $\circ$ is the Hadamard product. I_{ttRGB} and I_{ttA} are the RGB channels and pixel-level alpha channel of I_{tt}.

During the training, we introduce three loss functions to stabilize the training of the geometry transformation module: local L1 loss, global region loss, and adversarial loss. We use a robust smooth-L1 loss [60], as the local L1 loss directly restricts the output of the localization network from a numerical perspective:

$$L_1 = \mathrm{smooth}_{L1}(\hat{A}_n - A_n), \tag{13}$$

$$\mathrm{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise}, \end{cases} \tag{14}$$

where $\hat{A}_n$ and A_n represent the prediction and ground truth of the localization network output, respectively.

The region loss employs the DICE loss in Eq. 8 to guide the transformed text with a higher overlapping rate from the view of the region, and we globally apply it to the stroke-mask level and BBox-mask level in the image:

$$L_{region} = L_{dice}(\hat{I}_{ttA}, I_{ttA}) + L_{dice}(\hat{I}_{ttBM}, I_{ttBM}), \tag{15}$$

here, $\hat{I}_{ttA}$ and $\hat{I}_{ttBM}$ are generated by transforming P_{ptA} and P_{ptBM} using the matrices A_m and $\hat{A}_n$. I_{ttA} and I_{ttBM} are the corresponding ground truths that can be easily generated from stroke-level text masks and text BBoxes.

GAN [35], [37], [39] has been proven beneficial for the training of STN, so we adopt it in our implementation. However, we do not directly use the source image I_s as the "real image" in adversarial training because I_s is realistic in both the geometry and color spaces. Instead, we generate I_{comp} by warping I_{pt} using A_n, which only achieves realism in the geometry domain. I_{comp} is treated as a "real image" during the training of the GTM. The adversarial loss is defined as follows:

$$\begin{aligned} L_{D_{geo}} &= \mathbb{E}_{I_{comp}}[\mathrm{ReLU}(1 - D_{geo}(I_{comp}, I_{ttA}))] \\ &\quad + \mathbb{E}_{\hat{I}_{comp}}[\mathrm{ReLU}(1 + D_{geo}(\hat{I}_{comp}, \hat{I}_{ttA}))] \end{aligned} \tag{16}$$

$$\begin{aligned} L_{GTM} &= \lambda_1 L_1 + \lambda_2 L_{region} \\ &\quad - \mathbb{E}_{\hat{I}_{comp}}[D_{geo}(\hat{I}_{comp}, \hat{I}_{ttA})], \end{aligned} \tag{17}$$

where I_{comp} and I_{ttA} are concatenated as the inputs of the discriminator. λ_1 and λ_2 are set to 50 and 10, respectively, in our experiment.

3) Color Harmonization Module (CHM): We treat this text-color-changing task as an image-harmonization problem. We employ the region-aware adaptive instance normalization (RAIN) module [42] in a UNet-like architecture by adding RAIN modules after the convolutional layers in the decoding stage. RAIN is proposed as an activation function that normalizes the foreground features and aligns the normalized features with a computed scale and bias from the background features. In our task, we hope that it can transfer the style from the background into text instances, maintaining harmony between texts and the background. Given an input feature batch $F \in \mathbb{R}^{C \times H \times W}$ and resized foreground (text) mask $M \in \mathbb{R}^{H \times W}$, the formulation of RAIN$(\cdot)$ is expressed as:

$$\begin{aligned} \mathrm{RAIN}(F, M) &= \sigma(F, 1 - M)\left(\frac{F - \mu(F, M)}{\sigma(F, M)}\right) \\ &\quad + \mu(F, 1 - M), \end{aligned} \tag{18}$$

where $\mu(\cdot)$ and $\sigma(\cdot) \in \mathbb{R}^C$ are the channel-wise mean and standard deviation of the foreground or background features, respectively, computed independently across spatial dimensions for each channel.

$$\mu_c(F, M) = \frac{1}{\sum\limits_{h,w} M} \sum_{h,w} F_{c,h,w} \circ M_{h,w} \tag{19}$$

$$\sigma_c(F, M) = \sqrt{\frac{1}{\sum\limits_{h,w} M} \sum_{h,w} (F_{c,h,w} \circ M_{h,w} - \mu_c(F, M))^2 + \epsilon}, \tag{20}$$

where $\circ$ denotes the Hadamard product.

In addition, we adopt an adversarial training method. Adversarial loss can be expressed as follows:

$$\begin{aligned} L_{D_{col}} &= \mathbb{E}_{I_s}[\mathrm{ReLU}(1 - D_{col}(I_s, I_{ttA}))] \\ &\quad + \mathbb{E}_{\hat{I}_{out}}[\mathrm{ReLU}(1 + D_{col}(\hat{I}_{out}, \hat{I}_{ttA}))] \end{aligned} \tag{21}$$

$$L_{CHM} = \lambda_3 \| \hat{I}_{out} - I_s \| - \mathbb{E}_{\hat{I}_{out}}[D_{col}(\hat{I}_{out}, \hat{I}_{ttA})]. \tag{22}$$

Here, λ_3 is set to 5 in the experiment.

4) Inference Pipeline: After training the TLPNet and TAANet, they can be integrated into a generation pipeline to generate synthetic data. The inference process of our method is illustrated in the lower section of Fig. 4. Given a background image, We first use TLPNet to predict the text regions in the form of heatmaps. Subsequently, we randomly sample a reference rectangle with a higher 70% overlap rate with the text regions. At the same time, a plain text patch image with a size of 256 × 256 is generated by randomly selecting fonts, text, and color. Then, the reference rectangle, plain text patch image, and background image are passed through TAANet to produce a synthetic text image. Finally, post-processing applies various effects to the text, including shadows, 3D effects, texture, and blurring. In the composition of the multiple text instances within one background image, we abandon overlapped and small text instances. In the presence of semantic information, such as in the COCO dataset [61], the refinement of the synthesis can be achieved by discarding the text beyond the boundaries of semantic segmentation, allowing for the synthesis of text instances specifically on designated objects.

V. Experiment

A. Implementation Details

1) Training Configurations: Our implementation was based on the PyTorch framework. For training of TLPNet, we used the DecompST and the SCUT-EnsText datasets [62] to generate the training data pairs. As a result, we obtained a total of approximately 7900 training data pairs. The input size of the TLPNet was set to 768 × 768, and the batch size was 12 on an Nvidia GeForce RTX 3090 GPU. We employed the Adam [63] optimizer with a β of (0.5, 0.9), and the learning rate started at 0.0002 and decayed to nine-tenths after every 20 epochs in the training phase. During the training of TAANet, GTM and CHM were trained separately. This is because we adopted the L1 loss during the training of CHM, which is essential for effectively constraining the color of the output. The input size of TAANet was also 768 × 768, and the training batch size for GTM and CHM were set to 20 and 10, respectively, on a single Nvidia GeForce RTX 3090 GPU. The optimizer used was the same as in TLPNet, and the discriminators' learning rate started from 0.0004, with the same decay rate as that in TLPNet.

2) Inference Configurations: In the preparation stage, we need to collect some ingredients for synthesis, including background images, fonts, and a lexicon. The background images were collected from the COCO dataset [61] and Places2 dataset [64]. To ensure that the images closely resembled real scene images, we selected the image sets by excluding those with labels related to natural landscapes. Additionally, we applied filtering to the selected image sets using CRAFT [25] and DB [24] to remove any images with prominent text. Ultimately, we amassed a collection of approximately 200,000 background images. Furthermore, we gathered around 2000 fonts and compiled a lexicon by combining the MJ dataset [50] and the ST dataset [1]. Our LBTS dataset is generated by a machine with a single GeForce RTX 3080 GPU, AMD Ryzen7 3700X @ 3.6 GHz CPU, and 32G RAM. The TLPNet model consists of 24.7M parameters, while the TAANet model has 38.5M parameters (21.4M for GTM and 17.1M for CHM). The inference times for TLPNet on a single image and TAANet on one text instance are approximately 21ms and 81ms (11ms for GTM, and 70ms for CHM), respectively. Fig. 10 shows some generated samples from our LBTS dataset. We observed that the TLPNet exhibited a preference for predicting the text region in relatively flat areas, especially in regions with quadrilateral shapes. This tendency may stem from the bias in the training data, where most text instances exist on the signs, walls, or billboards. On the other hand, the geometry and color relationship between text and background is also reasonably aligned by the TAANet. The text perspective accurately follows the boundaries of text regions, and the text color is appropriately balanced, neither being obtrusive nor excessively dull.

B. Evaluation Metrics and Datasets

1) Evaluation Metrics: To verify the effectiveness of different text synthesis methods, a common method is to train the same text detector on different synthesized datasets and evaluate the trained detectors on several test sets of real datasets. The better performance of the text detector indicates a higher quality of the training data, implying a better text synthesis strategy. Following previous works [2], [4], synthetic datasets are evaluated from two perspectives: 1) as *independent training data* for detection models to assess the possibility that whether synthetic datasets can be a substitute for real-world datasets. 2) as *pretraining data* to initialize text detectors, where pretrained models fine-tuned with real-world data usually exhibit better performance than models directly trained from scratch with real-world data.

In our experiment, we selected EAST [17] and DB [24] as the baseline text detector to conduct comparison experiments. Both of them were previous state-of-the-art methods and are the most commonly used algorithms in the text detection task. In the implementation of EAST, ResNet-50 [59] was used as the backbone, and all the models were trained on two RTX

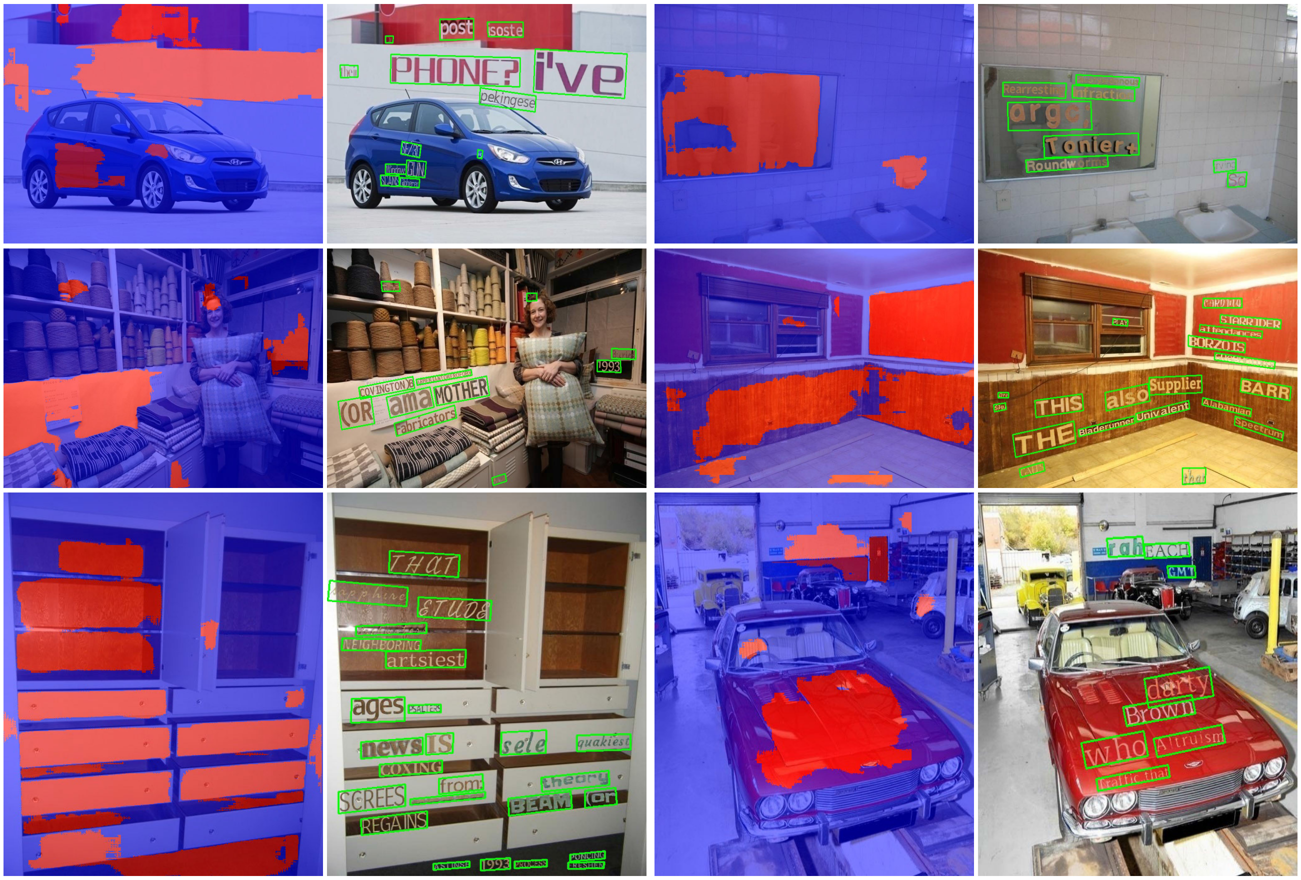


Fig. 10. Several sample images generated by our proposed synthesis engine. The left column of the paired images displays the predicted text regions using TLPNet, and the right column of that is our synthesized image.

2080Ti GPUs with a batch size of 28. For DB, we trained DB-ResNet-50 [24] on one RTX 3090 with a batch size of 20. The performance metrics of the text detector, recall (R), precision (P), and F-score (F), were calculated under the ICDAR2015 evaluation protocol [53] over all evaluation datasets.

2) Synthetic Dataset:

- **Oxford SynthText Dataset (ST)** [1] is a large-scale synthetic text dataset that consists of about 850,000 images. It is created from about 8000 background images and 1200 fonts. 10,000 data pairs were randomly sampled from this dataset to compose ST-10k.
- **Verisimilar Image Synthesis Dataset (VISD)** [2] contains 10,000 images synthesized from background images collected from the COCO dataset [61].
- **UnrealText (UT)** [4] initially consists of about 728,000 images in English/Latin. However, we discovered that some of these images either do not contain text or are partially black, potentially due to render failure or incorrect camera positioning. To ensure data quality, we filtered out the images without annotations and those where more than two-thirds of the pixels are completely black. As a result, approximately 670,000 images remained, and we also randomly sampled 10,000 images to form UT-10k for our experiment.

3) Real-World Dataset:

- **ICDAR 2013 (IC13)** [65] is a widely used scene text image dataset that includes 229 training images and 233 testing images.
- **ICDAR 2015 (IC15)** [53] comprises 1000 training images and 500 test images and addresses incidental scene text in the Latin alphabet.
- **ICDAR 2017 MLT (MLT17)** [66] contains 7200 images for training and 1800 images for validation. Text instances of this dataset are from nine different languages: Arabic, Bangla, Chinese, English, French, German, Italian, Japanese and Korean.
- **Total-Text** [67] is a comprehensive dataset of arbitrary-shaped text instances, including horizontal, multi-oriented, and curved textual variations. It contains 1255 training images and 300 test images. All images are annotated with polygons at the word level.

C. Comparison With State-of-the-Art Methods

To verify the effectiveness of the proposed text synthesis engine, we conducted evaluation experiments to compare our generated LBTS dataset with those of recent state-of-the-art approaches [1], [2], [4]. First, we standardized the total number of each synthesis dataset to 10k to conduct a fair comparison experiment. We trained EAST on each synthetic dataset with 200,000 steps, followed by fine-tuning on the corresponding real-world training set for an additional 200,000 steps. The performance of EAST was evaluated by the validation set of each real dataset every 1000 steps, and the best F-scores are recorded in Table II. For all the evaluation benchmarks, when we employed synthetic datasets as independent training

TABLE II

Comparison Between Previous Synthetic Datasets and Our LBTS Dataset on the ICDAR2013, ICDAR2015, ICDAR2017MLT Datasets Using EAST as the Baseline Detector. R: Recall, P: Precision, F: F-Score, Real: The Corresponding Training Set of the Evaluation Dataset

Train dataset	IC13			IC15			MLT17		
	R	P	F	R	P	F	R	P	F
ST-10k	71.69	73.09	72.38	50.22	64.98	56.65	40.78	55.93	47.17
VISD-10k	**75.71**	74.68	**75.19**	**59.94**	72.17	**65.49**	**42.58**	61.19	**50.21**
UT-10k	64.2	**86.58**	73.73	51.52	**77.2**	61.8	37.36	**65.65**	47.62
LBTS-10k	59.18	82.23	68.83	42.95	68.14	52.69	30.28	61.56	40.59
Real	68.22	86.66	76.34	74.58	84.32	79.15	56.09	72.94	63.41
ST-10k + Real	75.16	86.54	80.45	79.15	84.57	81.77	57.41	73.47	64.46
VISD-10k + Real	75.16	89.65	81.77	79.97	85.84	82.8	56.74	**75.06**	64.62
UT-10k + Real	75.25	87.75	81.02	80.07	85.68	82.78	57.11	74.11	64.51
LBTS-10k + Real	**75.53**	**89.89**	**82.08**	**81.03**	**86.66**	**83.75**	**57.63**	74.49	**64.98**

TABLE III

Comparison Between Previous Synthetic Datasets and Our LBTS Dataset on the ICDAR2015 and Total-Text Datasets Using DB as the Baseline Detector. R: Recall, P: Precision, F: F-Score, Real: The Corresponding Training Set of the Evaluation Dataset

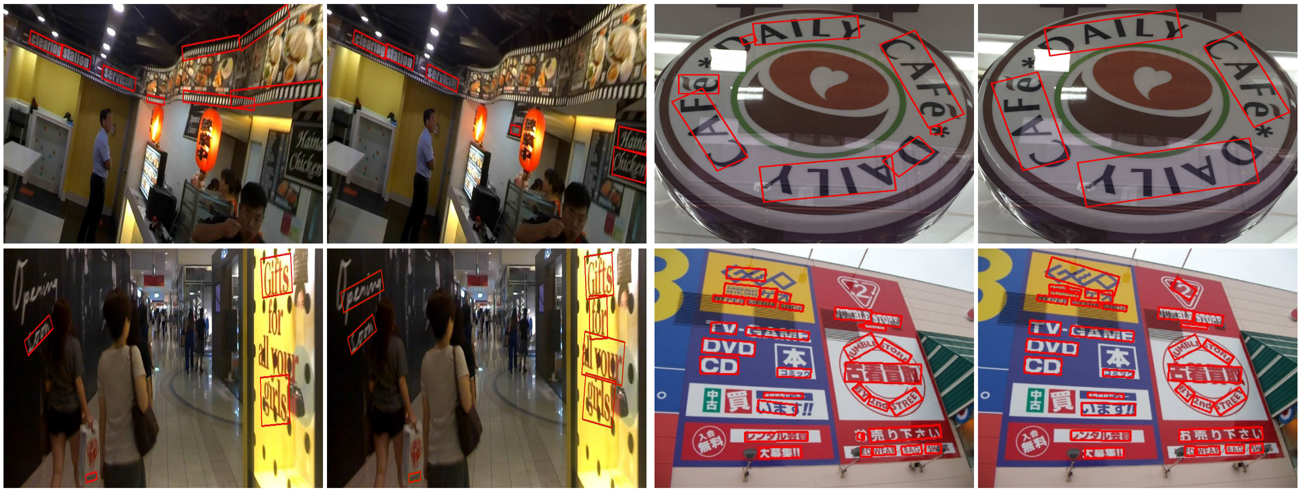
Train dataset	IC15			Total-Text		
	R	P	F	R	P	F
ST-10k	46.89	70.84	56.43	39.64	69.3	50.43
VISD-10k	**57.68**	71.57	**63.88**	**43.66**	**73.31**	**54.73**
UT-10k	54.94	**72.26**	62.42	41.4	63.5	50.12
LBTS-10k	37.07	65.98	47.47	41.17	66.23	50.78
Real	82.23	86.88	84.49	82.39	85.12	83.73
ST-10k + Real	82.67	89.99	86.17	**83.7**	87.25	85.44
VISD-10k + Real	83.1	89.29	86.08	82.71	87.74	85.15
UT-10k + Real	82.91	**90.11**	86.36	83.3	87.86	85.52
LBTS-10k + Real	**84.59**	89.32	**86.89**	82.12	**89.3**	**85.56**

data, EAST trained on VISD-10k achieved the highest F-score and Recall, and EAST trained on UT-10k achieved higher Precision. However, when we fine-tuned the pretrained EAST with real-world data, we observed that our LBTS-10k dataset outperformed all other synthetic datasets, obtaining 0.31%, 0.95%, and 0.36% improvement of the F-score on IC13, IC15, and MLT17 datasets over VISD-10k.

We also trained DB in a similar manner to compare the quality of synthesis datasets. Initially, DB was pretrained on each synthetic dataset for 100,000 steps and then fine-tuned on the IC15 or Total-text datasets for another 1200 epochs. During the training, we validated the model with the corresponding test set every 2000 steps, and Table III presents the best F-scores obtained. The results showed that using DB as the baseline detector yielded similar results as using EAST. When considering synthetic datasets as independent training data, the VISD-10k achieved the highest F-score for both IC15 and Total-Text datasets. However, by further fine-tuning the DB model, pretrained on synthetic data, with real data, our LBTS-10k dataset obtained a higher F-score than other datasets. Compared to the F-score of DB trained from scratch, DB pretrained with our dataset gained 2.4% and 1.83% on IC15 and Total-Text, respectively. Furthermore, in comparison to previous state-of-the-art datasets, we observed a commendable improvement of 0.53% in F-score on IC15, while achieving competitive performance on the Total-Text dataset. To verify the robustness of each synthetic dataset, three random samples of 10k data were extracted from each full-size dataset. These sampled 10k datasets were then used to conduct the evaluation experiments on IC15 using DB. The average F-measure for ST-10k, VISD-10k, UT-10k, and LBTS-10k were 86.24, 86.11, 86.39, and 86.78, respectively. The corresponding variances in F-measure were 0.017, 0.004, 0.019, and 0.015, indicating our LBTS datasets achieve consistently high performance across multiple samples. Fig. 11 displays some visual comparisons of baseline detectors with and without LBTS pretraining. Pretrained models effectively reduce detection errors and exhibit enhanced robustness in handling complex text instances.

To the best of our knowledge, this is the first report that highlights the performance discrepancy resulting from the use of synthetic datasets during the pretraining and fine-tuning stages. In our perspective, synthetic datasets play different roles when employed as independent training data or as pretraining data. When text detectors are solely trained on synthetic datasets and evaluated on real datasets, the performance of the text detector indicates the level of entangled “realism” between the synthetic dataset and real data to a certain extent. We believe that the realism of text encompasses multiple dimensions, such as text appearance, distribution, font, lighting conditions, and background image types. Both existing methods and our proposed LBTS approach impose constraints on the generated synthesis data in these dimensions to approximate the real-world domain. Those constraints are usually divided into several rules and steps based on prior knowledge. The “realism” we mentioned here denotes the degree of entangled “realism” achieved based on these constraints.

However, when synthetic datasets served as pretraining data, we hypothesize that dataset diversity becomes more crucial than “realism”. [68] is one extreme case that the models can be well pretrained without natural images. Synthetic data with greater diversity may enable convolutional layers to learn distinctive representations. These representations’ corresponding model weights are activated and reinforced if they are beneficial during the fine-tuning phase, thereby preventing the model from becoming trapped in local minima during gradient descent. We consider that the learning mechanism implemented in our LBTS engine introduces a



(a) EAST (from scratch) (b) EAST (Ours) (c) DB (from scratch) (d) DB (Ours)

Fig. 11. Visual comparisons of baseline detectors using pretraining. (a) Detection results of EAST trained from scratch. (b) Detection results of EAST pretrained with our LBTS dataset. (c) Detection results of DB trained from scratch. (d) Detection results of DB pretrained with our LBTS dataset. Zoom in for the best view.

TABLE IV

Quality Comparison Between Different Mixed Synthetic Datasets on ICDAR2013, ICDAR2015, ICDAR2017MLT Datasets Using EAST as the Baseline Detector. R: Recall, P: Precision, F: F-Score, Real: The Corresponding Training Set of the Evaluation Dataset

Train dataset	IC13			IC15			MLT17		
	R	P	F	R	P	F	R	P	F
VISD-5K + UT-5k	74.7	81.39	**77.9**	**65.86**	74.59	**69.96**	43.94	63.29	51.87
VISD-5k + LBTS-5k	74.16	79.07	76.53	60.66	72.16	65.92	42.51	62.53	50.61
UT-5k + LBTS-5k	69.41	**84.35**	76.15	58.02	**75.22**	65.51	40.65	**65.32**	50.11
VISD-3.3K + UT-3.3k + LBTS-3.3k	**75.98**	77.9	76.93	64.42	74.71	69.18	**45.22**	61.25	**52.03**
VISD-5k + UT-5k + Real	75.16	88.69	81.36	78.86	86.48	82.5	57.6	73.84	64.71
VISD-5k + LBTS-5k + Real	**76.44**	90	**82.67**	**80.65**	**87.01**	**83.71**	57.55	**74.47**	**64.93**
UT-5k + LBTS-5k + Real	75.43	**90.27**	82.19	80.07	86.98	83.38	**57.57**	73.94	64.74
VISD-3.3k + UT-3.3k + LBTS-3.3k + Real	75.53	88.35	81.44	80.16	86.63	83.27	56.45	74.4	64.19

greater degree of diversity compared to rule-based methods, resulting in our generated data performing better as pretraining data.

In addition, we created mixed synthetic datasets from different synthetic datasets to find out whether the data generated from different synthesis methods could play a complementary role during the training of the scene text detector. EAST was trained using the same configuration as the above experiment, and the evaluation results are summarized in Table IV. Without using real data, EAST achieved the best F-score when trained on VISD-5k + UT-5k, which was higher than the results obtained with VISD-10k or UT-10k individually. However, this synergetic effect disappeared when it served as pretraining data. The performance of EAST trained on the VISD-5k + UT-5k + Real is almost in the range of that achieved with UT-10k + Real to VISD-10k + Real, which cannot surpass the better performance between UT-10k + Real and VISD-10k + Real. A similar approximately linear relationship can also be found in other mixed datasets, including LBTS. On the other hand. when the mixed data serve as the pretraining data, we found that EAST trained with VISD-5k + LBTS-5k + Real or UT-5k + LBTS-5k + Real, performed better than that trained with synthetic data from a single source, such as VISD-10k + Real or UT-10k + Real.

Finally, we generated 100k synthetic images to test the scalability of our LBTS. We compared LBTS-100k with the full-size ST [1] and UT [4]. We trained EAST with 300,000 steps on different full-size datasets; the other configuration was the same as the above experiments. The evaluation results of EAST are presented in Table V. We observed that the performance of EAST improved when the number of generated datasets increased. Furthermore, EAST trained on LBTS-100k + Real achieved a competitive performance compared with that trained on ST-850k + Real and UT-670k + Real.

D. Ablation Study

In this section, we investigated the effectiveness of different settings of the proposed data-generation engine. The text location proposal network (TLPNet), geometry transformation module (GTM), color harmonization module (CHM), and postprocessing were the focus. The evaluation results of the EAST trained on the datasets generated by different configurations on the ICDAR2015 dataset are reported in Table VI.

- **Text Location Proposal Network** Given a background image, TLPNet aims to propose suitable regions for text embedding, which are usually relatively plain areas,

TABLE V

QUALITY COMPARISON BETWEEN DIFFERENT FULL-SIZE SYNTHETIC DATASETS ON ICDAR2013, ICDAR2015, ICDAR2017MLT DATASETS USING EAST AS THE BASELINE DETECTOR. R: RECALL, P: PRECISION, F: F-SCORE, REAL: THE CORRESPONDING TRAINING SET OF THE EVALUATION DATASET

Train dataset	IC13			IC15			MLT17		
	R	P	F	R	P	F	R	P	F
ST-850k	**71.78**	73.94	72.85	51.66	67.06	58.36	**40.97**	57.33	47.79
UT-670k	67.31	**85.6**	**75.36**	**53.06**	**81.69**	**64.33**	39.07	**66.73**	**49.28**
LBTS-100k	62.47	79.81	70.08	41.79	72.64	53.06	33.4	60.03	42.92
ST-850k + Real	75.34	88.71	81.48	79.54	85.95	82.62	57.52	74.37	64.87
UT-670k + Real	**76.8**	88.62	82.29	**82.19**	85.65	83.88	**57.8**	**74.32**	**65.02**
LBTS-100k + Real	76.26	**89.59**	**82.39**	81.95	**86.13**	**83.99**	57.67	74.22	64.91

TABLE VI

ABLATION STUDY: QUALITATIVE COMPARISON BETWEEN DIFFERENT CONFIGURATIONS OF OUR PROPOSED ENGINE ON ICDAR2015 DATASET USING EAST AS THE BASELINE DETECTOR

Train dataset	IC15		
	R	P	F
w/o TLPNet	35	56.66	43.27
w/o GTM	31.68	58.33	41.06
w/o CHM	37.41	67.39	48.11
w/o Postprocess	41.65	67.21	51.43
ALL	**42.95**	**68.14**	**52.69**
w/o TLPNet + Real	79.35	86.92	82.96
w/o GTM + Real	80.36	**87.15**	83.62
w/o CHM + Real	79.78	85.85	82.71
w/o Postprocess + Real	80.36	86.84	83.47
ALL + Real	**81.03**	86.66	**83.75**

as depicted in Fig. 10. To investigate its significance, we conducted an ablation study in which we replaced the output of TLPNet with an image, whose pixels value are all set to 1. This means the texts can appear at any location within the background image. The evaluation result, presented in Table VI emphasizes that TLPNet improves the quality of the generated synthetic data whether they served as the sole training data or the pretraining data.

- **Geometry Transformation Module** To assess the importance of the GTM, we replaced this module in our generation engine with a random transformation matrix generator. However, employing a completely random matrix generator is not advisable as it will heavily distort the text instances, resulting in extremely unrealistic results. For this reason, we adopted the random transformation matrix generator from a word-level SynthText engine [46] to reasonably transform the perspective of text instances, at least at the patch level. From Table VI, firstly, we observed that data generated with GTM serves as better independent training data and pretraining data for the text detector. This reveals the importance of our GTM function in the synthesis engine. Secondly, we noticed that when using synthetic data solely for training, there is a substantial performance gap between datasets generated w/o GTM and ALL. Nevertheless, this gap significantly diminishes when we incorporate real data for fine-tuning. This phenomenon further supports the conclusion drawn in the last subsection, highlighting that lower performance in the pretraining model does not necessarily lead to low performance in the fine-tuned model.
- **Color Harmonization Module** To evaluate the advantages of the CHM, the color-deciding process in our engine was replaced with that of the SynthText engine [1], where the text color is determined by referencing a learned dictionary based on the background's local statistic information. We can observe that the performance of EAST decreased when our CHM was missing.
- **Post-processing** To confirm the contribution of post-processing of our engine, we generated a dataset without applying post-processing and evaluated the quality of this dataset. Table VI implies that our post-processing techniques can enhance data diversity and improve the overall quality of generated data.

E. Discussion

Based on our comprehensive experimental results, although we cannot explicitly determine the specific type of data that benefits the training of text detectors, we can summarize several findings that prior studies have not addressed. First, we discovered that the performance of a text detector trained on both synthetic and real data is not strictly positively correlated with that trained only on synthetic data, even if the performance gap of the synthetic data is large. Second, the integration of different synthetic datasets generally improves the performance of the text detector; however, the extent of improvement differs based on the utilization of the mixed synthetic datasets. When using mixed synthetic data as independent training data, better performance can be achieved than that of datasets from a single source. However, when real data are involved in fine-tuning, the performance of the mixed data fails to surpass the best performance achieved by the single source dataset.

Our generation engine has several limitations. Firstly, the performance of TAANet, especially the GTM, is heavily influenced by the results of TLPNet. There exists a gap between the training data and inference data in TLPNet, where text-erased images usually have relatively large and flat areas with strong leading lines, such as the edges of signage or billboards, but the inference data are usually more diverse. A poor prediction of the text region often results in an unsatisfactory final output for human perception. This is because the GTM struggles to reasonably transform the perspective of text instances when the leading lines are missing in the background image. Secondly, in our proposed TAANet, the forward process is based on one text instance, thus, our method neglects to model the relationship between text instances. We opted to abandon text instances that were too close or that intersected with other texts, as it is uncommon for text to overlap in the real world. However, this trick usually leads to a disorganized layout of text instances and a reduction in generation efficiency. We believe that a unified training and generation structure may improve the generation results, and we expect future studies to successfully address these problems for learning-based scene-text image synthesis tasks.

VI. Conclusion

In this study, we first propose a new scene text dataset called DecompST, which can decompose real-world scene-text images into pure background images and pure text instances using text-erased images and stroke-level masks. Leveraging the DecompST dataset, we introduce a learning-based scene-text image synthesis engine, termed LBTS, which comprises a text location proposal network (TLPNet) and a text appearance adaptation network (TAANet). TLPNet is a segmentation network, capable of predicting suitable regions for text embedding. It is trained with the data pair of text-erased images and the mask of text regions, where text regions were extended from GT BBoxes based on appearance similarity and boundary information. TAANet consists of a geometry transformation module and a color harmonization module. These components can adaptively adjust the perspective and color of the synthetic text instance to ensure compatibility with the background. By combining our trained TLPNet and TAANet, we have developed a synthetic scene-text image generation engine and verified the effectiveness of our generated dataset using two popular baseline text detectors. Comprehensive experiments demonstrated the effectiveness of our proposed method in generating pretraining data for scene text detection.

References

[1] A. Gupta, A. Vedaldi, and A. Zisserman, “Synthetic data for text localisation in natural images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2315–2324.

[2] F. Zhan, S. Lu, and C. Xue, “Verisimilar image synthesis for accurate detection and recognition of texts in scenes,” in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 257–273.

[3] M. Liao, B. Song, S. Long, M. He, C. Yao, and X. Bai, “SynthText3D: Synthesizing scene text images from 3D virtual worlds,” *Sci. China Inf. Sci.*, vol. 63, no. 2, pp. 1–14, Feb. 2020.

[4] S. Long and C. Yao, “UnrealText: Synthesizing realistic scene text images from the unreal world,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Mar. 2020, pp. 5488–5497.

[5] X. Yang, D. He, D. Kifer, and C. L. Giles, “A learning-based text synthesis engine for scene text detection,” in *Proc. 30th Br. Mach. Vis. Conf.*, 2019, pp. 1–12.

[6] K. Sohn, X. Yan, and H. Lee, “Learning structured output representation using deep conditional generative models,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3483–3491.

[7] X. Chen and A. L. Yuille, “Detecting and reading text in natural scenes,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2004, pp. 366–373.

[8] L. Neumann and J. Matas, “A method for text localization and recognition in real-world images,” in *Proc. Asian Conf. Comput. Vis.*, Nov. 2011, pp. 770–783.

[9] A. Jamil, I. Siddiqi, F. Arif, and A. Raza, “Edge-based features for localization of artificial Urdu text in video images,” in *Proc. Int. Conf. Document Anal. Recognit.*, Sep. 2011, pp. 1120–1124.

[10] A. Mosleh, N. Bouguila, and A. B. Hamza, “Automatic inpainting scheme for video text detection and removal,” *IEEE Trans. Image Process.*, vol. 22, no. 11, pp. 4460–4472, Nov. 2013.

[11] W. Huang, Z. Lin, J. Yang, and J. Wang, “Text localization in natural images using stroke feature transform and text covariance descriptors,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1241–1248.

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

[13] W. Liu et al., “SSD: Single shot MultiBox detector,” in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2016, pp. 21–37.

[14] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, “Detecting text in natural image with connectionist text proposal network,” in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2016, pp. 56–72.

[15] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

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

[17] X. Zhou et al., “EAST: An efficient and accurate scene text detector,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2642–2651.

[18] C. Zhang et al., “Look more than once: An accurate detector for text of arbitrary shapes,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10544–10553.

[19] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai, “Multi-oriented text detection with fully convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4159–4167.

[20] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai, “Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes,” in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 71–88.

[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[22] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, “TextSnake: A flexible representation for detecting text of arbitrary shapes,” in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 19–35.

[23] Y. Li et al., “PSENet: Psoriasis Severity Evaluation Network,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, 2020, pp. 800–807.

[24] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, “Real-time scene text detection with differentiable binarization,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 7, 2020, pp. 11474–11481.

[25] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, “Character region awareness for text detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9357–9366.

[26] P. Dai, S. Yao, Z. Li, S. Zhang, and X. Cao, “ACE: Anchor-free corner evolution for real-time arbitrarily-oriented object detection,” *IEEE Trans. Image Process.*, vol. 31, pp. 4076–4089, 2022.

[27] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, “Color transfer between images,” *IEEE Comput. Graph. Appl.*, vol. 21, no. 4, pp. 34–41, Jul. 2001.

[28] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu, “Color harmonization,” in *Proc. ACM SIGGRAPH Papers (SIGGRAPH)*, 2006, p. 624.

[29] S. Xue, A. Agarwala, J. Dorsey, and H. Rushmeier, “Understanding and improving the realism of image composites,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–10, Aug. 2012.

[30] J. M. Di Martino, G. Facciolo, and E. Meinhardt-Llopis, “Poisson image editing,” *Image Process. Line*, vol. 6, pp. 300–325, Nov. 2016.

[31] J. Jia, J. Sun, C.-K. Tang, and H.-Y. Shum, “Drag-and-drop pasting,” *ACM Trans. Graph.*, vol. 25, no. 3, pp. 631–637, Jul. 2006.

[32] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, “Image melding,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–10, Aug. 2012.

[33] M. W. Tao, M. K. Johnson, and S. Paris, “Error-tolerant image compositing,” *Int. J. Comput. Vis.*, vol. 103, no. 2, pp. 178–189, Jun. 2013.

[34] K. Sunkavalli, M. K. Johnson, W. Matusik, and H. Pfister, “Multi-scale image harmonization,” *ACM Trans. Graph.*, vol. 29, no. 4, pp. 1–10, Jul. 2010.

[35] C.-H. Lin, E. Yumer, O. Wang, E. Shechtman, and S. Lucey, “ST-GAN: Spatial transformer generative adversarial networks for image compositing,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9455–9464.

[36] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.

[37] F. Zhan, H. Zhu, and S. Lu, "Spatial fusion GAN for image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3648–3657.

[38] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2242–2251.

[39] B.-C. Chen and A. Kae, "Toward realistic image compositing with adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8407–8416.

[40] Y.-H. Tsai, X. Shen, Z. Lin, K. Sunkavalli, X. Lu, and M.-H. Yang, "Deep image harmonization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2799–2807.

[41] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1510–1519.

[42] J. Ling, H. Xue, L. Song, R. Xie, and X. Gu, "Region-aware adaptive instance normalization for image harmonization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9357–9366.

[43] S. Qin, P. Ren, S. Kim, and R. Manduchi, "Robust and accurate text stroke segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 242–250.

[44] S. Bonechi, M. Bianchini, F. Scarselli, and P. Andreini, "Weak supervision for generating pixel–level annotations in scene text segmentation," *Pattern Recognit. Lett.*, vol. 138, pp. 1–7, Oct. 2020.

[45] O. Tursun, R. Zeng, S. Denman, S. Sivapalan, S. Sridharan, and C. Fookes, "MTRNet: A generic scene text eraser," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 39–44.

[46] Z. Tang, T. Miyazaki, Y. Sugaya, and S. Omachi, "Stroke-based scene text erasing using synthetic data for training," *IEEE Trans. Image Process.*, vol. 30, pp. 9306–9320, 2021.

[47] L. Wu et al., "Editing text in the wild," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 1500–1508.

[48] Q. Yang, J. Huang, and W. Lin, "SwapText: Image based texts transfer in scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14688–14697.

[49] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Proc. 21st Int. Conf. Pattern Recognit. (ICPR)*, Nov. 2012, pp. 3304–3308.

[50] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 1–20, Jan. 2016.

[51] M. Yim, Y. Kim, H.-C. Cho, and S. Park, "SynthTIGER: Synthetic text image GEneratoR towards better text recognition models," in *Proc. Int. Conf. Doc. Anal. Recognit.*, Sep. 2021, pp. 109–124.

[52] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 4, 2014, pp. 3581–3589.

[53] D. Karatzas et al., "ICDAR 2015 competition on robust reading," in *Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2015, pp. 1156–1160.

[54] N. Nayef et al., "ICDAR2019 robust reading challenge on multi-lingual scene text detection and recognition—RRC-MLT-2019," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 1582–1587.

[55] X. Xu, Z. Zhang, Z. Wang, B. Price, Z. Wang, and H. Shi, "Rethinking text segmentation: A novel dataset and a text-specific refinement approach," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 12040–12050.

[56] H.-S. Fang, J. Sun, R. Wang, M. Gou, Y.-L. Li, and C. Lu, "InstaBoost: Boosting instance segmentation via probability map guided copy-pasting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Nov. 2019, pp. 682–691.

[57] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.

[58] F. Zhan, S. Lu, C. Zhang, F. Ma, and X. Xie, "Adversarial Image Composition with Auxiliary Illumination," in *Proc. Asian Conf. Comput. Vis.*, 2020, pp. 234–250.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[60] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[61] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," 2014, *arXiv:1405.0312*.

[62] C. Liu, Y. Liu, L. Jin, S. Zhang, C. Luo, and Y. Wang, "EraseNet: End-to-end text removal in the wild," *IEEE Trans. Image Process.*, vol. 29, pp. 8760–8775, 2020.

[63] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015.

[64] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, Jun. 2018.

[65] D. Karatzas et al., "ICDAR 2013 robust reading competition," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Aug. 2013, pp. 1484–1493.

[66] N. Nayef et al., "ICDAR2017 robust reading challenge on multi-lingual scene text detection and script Identification–RRC-MLT," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1, Nov. 2017, pp. 1454–1459.

[67] C. K. Ch'ng and C. S. Chan, "Total-text: A comprehensive dataset for scene text detection and recognition," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1, Nov. 2017, pp. 935–942.

[68] H. Kataoka et al., "Pre-training without natural images," *Int. J. Comput. Vis.*, vol. 130, no. 4, pp. 990–1007, Apr. 2022.

Zhengmi Tang (Member, IEEE) received the B.E. degree from Xidian University, Shaanxi, China, in 2017, and the M.E. degree in cybernetics engineering from Hiroshima University, Japan, in 2020. He is currently pursuing the Ph.D. degree in communication engineering with the IIC-Laboratory, Tohoku University, Japan. His current research interests include computer vision, scene text detection, and data synthesis.

Tomo Miyazaki (Member, IEEE) received the B.E. degree from Yamagata University in 2006 and the Ph.D. degree from Tohoku University in 2011. From 2011 to 2012, he worked on geographic information systems with Hitachi Ltd. From 2013 to 2014, he was with Tohoku University as a Postdoctoral Researcher, where he has been an Assistant Professor since 2015. His current research interests include pattern recognition and image processing.

Shinichiro Omachi (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in information engineering from Tohoku University, Japan, in 1988, 1990, and 1993, respectively. He was an Assistant Professor with the Education Center for Information Processing, Tohoku University, from 1993 to 1996. Since 1996, he has been affiliated with the Graduate School of Engineering, Tohoku University, where he is currently a Professor. From 2000 to 2001, he was a Visiting Associate Professor with Brown University. His current research interests include pattern recognition, computer vision, image processing, image coding, and parallel processing. He is a member of the Institute of Electronics, Information and Communication Engineers, and the Information Processing Society of Japan. He received the IAPR/ICDAR Best Paper Award in 2007, the Best Paper Method Award of the 33rd Annual Conference of the GfKl in 2010, the ICFHR Best Paper Award in 2010, and the IEICE Best Paper Award in 2012. He served as the Vice Chair for the IEEE Sendai Section from 2020 to 2021. He served as the Editor-in-Chief for *IEICE Transactions on Information and Systems* from 2013 to 2015.