arXiv:2003.10608v6 [cs.CV] 18 Aug 2020

UnrealText: Synthesizing Realistic Scene Text Images from the Unreal World

Shangbang Long
Carnegie Mellon University
shangbal@cs.cmu.edu

Cong Yao
Megvii (Face++) Technology Inc.
yaocong2010@gmail.com

Lighting & Shadows | Suitable Text Region | Occlusion

Figure 1: Demonstration of the proposed **UnrealText** synthesis engine, which achieves photo-realistic lighting conditions, finds suitable text regions, and realizes natural occlusion (from *left* to *right*, zoomed-in views marked with green squares).

Abstract

Synthetic data has been a critical tool for training scene text detection and recognition models. On the one hand, synthetic word images have proven to be a successful substitute for real images in training scene text recognizers. On the other hand, however, scene text detectors still heavily rely on a large amount of manually annotated real-world images, which are expensive. In this paper, we introduce UnrealText, an efficient image synthesis method that renders realistic images via a 3D graphics engine. 3D synthetic engine provides realistic appearance by rendering scene and text as a whole, and allows for better text region proposals with access to precise scene information, e.g. normal and even object meshes. The comprehensive experiments verify its effectiveness on both scene text detection and recognition. We also generate a multilingual version for future research into multilingual scene text detection and recognition. Additionally, we re-annotate scene text recognition datasets in a case-sensitive way and include punctuation marks for more comprehensive evaluations. The code and the generated datasets are released at: https://jyouhou.github.io/UnrealText/.

1. Introduction

With the resurgence of neural networks, the past few years have witnessed significant progress in the field of scene text detection and recognition. However, these models are data-thirsty, and it is expensive and sometimes difficult, if not impossible, to collect enough data. Moreover, the various applications, from traffic sign reading in autonomous vehicles to instant translation, require a large amount of data specifically for each domain, further escalating this issue. Therefore, synthetic data and synthesis algorithms are important for scene text tasks. Furthermore, synthetic data can provide detailed annotations, such as character-level or even pixel-level ground truths that are rare for real images due to high cost.

Currently, there exist several synthesis algorithms [46, 10, 6, 50] that have proven beneficial. Especially, in scene text recognition, training on synthetic data [10, 6] alone has become a widely accepted standard practice. Some researchers that attempt training on both synthetic and real data only report marginal improvements [15, 20] on most datasets. Mixing synthetic and real data is only improving performance on a few difficult cases that are not yet well covered by existing synthetic datasets, such as seriously blurred or curved text. This is reasonable, since cropped text images have much simpler background, and synthetic data enjoys advantages in larger vocabulary size and diversity of backgrounds, fonts, and lighting conditions, as well as thousands of times more data samples.

On the contrary, however, scene text detection is still heavily dependent on real-world data. Synthetic data [6, 50] plays a less significant role, and only brings marginal improvements. Existing synthesizers for scene text detec-

tion follow the same paradigm. First, they analyze background images, e.g. by performing semantic segmentation and depth estimation using off-the-shelf models. Then, potential locations for text embedding are extracted from the segmented regions. Finally, text images (foregrounds) are blended into the background images, with perceptive transformation inferred from estimated depth. However, the analysis of background images with off-the-shelf models may be rough and imprecise. The errors further propagate to text proposal modules and result in text being embedded onto unsuitable locations. Moreover, the text embedding process is ignorant of the overall image conditions such as illumination and occlusions of the scene. These two factors make text instances outstanding from backgrounds, leading to a gap between synthetic and real images.

In this paper, we propose a synthetic engine that synthesizes scene text images from 3D virtual world. The proposed engine is based on the famous *Unreal Engine 4 (UE4)*, and is therefore named as *UnrealText*. Specifically, text instances are regarded as planar polygon meshes with text foregrounds loaded as texture. These meshes are placed in suitable positions in 3D world, and rendered together with the scene as a whole.

As shown in Fig. 1, the proposed synthesis engine, by its very nature, enjoys the following advantages over previous methods: (1) Text and scenes are rendered together, achieving realistic visual effects, e.g. illumination, occlusion, and perspective transformation. (2) The method has access to precise scene information, e.g. normal, depth, and object meshes, and therefore can generate better text region proposals. These aspects are crucial in training detectors.

To further exploit the potential of UnrealText, we design three key components: (1) A view finding algorithm that explores the virtual scenes and generates camera viewpoints to obtain more diverse and natural backgrounds. (2) An environment randomization module that changes the lighting conditions regularly, to simulate real-world variations. (3) A mesh-based text region generation method that finds suitable positions for text by probing the 3D meshes.

The contributions of this paper are summarized as follows: (1) We propose a brand-new scene text image synthesis engine that renders images from 3D world, which is entirely different from previous approaches that embed text on 2D background images, termed as **UnrealText**. The proposed engine achieves realistic rendering effects and high scalability. (2) With the proposed techniques, the synthesis engine improves the performance of detectors and recognizers significantly. (3) We also generate a large scale multilingual scene text dataset that will aid further research. (4) Additionally, we notice that many of the popular scene text recognition datasets are only annotated in an incomplete way, providing only case-insensitive word annotations. With such limited annotations, researchers are unable to carry out comprehensive evaluations, and tend to overestimate the progress of scene text recognition algorithms. To address this issue, we re-annotate these datasets to include both *upper-case* and *lower-case characters*, *digits*, *punctuation marks*, and *spaces* if there are any. We urge researchers to use the new annotations and evaluate in such a full-symbol mode for better understanding of the advantages and disadvantages of different algorithms.

2. Related Work

2.1. Synthetic Images

The synthesis of photo-realistic datasets has been a popular topic, since they provide detailed ground-truth annotations at multiple granularity, and cost less than manual annotations. In scene text detection and recognition, the use of synthetic datasets has become a standard practice. For scene text recognition, where images contain only one word, synthetic images are rendered through several steps [46, 10], including font rendering, coloring, homography transformation, and background blending. Later, GANs [5] are incorporated to maintain style consistency for implanted text [51], but it is only for single-word images. As a result of these progresses, synthetic data alone are enough to train state-of-the-art recognizers.

To train scene text detectors, SynthText [6] proposes to generate synthetic data by printing text on background images. It first analyzes images with off-the-shelf models, and search suitable text regions on semantically consistent regions. Text are implanted with perspective transformation based on estimated depth. To maintain semantic coherency, VISD [50] proposes to use semantic segmentation to filter out unreasonable surfaces such as human faces. They also adopt an adaptive coloring scheme to fit the text into the artistic style of backgrounds. However, without considering the scene as a whole, these methods fail to render text instances in a photo-realistic way, and text instances are too outstanding from backgrounds. So far, the training of detectors still relies heavily on real images.

Although GANs and other learning-based methods have also shown great potential in generating realistic images [48, 17, 12], the generation of scene text images still require a large amount of manually labeled data [51]. Furthermore, such data are sometimes not easy to collect, especially for cases such as low resource languages.

More recently, synthesizing images with 3D graphics engine has become popular in several fields, including human pose estimation [43], scene understanding/segmentation [28, 24, 33, 35, 37], and object detection [29, 42, 8]. However, these methods either consider simplistic cases, e.g. rendering 3D objects on top of static background images [29, 43] and randomly arranging scenes filled with objects [28, 24, 35, 8], or passively use off-the-

shelf 3D scenes without further changing it [33]. In contrast to these researches, our proposed synthesis engine implements active and regular interaction with 3D scenes, to generate realistic and diverse scene text images.

This paper is also a sequel to our previous attempt, the SynthText3D[16]. SynthText3D closely follows the designs of the SynthText method. While SynthText uses off-the-shelf computer vision models to estimate segmentation and depth maps for background images, SynthText3D uses the ground-truth segmentation and depth maps provided by the 3D engines. The rendering process of SynthText3D does not involve interactions with the 3D worlds, such as the object meshes. As a result, SynthText3D is faced with at least these two limitations: (1) the camera locations and rotations are labeled by human, limiting the scalability as well as diversity; (2) the generated text regions are limited to well defined regions that the camera is facing upfront, resulting in a unfavorable location bias.

2.2. Scene Text Detection and Recognition

Scene text detection and recognition, possibly as the most human-centric computer vision task, has been a popular research topic for many years [49, 21]. In scene text detection, there are mainly two branches of methodologies: Top-down methods that inherit the idea of region proposal networks from general object detectors that detect text instances as rotated rectangles and polygons [19, 53, 11, 52, 47]; Bottom-up approaches that predict local segments and local geometric attributes, and compose them into individual text instances [38, 22, 2, 40]. Despite significant improvements on individual datasets, those most widely used benchmark datasets are usually very small, with only around 500 to 1000 images in test sets, and are therefore prone to over-fitting. The generalization ability across different domains remains an open question, and is not studied yet. The reason lies in the very limited real data and that synthetic data are not effective enough. Therefore, one important motivation of our synthesis engine is to serve as a stepping stone towards general scene text detection.

Most scene text recognition models consist of CNN-based image feature extractors and attentional LSTM [9] or transformer [44]-based encoder-decoder to predict the textual content [3, 39, 15, 23]. Since the encoder-decoder module is a language model in essence, scene text recognizers have a high demand for training data with a large vocabulary, which is extremely difficult for real-world data. Besides, scene text recognizers work on image crops that have simple backgrounds, which are easy to synthesize. Therefore, synthetic data are necessary for scene text recognizers, and synthetic data alone are usually enough to achieve state-of-the-art performance. Moreover, since the recognition modules require a large amount of data, synthetic data are also necessary in training **end-to-end text spotting** systems [18, 7, 30].

3. Scene Text in 3D Virtual World

3.1. Overview

In this section, we give a detailed introduction to our scene text image synthesis engine, *UnrealText*, which is developed upon UE4 and the UnrealCV plugin [31]. The synthesis engine: (1) produces **photo-realistic** images, (2) is **efficient**, taking about only 1-1.5 second to render and generate a new scene text image and, (3) is **general and compatible** to off-the-shelf 3D scene models. As shown in Fig. 2, the pipeline mainly consists of a *Viewfinder* module (section 3.2), an *Environment Randomization* module (section 3.3), a *Text Region Generation* module (section 3.4), and a *Text Rendering* module (section 3.5).

Firstly, the viewfinder module explores around the 3D scene with the camera, generating camera viewpoints. Then, the environment lighting is randomly adjusted. Next, the text regions are proposed based on 2D scene information and refined with 3D mesh information in the graphics engine. After that, text foregrounds are generated with randomly sampled fonts, colors, and text content, and are loaded as planar meshes. Finally, we retrieve the RGB image and corresponding text locations as well as text content to make the synthetic dataset.

3.2. Viewfinder

The aim of the viewfinder module is to automatically determine a set of camera locations and rotations from the whole space of 3D scenes that are reasonable and non-trivial, getting rid of unsuitable viewpoints such as from inside object meshes (e.g. Fig. 3 bottom right).

Learning-based methods such as navigation and exploration algorithms may require extra training data and are not guaranteed to generalize to different 3D scenes. Therefore, we turn to rule-based methods and design a *physically-constrained 3D random walk* (Fig. 3 first row) equipped with *auxiliary camera anchors*.

3.2.1 Physically-Constrained 3D Random Walk

Starting from a valid location, the physically-constrained 3D random walk aims to find the next valid and non-trivial location. In contrast to being valid, locations are invalid if they are inside object meshes or far away from the scene boundary, for example. A non-trivial location should be not too close to the current location. Otherwise, the new viewpoint will be similar to the current one. The proposed 3D random walk uses ray-casting [36], which is constrained by physically, to inspect the physical environment to determine valid and non-trivial locations.

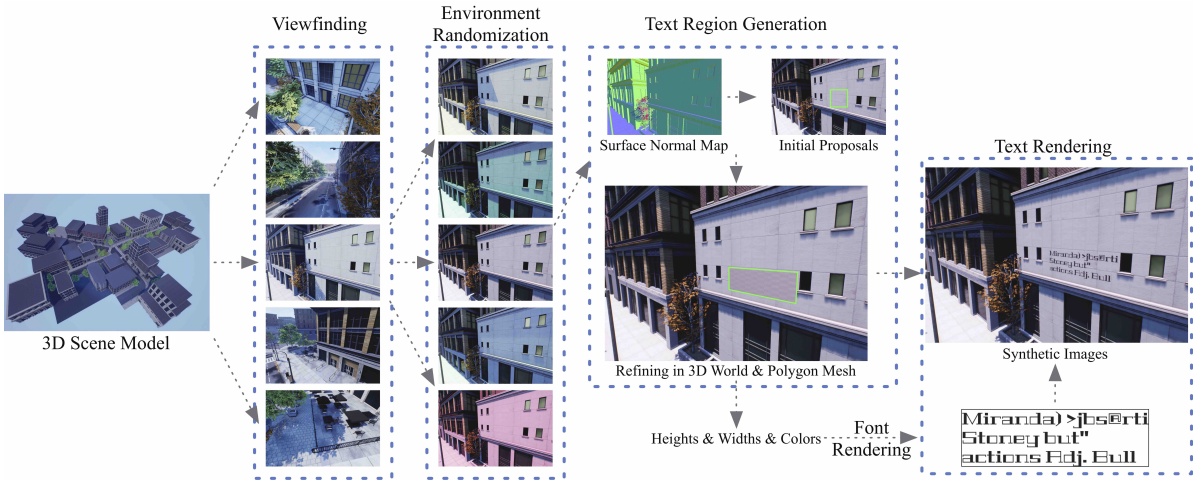


Figure 2: The pipeline of the proposed synthesis method. The arrows indicate the order. For simplicity, we only show one text region. From left to right: scene overview, diverse viewpoints, various lighting conditions (light color, intensity, shadows, etc.), text region generation and text rendering.

In each step, we first randomly change the pitch and yaw values of the camera rotation, making the camera pointing to a new direction. Then, we cast a ray from the camera location towards the direction of the viewpoint. The ray stops when it hits any object meshes or reaches a fixed maximum length. By design, the path from the current location to the stopping position is free of any barrier, i.e. not inside of any object meshes. Therefore, points along this ray path are all valid. Finally, we randomly sample one point between the $\frac{1}{3}$-th and $\frac{2}{3}$-th of this path, and set it as the new location of the camera, which is non-trivial. The proposed random walk algorithm can generate diverse camera viewpoints.

3.2.2 Auxiliary Camera Anchors

The proposed random walk algorithm, however, is inefficient in terms of exploration. Therefore, we manually select a set of N camera anchors across the 3D scenes as starting points. After every T steps, we reset the location of the camera to a randomly sampled camera anchor. We set $N = 150$-200 and $T = 100$. Note that the selection of camera anchors requires only little carefulness. We only need to ensure coverage over the space. It takes around 20 to 30 seconds for each scene, which is trivial and not a bottleneck of scalability. The manual but efficient selection of camera is compatible with the proposed random walk algorithm that generates diverse viewpoints.

3.3. Environment Randomization

To produce real-world variations such as lighting conditions, we randomly change the intensity, color, and direction of all light sources in the scene. In addition to illuminations, we also add fog conditions and randomly adjust its intensity. The environment randomization proves to increase the diversity of the generated images and results in stronger detector performance. The proposed randomization can also benefit sim-to-real domain adaptation [41].

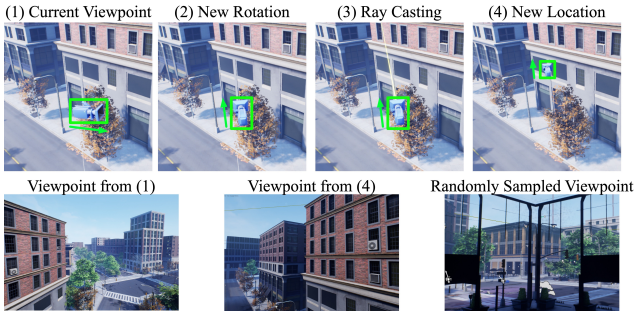


Figure 3: In the first row (1)-(4), we illustrate the *physically-constrained 3D random walk*. For better visualization, we use a camera object to represent the viewpoint (marked with green boxes and arrows). In the second row, we compare viewpoints from the proposed method with randomly sampled viewpoints.

3.4. Text Region Generation

In real-world, text instances are usually embedded on well-defined surfaces, e.g. traffic signs, to maintain good legibility. Previous works find suitable regions by using estimated scene information, such as gPb-UCM [1] in SynthText [6] or saliency map in VISD [50] for approximation. However, these methods are imprecise and often fail to find appropriate regions. Therefore, we propose to find text regions by probing around object meshes in 3D world. Since inspecting all object meshes is time-consuming, we propose a 2-staged pipeline: (1) We retrieve ground truth surface normal map to generate initial text region propos-

als; (2) Initial proposals are then projected to and refined in the 3D world using object meshes. Finally, we sample a subset from the refined proposals to render. To avoid occlusion among proposals, we project them back to screen space, and discard regions that overlap with each other one by one in a shuffled order until occlusion is eliminated.

3.4.1 Initial Proposals from Normal Maps

In computer graphics, normal values are unit vectors that are perpendicular to a surface. Therefore, when projected to 2D screen space, a region with similar normal values tends to be a well-defined region to embed text on. We find valid image regions by applying sliding windows of 64×64 pixels across the surface normal map, and retrieve those with *smooth* surface normal: the minimum cosine similarity value between any two pixels is larger than a threshold t. We set t to 0.95, which proves to produce reasonable results. We randomly sample at most 10 non-overlapping valid image regions to make the initial proposals. Making proposals from normal maps is an efficient way to find potential and visible regions.

3.4.2 Refining Proposals in 3D Worlds

As shown in Fig. 4, rectangular initial proposals in 2D screen space will be distorted when projected into 3D world. Thus, we need to first rectify the proposals in 3D world. We project the center point of the initial proposals into 3D space, and re-initialize *orthogonal* squares on the corresponding mesh surfaces around the center points: the horizontal sides are *orthogonal* to the gravity direction. The side lengths are set to the shortest sides of the quadrilaterals created by projecting the four corners of initial proposals into the 3D space. Then we enlarge the widths and heights along the horizontal and vertical sides alternatively. The expansion of one direction stops when the sides of that direction get off the surface[1], hit other meshes, or reach the preset maximum expansion ratio. The proposed refining algorithm works in 3D world space, and is able to produce natural homography transformation in 2D screen space.

3.5. Text Rendering

Generating Text Images: Given text regions as proposed and refined in section 3.4, the text generation module samples text content and renders text images with certain fonts and text colors. The numbers of lines and characters per line are determined by the font size and the size of refined proposals in 2D space to make sure the characters are not too small and ensure legibility. For a fairer comparison, we also use the same font set from Google Fonts [2] as SynthText does. We also use the same text corpus, Newsgroup20. The generated text images have zero alpha values on non-stroke pixels, and non zero for others.

Rendering Text in 3D World: We first perform triangulation for the refined proposals to generate planar triangular meshes that are closely attached to the underlying surface. Then we load the text images as texture onto the generated meshes. We also randomly sample the texture attributes, such as the ratio of diffuse and specular reflection.

[1] when the distances from the rectangular proposals' corners to the nearest point on the underlying surface mesh exceed certain threshold

[2] https://fonts.google.com/

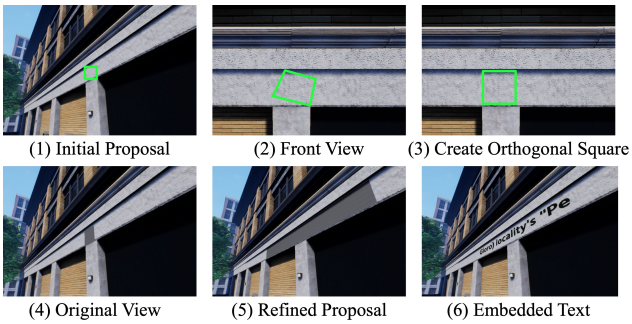


Figure 4: Illustration of the refinement of initial proposals. We draw green bounding boxes to represent proposals in 2D screen space, and use planar meshes to represent proposals in 3D space. (1) Initial proposals are made in 2D space. (2) When we project them into 3D world and inspect them from the front view, they are in distorted forms. (3) Based on the sizes of the distorted proposals and the positions of the center points, we re-initialize orthogonal squares on the same surfaces with horizontal sides orthogonal to the gravity direction. (5) Then we expand the squares. (6) Finally, we obtain text regions in 2D screen space with natural perspective distortion.

3.6. Implementation Details

The proposed synthesis engine is implemented based on UE4.22 and the UnrealCV plugin. On an ubuntu workstation with an 8-core Intel CPU, an NVIDIA GeForce RTX 2070 GPU, and 16G RAM, the synthesis speed is 0.7-1.5 seconds per image with a resolution of 1080×720, depending on the complexity of the scene model.

We collect 30 scene models from the official UE4 marketplace. The engine is used to generate $600K$ scene text images with English words. With the same configuration, we also generate a multilingual version, making it the largest multilingual scene text dataset.

4. Experiments on Scene Text Detection

4.1. Settings

We first verify the effectiveness of the proposed engine by training detectors on the synthesized images and evaluating them on real image datasets. We use a previous yet time-tested state-of-the-art model, EAST [53], which is fast and

accurate. EAST also forms the basis of several widely recognized end-to-end text spotting models [18, 7]. We adopt an opensource implementation[3]. In all experiments, models are trained on 4 GPU with a batch size of 56. During the evaluation, the test images are resized to match a short side length of 800 pixels. For each experiment setting, we report the mean performance in 5 independent trials.

Benchmark Datasets We use the following scene text detection datasets for evaluation: (1) *ICDAR 2013 Focused Scene Text* (IC13) [14] containing horizontal text with zoomed-in views. (2) *ICDAR 2015 Incidental Scene Text* (IC15) [13] consisting of images taken without carefulness with Google Glass. Images are blurred and text are small. (3) *MLT 2017* [27] for multilingual scene text detection, which is composed of scene text images of 9 languages. Note that the images in IC13 and MLT17 have varying resolutions. Therefore, it is necessary to resize them to the same level of resolutions before evaluation.

4.2. Experiments Results

Pure Synthetic Data We first train the EAST models on different synthetic datasets alone, to compare our method with previous ones in a direct and quantitative way. Note that UnrealText, SynthText3D, SynthText, and VISD have different numbers of images, so we also need to control the number of images used in experiments. Results are summarized in Tab. 1.

Firstly, we control the total number of images to 10K, which is also the full size of the smallest synthetic datasets, VISD and SynthText3D. We observe a considerable improvement on IC15 over previous state-of-the-art by $+0.9\%$ in F1-score, and significant improvements on IC13 ($+2.7\%$) and MLT 2017 ($+2.8\%$). Secondly, we also train models on the full set of SynthText and ours, since scalability is also an important factor for synthetic scene text images, especially when considering the demand to train recognizers. Extra training images further improve F1 scores on IC15, IC13, and MLT by $+2.6\%$, $+2.3\%$, and $+2.1\%$. Models trained with our UnrealText data outperform all other synthetic datasets. Besides, the subset of $10K$ images with our method even surpasses $800K$ SynthText images significantly on all datasets. The experiment results demonstrate the effectiveness of our proposed synthetic engine and datasets.

Complementary Synthetic Data One unique characteristic of the proposed UnrealText is that, the images are generated from 3D scene models, instead of real background images, resulting in potential domain gap due to different artistic styles. We conduct experiments by training on both UnrealText data ($5K$) and VISD ($5K$), as also shown in Tab. 1 (last row, marked with *italics*), which achieves better performance than other $10K$ synthetic datasets. The combination of UnrealText and VISD is also superior to the combination of SynthText3D and VISD. This result demonstrates that, our UnrealText is complementary to existing synthetic datasets that use real images as backgrounds. While UnrealText simulates photo-realistic effects, synthetic data with real background images can help adapt to real-world datasets.

Training Data	IC15	IC13	MLT 2017
SynthText 10K	46.3	60.8	38.9
VISD 10K (full)	64.3	74.8	51.4
SynthText3D 10K (full)	63.4	75.6	48.3
UnrealText 10K	**65.2**	**78.3**	**54.2**
SynthText 800K (full)	58.0	67.7	44.8
UnrealText 600K (full)	**67.8**	**80.6**	**56.3**
SynthText3D 5K + VISD 5K	*65.4*	*78.6*	*52.2*
UnrealText 5K + VISD 5K	***66.9***	***80.4***	***55.7***

Table 1: Detection results (F1-scores) of EAST models trained on different synthetic data.

Combining Synthetic and Real Data One important role of synthetic data is to serve as data for pretraining, and to further improve the performance on domain specific real datasets. We first pretrain the EAST models with different synthetic data, and then use domain data to finetune the models. The results are summarized in Tab. 2. On all domain-specific datasets, models pretrained with our synthetic dataset surpasses others by considerable margins, verifying the effectiveness of our synthesis method in the context of boosting performance on domain specific datasets.

Pretraining on Full Dataset As shown in the last rows of Tab. 2, when we pretrain the detector models with our full dataset, the performances are improved significantly, demonstrating the advantage of the scalability of our engine. Especially, The EAST model achieves an F1 score of 74.1 on MLT17, which is even better than recent state-of-the-art results, including 73.9 by CRAFT[2] and 73.1 by LOMO [52]. Although the margin is not great, it suffices to claim that the EAST model revives and reclaims state-of-the-art performance with the help of our synthetic dataset.

Results with Mask-RCNN As the EAST algorithm we use above is specifically designed for scene text and that the evaluation with F1 scores may not be comprehensive, we provide results with Mask-RCNN [**?**] which is a general object detector. We evaluate the models using the Average Precision (AP) metrics which are more comprehensive and less affected by the tricky choice of threshold values. We use the opensource implementation Detectron2 [4]. The rotated bounding boxes of text instances are used as the mask annotations. We select a default Mask-RCNN configuration with ResNet-50+FPN as the backbone and train the model

[3] https://github.com/argman/EAST

[4] https://github.com/facebookresearch/detectron2

Evaluation on ICDAR 2015

Training Data	P	R	F1
IC15	84.6	78.5	81.4
IC15 + SynthText 10K	85.6	79.5	82.4
IC15 + VISD 10K	86.3	80.0	83.1
IC15 + SynthText3D 10K	86.6	80.4	83.4
IC15 + UnrealText 10K	**86.9**	**81.0**	**83.8**
IC15 + UnrealText 600K	*88.5*	*80.8*	*84.5*

Evaluation on ICDAR 2013

Training Data	P	R	F1
IC13	82.6	70.0	75.8
IC13 + SynthText 10K	85.3	72.4	78.3
IC13 + VISD 10K	85.9	73.1	79.0
IC13 + SynthText3D 10K	86.4	73.0	79.1
IC13 + UnrealText 10K	**88.5**	**74.7**	**81.0**
IC13 + UnrealText 600K	*92.3*	*73.4*	*81.8*

Evaluation on MLT 2017

Training Data	P	R	F1
MLT 2017	72.9	67.4	70.1
MLT 2017 + SynthText 10K	73.1	67.7	70.3
MLT 2017 + VISD 10K	73.3	67.9	70.5
MLT 2017 + SynthText3D 10K	73.8	67.6	70.6
MLT 2017 + UnrealText 10K	**74.6**	**68.7**	**71.6**
MLT 2017 + UnrealText 600K	*82.2*	*67.4*	*74.1*

Table 2: Detection performances of EAST models pre-trained on synthetic and then finetuned on real datasets.

for 1x schedule long. All the hyperparameters are set to default values. The results are summarized in Tab. 3. We notice that the two synthetic datasets with natural images as backgrounds, i.e. SynthText and VISD, result in similar performances. SynthText3D and our UnrealText are significantly better than them. UnrealText is further a significant improvement over SynthText3D. When we combine UnrealText and SynthText, the two highly scalable engines, the performances are even better.

Training Data	IC15	IC13	MLT 2017
SynthText 10K	13.6/13.4	41.1/41.7	19.6/17.5
VISD 10K (full)	13.8/13.5	37.6/37.6	18.1/18.4
SynthText3D 10K (full)	19.4/19.8	37.9/39.3	22.8/22.5
UnrealText 10K	**25.1/23.7**	**50.1/49.2**	**24/23.6**
SynthText 800K (full)	19.6/20.3	47.5/48.2	24.2/24.8
UnrealText 600K (full)	**26.2/25**	**51.5/52.2**	**27.8/27.3**
UnrealText full + SynthText full	***27.7/27.5***	***62.4/63.7***	***32.4/32.1***

Table 3: Detection results (Box-AP/Mask-AP) of Mask-RCNN models trained on different synthetic data.

4.3. Module Level Ablation Analysis

One reasonable concern about synthesizing from 3D virtual scenes lies in the scene diversity. In this section, we address the importance of the proposed view finding module and the environment randomization module in increasing the diversity of synthetic images.

Ablating Viewfinder Module We derive two baselines from the proposed viewfinder module: (1) *Random Viewpoint + Manual Anchor* that randomly samples camera locations and rotations from the norm-ball spaces centered around auxiliary camera anchors. (2) *Random Viewpoint Only* that randomly samples camera locations and rotations from the whole scene space, without checking their quality. For experiments, we fix the number of scenes to 10 to control scene diversity and generate different numbers of images, and compare their performance curve. By fixing the number of scenes, we compare how well different view finding methods can exploit the scenes.

Ablating Environment Randomization We remove the environment randomization module, and keep the scene models unchanged during synthesis. For experiments, we fix the total number of images to $10K$ and use different number of scenes. In this way, we can compare the diversity of images generated with different methods.

We train the EAST models with different numbers of images or scenes, evaluate them on the 3 real datasets, and compute the arithmetic mean of the F1-scores. As shown in Fig. 5 (a), we observe that the proposed combination, i.e. *Random Walk + Manual Anchor*, achieves significantly higher F1-scores consistently for different numbers of images. Especially, larger sizes of training sets result in greater performance gaps. We also inspect the images generated with these methods respectively. When starting from the same anchor point, the proposed random walk can generate more diverse viewpoints and can traverse much larger area. In contrast, the *Random Viewpoint + Manual Anchor* method degenerates either into random rotation only when we set a small norm ball size for random location, or into *Random Viewpoint Only* when we set a large norm ball size. As a result, the *Random Viewpoint + Manual Anchor* method requires careful manual selection of anchors, and we also need to manually tune the norm ball sizes for different scenes, which restricts the scalability of the synthesis engine. Meanwhile, our proposed random walk based method is more flexible and robust to the selection of manual anchors. As for the *Random Viewpoint Only* method, a large proportion of generated viewpoints are invalid, e.g. inside other object meshes, which is out-of-distribution for real images. This explains why it results in the worst performances.

From Fig. 5 (b), the major observation is that environment randomization module improves performances over different scene numbers consistently. Besides, the improvement is more significant as we use fewer scenes. Therefore, we can draw a conclusion that, the environment randomization helps increase image diversity and at the same time, can reduce the number of scenes needed. Furthermore, the

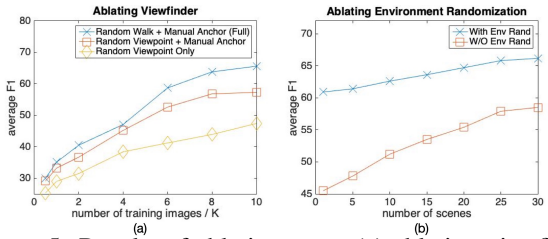


Figure 5: Results of ablation tests: (a) ablating viewfinder module; (b) ablating environment randomization module.

random lighting conditions realize different real-world variations, which we also attribute as a key factor.

5. Experiments on Scene Text Recognition

In addition to the superior performances in training scene text detection models, we also verify its effectiveness in the task of scene text recognition.

5.1. Recognizing Latin Scene Text

5.1.1 Settings

Model We select a widely accepted baseline method, ASTER [39], and adopt the implementation[5] that ranks top-1 on the ICDAR 2019 ArT competition on curved scene text recognition (Latin) by [20]. The models are trained with a batch size of 512. A total of 95 symbols are recognized, including an End-of-Sentence mark, 52 case sensitive alphabets, 10 digits, and 32 printable punctuation symbols.

Training Datasets From the $600K$ English synthetic images, we obtain a total number of $12M$ word-level image regions to make our training dataset. Also note that, our synthetic dataset provide character level annotations, which will be useful in some recognition algorithms.

Evaluation Datasets We evaluate models trained on different synthetic datasets on several widely used real image datasets: *IIIT* [25], *SVT* [45], *ICDAR 2015 (IC15)* [13], *SVTP* [32], *CUTE* [34], and *Total-Text*[4].

Some of these datasets, however, have *incomplete* annotations, including IIIT, SVT, SVTP, CUTE. While the word images in these datasets contain punctuation symbols, digits, upper-case and lower-case characters, the aforementioned datasets, in their current forms, only provide case-insensitive annotations and ignore all punctuation symbols. In order for more comprehensive evaluation of scene text recognition, we re-annotate these 4 datasets in a case-sensitive way and also include punctuation symbols. We also release the new annotations and we believe that they will become better benchmarks for scene text recognition in the future.

[5] https://github.com/Jyouhou/ICDAR2019-ArT-Recognition-Alchemy

5.1.2 Experiment Results

Experiment results are summarized in Tab. 7. First, we compare our method with previous synthetic datasets. We have to limit the size of training datasets to $1M$ since VISD only publishes $1M$ word images. Our synthetic data achieves consistent improvements on all datasets. Especially, it surpasses other synthetic datasets by a considerable margin on datasets with diverse text styles and complex backgrounds such as SVTP ($+2.4\%$). The experiments verify the effectiveness of our synthesis method in scene text recognition especially in the complex cases.

Since small scale experiments are not very helpful in how researchers should utilize these datasets, we further train models on combinations of Synth90K, SynthText, and ours. We first limit the total number of training images to $9M$. When we train on a combination of all 3 synthetic datasets, with $3M$ each, the model performs better than the model trained on $4.5M \times 2$ datasets only. We further observe that training on $3M \times 3$ synthetic datasets is comparable to training on the whole Synth90K and SynthText, while using much fewer training data. This result suggests that the best practice is to combine the proposed synthetic dataset with previous ones.

5.2. Recognizing Multilingual Scene Text

5.2.1 Settings

Although MLT 2017 has been widely used as a benchmark for detection, the task of recognizing multilingual scene text still remains largely untouched, mainly due to lack of a proper training dataset. To pave the way for future research, we also generate a multilingual version with $600K$ images containing 10 languages as included in MLT 2019 [26]: *Arabic*, *Bangla*, *Chinese*, *English*, *French*, *German*, *Hindi*, *Italian*, *Japanese*, and *Korean*. Text contents are sampled from corpus extracted from the Wikimedia dump[6].

Model We use the same model and implementation as Section 5.1, except that the symbols to recognize are expanded to all characters that appear in the generated dataset.

Training and Evaluation Data We crop from the proposed multilingual dataset. We discard images with widths shorter than 32 pixels as they are too blurry, and obtain $4.1M$ word images in total. We compare with the multilingual version of SynthText provided by *MLT 2019* competition that contains a total number $1.2M$ images. For evaluation, we randomly split 1500 images for each language (including *symbols* and *mixed*) from the training set of MLT 2019. The rest of the training set is used for training.

[6] https://dumps.wikimedia.org

Training Data	Latin	Arabic	Bangla	Chinese	Hindi	Japanese	Korean	Symbols	Mixed	Overall
ST (1.2M)	34.6	**50.5**	**17.7**	43.9	15.7	21.2	**55.7**	**44.7**	9.8	34.9
UnrealText (1.2M)	**42.2**	50.3	16.5	**44.8**	**30.3**	**21.7**	54.6	16.7	**25.0**	**36.5**
UnrealText (full, 4.1M)	*44.3*	*51.1*	*19.7*	*47.9*	*33.1*	*24.2*	*57.3*	*25.6*	*31.4*	*39.5*
MLT19-train (90K)	64.3	47.2	46.9	11.9	46.9	23.3	39.1	35.9	3.6	45.7
MLT19-train (90K) + ST (1.2M)	63.8	62.0	48.9	**50.7**	47.7	33.9	**64.5**	**45.5**	10.3	54.7
MLT19-train (90K) + UnrealText (1.2M)	**67.8**	**63.0**	**53.7**	47.7	**64.0**	**35.7**	62.9	44.3	**26.3**	**57.9**

Table 4: Multilingual scene text recognition results (word level accuracy). *Latin* aggregates *English*, *French*, *German*, and *Italian*, as they are all marked as *Latin* in the MLT dataset.

Training Data	IIIT	SVT	IC15	SVTP	CUTE	Total
90K [10] (1M)	51.6	39.2	35.7	37.2	30.9	30.5
ST [6] (1M)	53.5	30.3	38.4	29.5	31.2	31.1
VISD [50] (1M)	53.9	37.1	37.1	36.3	30.5	30.9
UnrealText (1M)	**54.8**	**40.3**	**39.1**	**39.6**	**31.6**	**32.1**
ST+90K($4.5M \times 2$)	80.5	70.1	58.4	60.0	63.9	43.2
ST+90K+UnrealText($3M \times 3$)	**81.6**	**71.9**	61.8	61.7	**67.7**	**45.7**
ST+90K($16M$)	81.2	71.2	**62.0**	**62.3**	65.1	44.7

Table 5: Results on English datasets (word level accuracy).

5.2.2 Experiment Results

Experiment results are shown in Tab. 4. When we only use synthetic data and control the number of images to $1.2M$, ours result in a considerable improvement of 1.6% in overall accuracy, and significant improvements on some scripts, e.g. *Latin* ($+7.6\%$) and *Mixed* ($+21.6\%$). Using the whole training set of $4.1M$ images further improves overall accuracy to 39.5%. When we train models on combinations of synthetic data and our training split of MLT19, as shown in the bottom of Tab. 4, we can still observe a considerable margin of our method over SynthText by 3.2% in overall accuracy. The experiment results demonstrate that our method is also superior in multilingual scene text recognition, and we believe this result will become a stepping stone to further research.

6. Limitation and Future Work

There are several aspects that are worth diving deeper into: (1) Overall, the engine is based on rules and human-selected parameters. The automation of the selection and search for these parameters can save human efforts and help adapt to different scenarios. (2) While rendering small text can help training detectors, the low image quality of the small text makes recognizers harder to train and harms the performance. Designing a method to mark the illegible ones as *difficult* and excluding them from loss calculation may help mitigate this problem. (3) For multilingual scene text, scripts except *Latin* have much fewer available fonts that we have easy access to. To improve performance on more languages, researchers may consider learning-based methods to transfer *Latin* fonts to other scripts.

7. Conclusion

In this paper, we introduce a scene text image synthesis engine that renders images with 3D graphics engines, where text instances and scenes are rendered as a whole. In experiments, we verify the effectiveness of the proposed engine in both scene text detection and recognition models. We also study key components of the proposed engine. We believe our work will be a solid stepping stone towards better synthesis algorithms.

Acknowledgement

This research was supported by National Key R&D Program of China (No. 2017YFA0700800).

References

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[2] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9365–9374, 2019.

[3] Zhanzhan Cheng, Xuyang Liu, Fan Bai, Yi Niu, Shiliang Pu, and Shuigeng Zhou. Arbitrarily-oriented text recognition. *CVPR2018*, 2017.

[4] Chee Kheng Ch'ng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *Proc. ICDAR*, volume 1, pages 935–942, 2017.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NIPS*, pages 2672–2680, 2014.

[6] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proc. CVPR*, pages 2315–2324, 2016.

[7] Tong He, Zhi Tian, Weilin Huang, Chunhua Shen, Yu Qiao, and Changming Sun. An end-to-end textspotter with explicit alignment and attention. In *Proc. CVPR*, pages 5020–5029, 2018.

[8] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Martina Marek, and Martin Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object instance detection. *CoRR*, abs/1902.09967, 2019.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.

[11] Yingying Jiang, Xiangyu Zhu, Xiaobing Wang, Shuli Yang, Wei Li, Hua Wang, Pei Fu, and Zhenbo Luo. R2cnn: rotational region cnn for orientation robust scene text detection. *arXiv preprint arXiv:1706.09579*, 2017.

[12] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. *arXiv preprint arXiv:1904.11621*, 2019.

[13] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.

[14] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere de las Heras. Icdar 2013 robust reading competition. In *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1484–1493. IEEE, 2013.

[15] Hui Li, Peng Wang, Chunhua Shen, and Guyu Zhang. Show, attend and read: A simple and strong baseline for irregular text recognition. *AAAI*, 2019.

[16] Minghui Liao, Boyu Song, Shangbang Long, Minghang He, Cong Yao, and Xiang Bai. Synthtext3d: synthesizing scene text images from 3d virtual worlds. *Science China Information Sciences*, 63(2):120105, 2020.

[17] Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. St-gan: Spatial transformer generative adversarial networks for image compositing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9455–9464, 2018.

[18] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. Fots: Fast oriented text spotting with a unified network. *Proc. CVPR*, 2018.

[19] Yuliang Liu and Lianwen Jin. Deep matching prior network: Toward tighter multi-oriented text detection. In *Proc. CVPR*, 2017.

[20] Shangbang Long, Yushuo Guan, Bingxuan Wang, Kaigui Bian, and Cong Yao. Alchemy: Techniques for rectification based irregular scene text recognition. *arXiv preprint arXiv:1908.11834*, 2019.

[21] Shangbang Long, Xin He, and Cong Yao. Scene text detection and recognition: The deep learning era. *arXiv preprint arXiv:1811.04256*, 2018.

[22] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *Proc. ECCV*, 2018.

[23] Pengyuan Lyu, Zhicheng Yang, Xinhang Leng, Xiaojun Wu, Ruiyu Li, and Xiaoyong Shen. 2d attentional irregular scene text recognizer. *arXiv preprint arXiv:1906.05708*, 2019.

[24] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. Scenenet RGB-D: 5m photorealistic images of synthetic indoor trajectories with ground truth. *CoRR*, abs/1612.05079, 2016.

[25] Anand Mishra, Karteek Alahari, and CV Jawahar. Scene text recognition using higher order language priors. In *BMVC-British Machine Vision Conference*. BMVA, 2012.

[26] Nibal Nayef, Yash Patel, Michal Busta, Pinaki Nath Chowdhury, Dimosthenis Karatzas, Wafa Khlif, Jiri Matas, Umapada Pal, Jean-Christophe Burie, Cheng-lin Liu, et al. Icdar2019 robust reading challenge on multi-lingual scene text detection and recognition–rrc-mlt-2019. *arXiv preprint arXiv:1907.00945*, 2019.

[27] Nibal Nayef, Fei Yin, Imen Bizid, Hyunsoo Choi, Yuan Feng, Dimosthenis Karatzas, Zhenbo Luo, Umapada Pal, Christophe Rigaud, Joseph Chazalon, et al. Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-rrc-mlt. In *Proc. ICDAR*, volume 1, pages 1454–1459. IEEE, 2017.

[28] Jeremie Papon and Markus Schoeler. Semantic pose using deep networks trained on synthetic rgb-d. In *Proc. ICCV*, pages 774–782, 2015.

[29] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models. In *Proc. ICCV*, pages 1278–1286, 2015.

[30] Siyang Qin, Alessandro Bissacco, Michalis Raptis, Yasuhisa Fujii, and Ying Xiao. Towards unconstrained end-to-end text spotting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4704–4714, 2019.

[31] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. In *Proc. ECCV*, pages 909–916, 2016.

[32] Trung Quy Phan, Palaiahnakote Shivakumara, Shangxuan Tian, and Chew Lim Tan. Recognizing text with perspective distortion in natural scenes. In *Proc. ICCV*, pages 569–576, 2013.

[33] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.

[34] Anhar Risnumawan, Palaiahankote Shivakumara, Chee Seng Chan, and Chew Lim Tan. A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications*, 41(18):8027–8048, 2014.

[35] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proc. CVPR*, pages 3234–3243, 2016.

[36] Scott D Roth. Ray casting for modeling solids. *Computer Graphics & Image Processing*, 18(2):109–144, 1982.

[37] Fatemeh Sadat Saleh, Mohammad Sadegh Aliakbarian, Mathieu Salzmann, Lars Petersson, and Jose M Alvarez. Effective use of synthetic data for urban scene semantic segmentation. In *Proc. ECCV*, pages 86–103, 2018.

[38] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting oriented text in natural images by linking segments. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[39] Baoguang Shi, Mingkun Yang, XingGang Wang, Pengyuan Lyu, Xiang Bai, and Cong Yao. Aster: An attentional scene text recognizer with flexible rectification. *IEEE transactions on pattern analysis and machine intelligence*, 31(11):855–868, 2018.

[40] Zhuotao Tian, Michelle Shu, Pengyuan Lyu, Ruiyu Li, Chao Zhou, Xiaoyong Shen, and Jiaya Jia. Learning shape-aware embedding for scene text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4234–4243, 2019.

[41] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[42] Jonathan Tremblay, Thang To, and Stan Birchfield. Falling things: A synthetic dataset for 3d object detection and pose estimation. In *Proc. CVPR Workshops*, pages 2038–2041, 2018.

[43] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Proc. CVPR*, pages 109–117, 2017.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. NIPS*, pages 5998–6008, 2017.

[45] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *2011 IEEE International Conference on Computer Vision (ICCV),*, pages 1457–1464. IEEE, 2011.

[46] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *2012 21st International Conference on Pattern Recognition (ICPR)*, pages 3304–3308. IEEE, 2012.

[47] Xiaobing Wang, Yingying Jiang, Zhenbo Luo, Cheng-Lin Liu, Hyunsoo Choi, and Sungjin Kim. Arbitrary shape scene text detection with adaptive text region representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6449–6458, 2019.

[48] Xinlong Wang, Zhipeng Man, Mingyu You, and Chunhua Shen. Adversarial generation of training examples: Applications to moving vehicle license plate recognition. *arXiv preprint arXiv:1707.03124*, 2017.

[49] Qixiang Ye and David Doermann. Text detection and recognition in imagery: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 37(7):1480–1500, 2015.

[50] Fangneng Zhan, Shijian Lu, and Chuhui Xue. Verisimilar image synthesis for accurate detection and recognition of texts in scenes. In *Proc. ECCV*, 2018.

[51] Fangneng Zhan, Hongyuan Zhu, and Shijian Lu. Spatial fusion gan for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3653–3662, 2019.

[52] Chengquan Zhang, Borong Liang, Zuming Huang, Mengyi En, Junyu Han, Errui Ding, and Xinghao Ding. Look more than once: An accurate detector for text of arbitrary shapes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[53] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. EAST: An efficient and accurate scene text detector. In *Proc. CVPR*, 2017.

A. Scene Models

In this work, we use a total number of 30 scene models which are all obtained from the Internet. However, most of these models are not free. Therefore, we are not allowed to share the models themselves. Instead, we list the models we use and their links in Tab. 6.

B. New Annotations for Scene Text Recognition Datasets

During the experiments of scene text recognition for English scripts, we notice that among the most widely used benchmark datasets, several have incomplete annotations. They are IIIT5K, SVT, SVTP, and CUTE-80. The annotations of these datasets are case-insensitive, and ignore punctuation marks.

The common practice for recent scene text recognition research is to convert both prediction and ground-truth text strings to lower-case and then compare them. This means that the current evaluation is flawed. It ignores letter case and punctuation marks which are crucial to the understanding of the text contents. Besides, evaluating on a much smaller vocabulary set results in over-optimism of the performance of the recognition models.

To aid further research, we use the Amazon mechanical Turk (AMT) to re-annotate the aforementioned 4 datasets, which amount to 6837 word images in total. Each word image is annotated by 3 workers, and we manually check and correct images where the 3 annotations differ. The annotated datasets are released via GitHub at https://github.com/Jyouhou/Case-Sensitive-Scene-Text-Recognition-Datasets.

B.1 Samples

We select some samples from the 4 datasets to demonstrate the new annotations in Fig. 6.

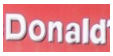
Dataset	Sample Image	Original Annotation	New Annotation
CUTE80		TEAM	Team
IIIT5K		15	15%.
SVT		DONALD	Donald'
SVTP		MARLBORO	Marlboro

Figure 6: Examples of the new annotations.

B.2 Benchmark Performances

As we are encouraging case-sensitive (also with punctuation marks) evaluation for scene text recognition, we would like to provide benchmark performances on those widely used datasets. We evaluate two implementations of the ASTER models, by Long *et al.*[7] and Baek *et al*[8] respectively. Results are summarized in Tab. 7.

The two benchmark implementations perform comparably, with Baek's better on straight text and Long's better at curved text. Compared with evaluation with *lower case + digits*, the performance drops considerably for both models when we evaluate with all symbols. These results indicate that it may still be a challenge to recognize a larger vocabulary, and is worth further research.

[7] https://github.com/Jyouhou/ICDAR2019-ArT-Recognition-Alchemy

[8] https://github.com/clovaai/deep-text-recognition-benchmark

Scene Name	Link
Urban City	https://www.unrealengine.com/marketplace/en-US/product/urban-city
Medieval Village	https://www.unrealengine.com/marketplace/en-US/product/medieval-village
Loft	https://ue4arch.com/shop/complete-projects/archviz/loft/
Desert Town	https://www.unrealengine.com/marketplace/en-US/product/desert-town
Archinterior 1	https://www.unrealengine.com/marketplace/en-US/product/archinteriors-vol-2-scene-01
Desert Gas Station	https://www.unrealengine.com/marketplace/en-US/product/desert-gas-station
Modular School	https://www.unrealengine.com/marketplace/en-US/product/modular-school-pack
Factory District	https://www.unrealengine.com/marketplace/en-US/product/factory-district
Abandoned Factory	https://www.unrealengine.com/marketplace/en-US/product/modular-abandoned-factory
Buddhist	https://www.unrealengine.com/marketplace/en-US/product/buddhist-monastery-environment
Castle Fortress	https://www.unrealengine.com/marketplace/en-US/product/castle-fortress
Desert Ruin	https://www.unrealengine.com/marketplace/en-US/product/modular-desert-ruins
HALArchviz	https://www.unrealengine.com/marketplace/en-US/product/hal-archviz-toolkit-v1
Hospital	https://www.unrealengine.com/marketplace/en-US/product/modular-sci-fi-hospital
HQ House	https://www.unrealengine.com/marketplace/en-US/product/hq-residential-house
Industrial City	https://www.unrealengine.com/marketplace/en-US/product/industrial-city
Archinterior 2	https://www.unrealengine.com/marketplace/en-US/product/archinteriors-vol-4-scene-02
Office	https://www.unrealengine.com/marketplace/en-US/product/retro-office-environment
Meeting Room	https://drive.google.com/file/d/0B_mjKk7NOcnEUWZuRDVFQ09STE0/view
Old Village	https://www.unrealengine.com/marketplace/en-US/product/old-village
Modular Building	https://www.unrealengine.com/marketplace/en-US/product/modular-building-set
Modular Home	https://www.unrealengine.com/marketplace/en-US/product/supergenius-modular-home
Dungeon	https://www.unrealengine.com/marketplace/en-US/product/top-down-multistory-dungeons
Old Town	https://www.unrealengine.com/marketplace/en-US/product/old-town
Root Cellar	https://www.unrealengine.com/marketplace/en-US/product/root-cellar
Victorian	https://www.unrealengine.com/marketplace/en-US/product/victorian-street
Spaceship	https://www.unrealengine.com/marketplace/en-US/product/spaceship-interior-environment-set
Top-Down City	https://www.unrealengine.com/marketplace/en-US/product/top-down-city
Scene Name	https://www.unrealengine.com/marketplace/en-US/product/urban-city
Utopian City	https://www.unrealengine.com/marketplace/en-US/product/utopian-city

Table 6: The list of 3D scene models used in this work.

Implementation	Case	IIIT	SVT	IC13	IC15	SVTP	CUTE80	Total
Long *et al.*	All	81.2	71.2	86.9	62.0	62.3	65.1	44.7
Baek *et al*	All	81.5	71.7	88.9	62.1	62.6	64.9	41.5
Long *et al.*	lower case + digits	89.5	84.1	89.9	68.8	73.5	76.3	58.2
Baek *et al*	lower case + digits	86.5	83.5	93.0	70.3	75.1	68.4	46.0

Table 7: Results on English datasets (word level accuracy). *All* indicates that the evaluation considers lower case characters, upper case characters, numerical digits, and punctuation marks.