SynthText3D: Synthesizing Scene Text Images from 3D Virtual Worlds

Minghui Liao*, Boyu Song*, Minghang He, Shangbang Long, Cong Yao, Xiang Bai
Huazhong University of Science and Technology, Peking University, MEGVII (Face++) Inc.
mhliao@hust.edu.cn, boyusong@pku.edu.cn, minghanghe09@hust.edu.cn
shangbang.long@pku.edu.cn, yaocong2010@gmail.com, xbai@hust.edu.cn

Abstract

With the development of deep neural networks, the demand for a significant amount of annotated training data becomes the performance bottlenecks in many fields of research and applications. Image synthesis can generate annotated images automatically and freely, which gains increasing attention recently. In this paper, we propose to synthesize scene text images from the 3D virtual worlds, where the precise descriptions of scenes, editable illumination/visibility, and realistic physics are provided. Different from the previous methods which paste the rendered text on static 2D images, our method can render the 3D virtual scene and text instances as an entirety. In this way, complex perspective transforms, various illuminations, and occlusions can be realized in our synthesized scene text images. Moreover, the same text instances with various viewpoints can be produced by randomly moving and rotating the virtual camera, which acts as human eyes. The experiments on the standard scene text detection benchmarks using the generated synthetic data demonstrate the effectiveness and superiority of the proposed method. The code and synthetic data will be made available.

1. Introduction

Performances of deep learning methods are highly relevant to the quality and quantity of training data. However, it is expensive and time-consuming to annotate the training data in domain-specific tasks. Image synthesis provides an effective way to alleviate this problem by generating large amounts of data with detailed annotations for specific tasks, which is labor-free and immune to human errors. As for scene text, it is elusive and prone to imprecision to annotate a training dataset that covers scene text instances in various occasions due to the diversity and variation of backgrounds, languages, fonts, lighting conditions, etc.. Therefore, synthetic data and related technologies are crucial for scene text tasks in research and applications. Moreover, in contrast to manually annotated data, whose labels are usually in line or word level, synthetic data can provide detailed information, such as character-level or even pixel-level annotations.

Previous scene text image synthesis methods have shown the potential of the synthetic data in scene text detection and recognition. The essential target of the scene text synthesizers is to generate scene text images similar to real-world images. In the previous methods, static background images are first analyzed using semantic segmentation/gpb-UCM and depth estimation, then text regions are proposed according to the predicted depth and semantic information. Here, “static” means single-view images, instead of multi-view images from the same scenarios. Zhan et al. further improve the synthetic effectiveness by introducing “semantic coherence”. “Homogeneous
regions”, and adaptive appearances.

These methods and their data are widely used in scene text detection and recognition, bringing promising performance gains. However, different from their synthesis procedures, humans usually perceive scene text from 3D worlds, where text may possess the variations of viewpoint, illumination, and occlusions. However, such variations are difficult to produce by inserting text into static 2D background images due to the lack of 3D information. Moreover, the candidate text regions generated by the results of segmentation algorithms may be rough and imprecise. These factors lead to a gap between the text images synthesized by existing engines and the real-world text images.

Instead, we propose an image synthesis engine incorporating 3D information, which is named as SynthText3D, generating synthetic scene text images from 3D virtual worlds. Specifically, text instances in various fonts are firstly embedded into suitable positions in a 3D virtual world. Then we render the virtual scene containing text instances with various illumination conditions and different visibility degrees in the 3D virtual world, where text and scene are rendered as integrity. Finally, we set the camera with different locations and orientations to make the projected 2D text images in various viewpoints.

Benefiting from the variable 3D viewpoint and the powerful virtual engine, SynthText3D is intuitively superior to previous methods in visual effects (see Fig. [1]) because of the reasons below: 1) Text and scenes in the 3D virtual world are rendered as an entirety, which makes the illumination/visibility, perspective transformation, and occlusion simulation more realistic. 2) We can obtain accurate surface normal information directly from the engine, which is beneficial to finding proper regions to place text instances. 3) SynthText3D can generate text instances with diverse viewpoints, various illuminations and different visibility degrees, which is akin to the way of observing of human eyes.

The contributions of this paper are three-fold: 1) We propose a novel method for generating synthetic scene text images from 3D virtual worlds, which is entirely different from the previous methods which render text with static background images. 2) The synthetic images produced from 3D virtual worlds yield fantastic visual effects, including complex perspective transforms, various illuminations, and occlusions. 3) We validate the superiority of the produced synthetic data by conducting experiments on standard scene text detection benchmarks. Our synthetic data is complementary to the existing synthetic data.


dedicated to the synthesis of images for scene text detection and recognition. Synth90K [8] proposes to generate synthesized images to train a scene text recognizer. Text are rendered through several steps, including font rendering, bordering/shadowing, coloring and projective distortion, before they are blended with crops from natural images and mixed with noises. However, the generated images are only cropped local regions which can not be directly used to train text detectors. SynthText [4] is then proposed to synthesize scene images with text printed on them for scene text detection. It searches for suitable areas, determines orientation, and applies perspective distortion according to semantic segmentation maps and depth maps, which are predicted by existing algorithms. The synthesized images have more realistic appearances as text seem to be correctly placed on the surface of objects in images. To compensate for the drawback that SynthText would print text onto unreasonable surfaces such as human faces which contradicts real-world scenarios, Zhan et al. [38] proposed selective semantic segmentation in order that word instances are only printed on sensible objects. They further replace the coloring scheme in SynthText and render the text instances adaptively to fit into the artistic styles of their backgrounds.

Different from these methods, which render text with static images, our method renders the text and the scene as integrity in the 3D virtual worlds.

2.2. Image Synthesis in 3D Virtual Worlds

In the past few years, synthesizing images with 3D models using graphics engines have made a figure and attracted much attention in several fields, including human pose estimations [34], indoor scene understanding [25, 24], outdoor/urban scene understanding [30, 32], and object detection [26, 33, 7]. The use of 3D models falls into one of the following categories: (1) Rendering 3D objects on top of static background real-world images [26, 34]; (2) Randomly arranging scenes filled with objects [25, 24, 30, 7]. (3) Using commercial game engine, such as Grand Theft Auto V (GTA V) [28, 23, 32] and the UnrealCV Project [27, 5, 33].

2.3. Scene Text Detection

Scene text detection has become an increasingly popular and essential research topic [37]. There are mainly two branches of ideology in fashion: Top-down methods [17, 13, 9, 22, 16, 6, 39, 20] treat text instances as generic objects under the framework of general object detection [29, 15], adapting those widely recognized object detection methods for unique challenges in scene text detection. These include rotating bounding boxes [12] and direct regression [39, 6] for multi-oriented text, region proposals of varying sizes [9] for long text. Bottom-up methods [36, 18, 2, 21], on the other hand, predict shape-agnostic local attributes or text segments, and therefore can predict irregular text [18] or very long text instances [21]. These
3. Methodology

3.1. Overview

SynthText3D provides an effective and efficient way to generate large-scale synthetic scene text images. The 3D engine in our method is based on Unreal Engine 4 (UE4), with an UnrealCV [27] plugin, which provides high rendering quality, realistic physics, editable illumination, and environment. We use 9 3D scene models including 5 indoor models and 4 outdoor models from the official marketplace of Unreal Engine to generate the synthetic data in this paper.

The pipeline of Synthtext3D mainly consists of a Camera Anchor Generation module (Sec. 3.2), a Text Region Generation module (Sec. 3.3), a Text Generation module (Sec. 3.4), and a 3D Rendering module (Sec. 3.5), as illustrated in Fig. 2.

Firstly, we initialize a small number of camera anchors for each 3D model manually. Then, the RGB image and the accurate surface normal map of each camera anchor can be obtained from the 3D engine. Next, the available text regions are generated based on the surface normal map. After that, we randomly select several regions from all the available text regions and generate text of random fonts, text content, and writing structures based on the size of the text regions. Text colors of the selected text regions are generated according to the background RGB image of the corresponding regions. Finally, we map the 2D text regions into the 3D virtual world and place the corresponding text in it.

Deep-learning frameworks require a large amount of labeled data, making it promising for the development of synthesis algorithms for scene text images.

3.2. Camera Anchor Generation

Previous image synthesis methods for scene text conduct manual inspection [4], or use annotated datasets [38] to discard the background images which contain text. Different from these methods which need a large number of static background images, we construct a small set of camera viewpoints (about 20 to 30 for each 3D model) in the virtual scenes, which are treated as the initial anchors. During collection, the operator controls the camera navigating in the scene, choosing views that suitable for placing text.

We select the camera anchors following a simple rule: at least one suitable region exists in the view. The human-guided camera anchor generation can get rid of unreasonable anchors such as these inside of objects or in the dim light (see Fig. 3).

3.3. Text Region Generation

Given a camera anchor, we can obtain a visible portion of the 3D scene, which contains the RGB image, the depth image, and the surface normal map of the view. Here, the generation of the text regions is based on the surface normal map. A surface normal of a coordinate point in a 3D virtual world is defined as a unit vector that is perpendicular to the tangent plane to that surface at the current coordinate point. Thus, the surface normal map is invariant to the viewpoint of the camera, which is more robust than the depth map. First, we generate a normal boundary map according to the surface normal map. Then, a stochastic binary search algorithm is adopted to generate text regions.

Synthetic images with various illuminations and different viewpoints can be produced from the rendered 3D virtual world.

https://www.unrealengine.com/marketplace
regions. At each initial location, we start with an initial rectangular bounding box with a minimum size that is large enough to place text, which is set as $32 \times 64$ pixels. The stride of the initial rectangular boxes is randomly chosen from $(12, 24, 36)$. To retrieve the maximum areas, we propose a stochastic binary search method. At each search step, we randomly expand one side of the rectangle. The expansion follows the rule of a binary search: the lower bound is set as the current edge location, the upper bound is set as the corresponding edge of the image. If the expanded rectangle does not cross the normal boundary, then the lower bound is updated to the middle point; otherwise, the upper bound is updated to the middle point. The algorithm converges when the upper bound and lower bound of each edge equals each other. After box generation at all anchors, we check each box one by one in a random order, and if any box overlaps with any other box, it is discarded. Such randomness is used to enhance the diversity of boxes. Note that stochastic binary search method is aimed to randomly find suitable text regions. Thus, we don’t strictly find the maximum regions. Examples of results generated by the algorithm can be found in Fig. 4c.

3.4. Text Generation

Once the text regions are generated, we randomly sample several available text regions for text rendering. Given a text region along with its RBG image, the text generation module is aimed to sample text content with certain appearances, including the fonts and text colors. In order to compare with SynthText [4] fairly, we use the same text sources as [4], which is from the Newsgroup 20 dataset. The text content is randomly sampled from the text sources with three structures, including words, lines (up to 3), and paragraphs.

We randomly sample the fonts from Google Fonts[3] which is used to generate the texture of the text. The text color is determined by the background of the text region, using the same color palette model in [4]. The texture and the color of each region will be fed into the 3D Rendering module to perform 3D rendering.

3.5. 3D Rendering

The 3D rendering includes two sub-modules. The first is the text placing module, which is aimed to place text into the 3D virtual worlds. The second is the rendering module, where the illumination and visibility adjustment, viewpoint transformation, occlusions can be performed.

3.5.1 Text Placing

With 2D text regions prepared, the text placing module first projects the 2D regions into the virtual scene by estimating

\[\text{https://fonts.google.com/}\]
Figure 4: Illustration of Text Region Generation. (a) are the original images; (b) are the surface normal maps; (c) are the normal boundary maps; (d) and (e) are the generated text regions.

Figure 5: Illustration of text placing.

2D-to-3D Region Projection Previous works (e.g., [4, 38]) achieve this by predicting a dense depth map using CNN. In our method, we transform the 2D regions into the 3D virtual world according to the geometry information.

We adopt a coarse-to-fine strategy to project the 2D text regions into the 3D virtual worlds for projection precision. Assume that $p_j = (x_j, y_j)$ is a coordinate point in the 2D map, $d_j$ is the depth value of $p_j$, the coarse-grained 3D coordinates $P = (X_j, Y_j, Z_j)$ can be calculated as follows:

$$\begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} = K \times d_j \times \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix}$$

where $K$ is the internal reference matrix of the camera, which is an attribute parameter of the camera. The coarseness comes from the limitation of depth map where the pixel value is presented by an integer between [0,1024). However, as illustrated in Fig.6, a deviation occurs between the location of integer depth and float one.

In order to get the fine-grained coordinates, we adopt ray casting [31], a process of determining whether and where a ray intersects an object. As shown in Fig.6, we initialize the ray $V = (P_i - P_v)$, where $P_i$ is a coarse point estimated in Eq.2. Then, the fine-grained point $P_f$ can be obtained by the ray casting, which detects the nearest hit point to $P_v$.

After projection, the regular bounding boxes are deformed to irregular 3D regions due to the geometric projective transformation shown in Eq.2. In the real world, the shapes of text are usually rectangles, and the orientation of the text are horizontal or vertical. We achieve this by clipping the sides of 3D quadrilaterals iteratively until it turns to 3D axis-aligned bounding boxes.

Text Deformation In natural situations, not all text regions are flat planes, such as the surface of the bottle and the clothes, where text need to be deformed to fit the target surface.

The deformation algorithm is illustrated as Fig.5. We treat the text plane as a triangulated mesh, and text as the texture map of the mesh. Four corner vertexes of the mesh are first fixed to the region corners, and then intermediate vertexes are transformed to the nearest position on the surface of the target object. Finally, the texture coordinates of vertexes are estimated according to the Euclidean distance relative to the corner vertexes.
3.5.2 Rendering

We establish several environment settings for each scene respectively. For each indoor scene, we build three kinds of illuminations: normal illuminance, brightness, and darkness. In addition to the illuminations, we add a fog environment for outdoor scenes.

After the text are placed into the 3D virtual world, we can augment to produce synthetic images with the following steps. First, we render the virtual objects along with text under various light intensity or color circumstances. Second, we randomly adjust the camera positions to conduct viewpoint transformation. For each camera anchor, we sample 20 images using the rendering mentioned above. In this way, we can produce the synthetic images of the same text instances with various viewpoints, which contain rich perspective transformations.

Note that we give a video in the supplementary materials to show our rendering process in the 3D virtual worlds.

4. Experiments

The synthetic images are synthesized with a workstation with a 3.20GHz Intel Core i7-8700 CPU, an Nvidia GeForce GTX 1060 GPU, and 16G RAM. The speed in the synthesizing period is about 2 seconds/image, with a size of 720 × 1080.

4.1. Datasets

SynthText [4] is a synthetic dataset consisting of 800k synthetic images, which are generated with 8k background images. Each background image is used to generate images of different text for 100 times. Other than using the whole set, we also randomly select 10k images from them, which is named as SynthText-10k.

Verisimilar Image Synthesis Dataset (VISD) is a synthetic dataset proposed in [38]. It contains 10k images synthesized with 10k background images. Thus, there are no repeated background images for this dataset. The rich background images make this dataset more diverse.

ICDAR 2013 Focus Text (ICDAR 2013) [11] is a dataset proposed in Challenge 2 of the ICDAR 2013 Robust Reading Competition. There are 229 images in the training set and 233 images in the test set.

ICDAR 2015 Incidental Text (ICDAR 2015) [10] is a popular benchmark for scene text detection. It consists of 1000 training images and 500 testing images, which are incidentally captured by Google glasses.

MLT [7] is a dataset for ICDAR 2017 competition on multilingual scene text detection. It is composed of complete scene images which come from 9 languages representing 6 different scripts. There are 7,200 training images, 1,800 validation images and 9,000 testing images in this dataset.

[4] http://rrc.cvc.uab.es/?ch=8&com=introduction
4.2. Visual Analysis

Two critical elements affect the visual effect of the synthetic data. One is the suitable text regions. Another one is the rendering effect.

4.2.1 Suitable Text Regions

Previous methods \cite{4, 38} use saliency/gpb-UCM or depth maps to determine the suitable text regions in the static background. However, the segmentation or depth maps are estimated, which are not accurate enough. Thus, the text can be placed on unsuitable regions (see Fig. 8a). However, we can obtain accurate depth, segmentation, and surface normal information from the 3D engine easily. In this way, the text regions in our synthetic data are more realistic, as shown in Fig. 8b.

4.2.2 Rendering Effect

Our proposed SynthText3D can synthesis scene text images by rendering text and the background scene together. Thus, it can achieve more complex rendering than previous methods which render text with static images, such as the illumination and visibility adjustment, diverse viewpoints, occlusions, as shown in Fig. 7.

In Fig. 7a, we can see that the illuminations are diverse among the four images and the last image is in a foggy environment. The illumination effect can also be embodied in Fig. 7c where the text and scene are in a dark environment. The viewpoint transformation is illustrated in Fig. 7b. Profiting from the 3D perspective, we can change the viewpoint of the same text block to achieve realistic perspective transformation flexibly. Last but not least, occlusions can be performed by adjusting the viewpoint, as shown in Fig. 7c.

As known, the rendering mentioned above is commonly seen in real-world images. However, limited to the static background images, which means the camera position, direction, and environment illumination are fixed, it is theoretically limited for previous methods to simulate such rendering. Thus, the rendering effect of our proposed synthetic data is more realistic.

4.3. Scene Text Detection

We conduct experiments on standard scene text detection benchmarks using an existing scene text detection model.
| Training data        | ICDAR 2015 | ICDAR 2013 | MLT  |
|---------------------|------------|------------|------|
|                     | P | R | F | P | R | F | P | R | F |
| SynthText 10k       | 40.1 | 54.8 | 46.3 | 54.5 | 69.4 | 61.1 | 34.3 | 41.4 | 37.5 |
| SynthText 800k      | 67.1 | 51.0 | 57.9 | 68.9 | 66.4 | 67.7 | 53.9 | 36.5 | 43.5 |
| VISD 10k            | 73.3 | 59.5 | 65.7 | 73.2 | 68.5 | 70.8 | 58.9 | 40.0 | 47.6 |
| Ours 10k            | 64.5 | 56.7 | 60.3 | 75.8 | 65.6 | 70.4 | 50.4 | 39.0 | 44.0 |
| Ours 5k + VISD 5k   | 71.1 | 64.4 | **67.6** | 76.5 | 71.4 | **73.8** | 57.6 | 44.2 | **49.8** |

Table 1: Detection results with different synthetic data. “5k”, “10k” and “800k” indicate the number of images used for training.

| Training Data        | P | R | F |
|---------------------|---|---|---|
| Real (1k)           | 84.8 | 78.1 | 81.3 |
| SynthText (10k) + Real (1k) | 85.7 | 79.5 | 82.5 |
| VISD (10k) + Real (1k) | 86.5 | 80.0 | 83.1 |
| Ours (10k) + Real (1k) | 86.6 | 79.2 | 82.7 |

Table 2: Detection results on ICDAR 2015. “Real” means the training set of ICDAR 2015. “VISID” is short for Verisimilar Image Synthesis Dataset [38].

EAST [39]. EAST is a previous state-of-the-art method which is both fast and accurate. We adopt an open-source implementation[5] to conduct all the experiments. We use ResNet-50 [5] as the backbone. All models are trained with 4 GPU with a batch size of 56.

4.3.1 Pure Synthetic Data

We train the EAST models purely with synthetic data to prove the synthesizing quality of our synthetic images. The experimental results are listed in Tab. 1. When using 10k images, our synthetic data outperforms SynthText [4] by 14, 9.3 and 6.5 percents in terms of F-measure on ICDAR 2015, ICDAR 2013 and MLT, respectively.

Note that “Ours 10k” surpasses “SynthText 800k” among all benchmarks, which effectively demonstrates the quality of our method, as the models trained with our synthetic data, compared to those trained with SynthText, can achieve better performance even with less training images.

It is not fair to directly compare with VISD [38] as its code and some important details were not given. The comparison with the milestone work SynthText [4] is more convincing because we follow the text generation step with it, including the corpus, color palette, et al. Though the detection F-score of ours is lower than VISD, we would stress that our generated images possess more realistic look in term of perspective transforms, various illuminations, and occlusions, as shown in Fig. 7.

We conduct an experiment by training with mixed data (Ours 5K + VISD 5k), achieving top accuracies as shown in Tab. 1. This phenomenon demonstrates that text generated from 3D is complementary to existing synthesized data.

![Figure 9: Detection results with different training set. Green boxes: correct results; Red boxes: wrong result. As shown, the occlusion and perspective cases can be improved when our synthetic data are used.](https://github.com/argman/EAST)

4.3.2 Combination of Synthetic Data & Real Data

We also conduct experiments with the combination of the synthetic data and the real-world data, to demonstrate that our synthetic can further improve the performance based on the real-world data. The combination strategy is that we first pre-train the model with the synthetic data, and then fine-tune the model with real-world data. As shown in Tab. 2, 1.4 percents performance gain is acquired on ICDAR 2015 since some challenges cases are improved, such as occlusion and perspective transformation, as shown in Fig. 9.

VISD [38] uses 10k real-world background images, which provide rich objects and textures. However, our synthetic images are projected by the 3D virtual world based on about 200 camera anchors. This may be the reason that our improvement is slightly lower than VISD.

Note that all synthetic data listed in Tab. 2 can improve the performance similarly. The improvements are not significant since the synthetic data is only used for pre-training and the performance of the baseline is not weak. However, the improvements are free of labor since the synthetic data can be automatically produced.
4.4. Limitations

The main limitation of our method is that we need to select the camera anchors manually, though it is relatively easy to accomplish since there are only about 20-30 camera anchors for each virtual world. However, considering that previous methods also need manual inspection or annotations to filter background images which contain text, our manual selection is acceptable. We will try to improve it in the future work by introducing an algorithm to generate the camera anchors automatically.

5. Conclusion

In this paper, we proposed SynthText3D, a synthetic data engine, which can produce verisimilar scene text images for scene text detection from the 3D virtual worlds. Different from the previous methods which render text with static background images, SynthText3D has two advantages. One is that we can obtain accurate surface normal maps instead of the estimated segmentation or depth map, to determine the suitable text regions. Another one is that the text and scene can be rendered as an entirety in our method. In this way, we can achieve various illuminations and visibility, diverse viewpoints and occlusions. The visual effect and the experimental results on standard scene text benchmarks demonstrate the effectiveness of the synthesized data.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence, 33(5):898–916, 2011.
[2] D. Deng, H. Liu, X. Li, and D. Cai. Pixellink: Detecting scene text via instance segmentation. In Proc. AAAI, pages 6773–6780, 2018.
[3] O. Ganoni and R. Mukundan. A framework for visually realistic multi-robot simulation in natural environment. CoRR, abs/1708.01938, 2017.
[4] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In Proc. CVPR, pages 2315–2324, 2016.
[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proc. CVPR, 2016.
[6] W. He, X.-Y. Zhang, F. Yin, and C.-L. Liu. Deep direct regression for multi-oriented scene text detection. In Proc. ICCV, 2017.
[7] S. Hinterstoisser, O. Pauly, H. Heibel, M. Marek, and M. Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object instance detection. CoRR, abs/1902.09967, 2019.
[8] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. CoRR, abs/1406.2227, 2014.
[9] Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo. R2CNN: rotational region CNN for orientation robust scene text detection. CoRR, abs/1706.09579, 2017.
[10] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In Proc. ICDAR, pages 1156–1160, 2015.
[11] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. de las Heras. Icdar 2013 robust reading competition. In Proc. ICDAR, pages 1484–1493, 2013.
[12] M. Liao, B. Shi, and X. Bai. Textboxes++: A single-shot oriented scene text detector. IEEE Transactions on Image Processing, 27(8):3676–3690, 2018.
[13] 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, pages 4161–4167, 2017.
[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In Proc. ECCV, pages 740–755, 2014.
[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In Proc. ECCV, pages 21–37, 2016.
[16] Y. Liu and L. Jin. Deep matching prior network: Toward tighter multi-oriented text detection. In Proc. CVPR, 2017.
[17] Y. Liu, L. Jin, S. Zhang, and S. Zhang. Detecting curve text in the wild: New dataset and new solution. CoRR, abs/1712.02170, 2017.
[18] 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. ECCV, 2018.
[19] S. Lu, C. Tan, and J.-H. Lim. Robust and efficient saliency modeling from image co-occurrence histograms. IEEE transactions on pattern analysis and machine intelligence, 36(1):195–201, 2014.
[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. ECCV, 2018.
[21] P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai. Multi-oriented scene text detection via corner localization and region segmentation. In Proc. CVPR, 2018.
[22] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. IEEE Transactions on Multimedia, 20(11):3111–3122, 2018.
[23] M. Martinez, C. Sitawarin, K. Finch, L. Meincke, A. Yablonka, and A. L. Kornhauser. Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars. CoRR, abs/1712.01397, 2017.
[24] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison. Scenenet RGB-D: 5m photorealistic images of synthetic indoor trajectories with ground truth. CoRR, abs/1612.05079, 2016.
[25] J. Papon and M. Schoeler. Semantic pose using deep networks trained on synthetic rgb-d. In Proc. ICCV, pages 774–782, 2015.
[26] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3d models. In Proc. ICCV, pages 1278–1286, 2015.
[27] W. Qiu and A. Yuille. Unrealcv: Connecting computer vision to unreal engine. In Proc. ECCV, pages 909–916, 2016.
[28] C. Quiter and M. Ernst. deepdrive/deepdrive: 2.0, 2018.
[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proc. NIPS, pages 91–99, 2015.
[30] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In Proc. CVPR, pages 3234–3243, 2016.
[31] S. D. Roth. Ray casting for modeling solids. Computer Graphics & Image Processing, 18(2):109–144, 1982.
[32] F. S. Saleh, M. S. Aliakbarian, M. Salzmann, L. Petersson, and J. M. Alvarez. Effective use of synthetic data for urban scene semantic segmentation. In Proc. ECCV, pages 86–103, 2018.
[33] J. Tremblay, T. To, and S. Birchfield. Falling things: A synthetic dataset for 3d object detection and pose estimation. In Proc. CVPR Workshops, pages 2038–2041, 2018.
[34] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In Proc. CVPR, pages 109–117, 2017.
[35] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In Pattern Recognition (ICPR), 2012 21st International Conference on, pages 3304–3308, 2012.
[36] Y. Wu and P. Natarajan. Self-organized text detection with minimal post-processing via border learning. In Proc. CVPR, pages 5000–5009, 2017.
[37] Q. Ye and D. Doermann. Text detection and recognition in imagery: A survey. IEEE transactions on pattern analysis and machine intelligence, 37(7):1480–1500, 2015.
[38] F. Zhan, S. Lu, and C. Xue. Verisimilar image synthesis for accurate detection and recognition of texts in scenes. In Proc. ECCV, 2018.
[39] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. EAST: An efficient and accurate scene text detector. In Proc. CVPR, 2017.
Appendices

We detailedly describe the stochastic binary search mentioned in Sec. 3.3.2 in Algorithm 1.

```
Input: binary normal boundary map: \( M(H \times W) \), initial center point: \((X_0, Y_0)\)
Output: top-left point: \((X_1, Y_1)\), bottom-right point: \((X_2, Y_2)\)

\[ S_W = 64; \; S_H = 32; \quad \text{▷ The width and height of minimal initial box as mentioned in Sec. 3.3.2;} \]
\[ X_{(1,\text{upper})} = 0; \; X_{(1,\text{lower})} = X_0 - S_W/2; \]
\[ Y_{(1,\text{upper})} = 0; \; Y_{(1,\text{lower})} = Y_0 - S_H/2; \]
\[ X_{(2,\text{lower})} = X_0 + S_W/2; \; X_{(2,\text{upper})} = W; \]
\[ Y_{(2,\text{lower})} = Y_0 + S_H/2; \; Y_{(2,\text{upper})} = H; \]

while expandable do
  \[ \text{▷ “expandable” indicates whether it was successfully expanded (not reaching boundaries) in the last iteration;} \]
  randomly select side from \([\text{left}, \text{right}, \text{top}, \text{bottom}]\);
  if side is left \&\& abs(\(X_{(1,\text{lower})} - X_{(1,\text{upper})}\)) > 1 then
    \[ \text{mid} = \lfloor (X_{(1,\text{lower})} + X_{(1,\text{upper})})/2 \rfloor; \]
    if sum(\(M[Y_{(1,\text{lower})} : Y_{(2,\text{lower})}, \text{mid} : X_{(2,\text{lower})}]\)) >= 1 then
      \[ X_{(1,\text{upper})} = \text{mid}; \]
    else
      \[ X_{(1,\text{lower})} = \text{mid} \]
  end
  else
    \[ ...; \quad \text{▷ update } X_{(2,\bullet)}, Y_{(1,\bullet)}, Y_{(2,\bullet)} \text{ similarly when side is right, top, bottom;} \]
  end
\[ X_1 = X_{(1,\text{lower})}; \; Y_1 = Y_{(1,\text{lower})}; \]
\[ X_2 = X_{(2,\text{lower})}; \; Y_2 = Y_{(2,\text{lower})}; \]
return \((X_1, Y_1), (X_2, Y_2)\)
```

**Algorithm 1:** Stochastic binary search