TextNeRF: A Novel Scene-Text Image Synthesis Method based on Neural Radiance Fields

Supplementary Material

6. Derivation of Coordinate projection

Assume that $\mathbf{p}_1^{pixel} = (x_1^{pixel}, y_1^{pixel})$ is a point on the pixel plane corresponding to Camera_1, and it is converted to homogeneous coordinate for subsequent calculations:

$$(\mathbf{p}_1^{pixel})_{homo} = \begin{pmatrix} x_1^{pixel} \\ y_1^{pixel} \\ 1 \end{pmatrix}. \tag{12}$$

To obtain the 3D point $\mathbf{p}_1^{camera}$ in the coordinate system of Camera_1, the homogeneous coordinate $(\mathbf{p}_1^{pixel})_{homo}$ is multiplied by the inverse of the intrinsic matrix of Camera_1, then multiplied by the depth value of $\mathbf{p}_1^{pixel}$, which is an output of volume rendering:

$$\begin{aligned} \mathbf{p}_1^{camera} &= d_1\mathbf{K}_1^{-1}(\mathbf{p}_1^{pixel})_{homo} \\ &= d_1 \begin{bmatrix} f_1^x & 0 & c_1^x \\ 0 & f_1^y & c_1^y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{pmatrix} x_1^{pixel} \\ y_1^{pixel} \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{d_1(x_1^{pixel}-c_1^x)}{f_1^x} \\ \frac{d_1(y_1^{pixel}-c_1^y)}{f_1^y} \\ d_1 \end{pmatrix}. \end{aligned} \tag{13}$$

Then the 3D point $\mathbf{p}_1^{camera}$ is converted to homogeneous coordinate and multiplied by the pose matrix $\mathbf{P}_1$ of Camera_1 to obtain the homogeneous coordinate $(\mathbf{p}_1^{world})_{homo}$ in the world coordinate system:

$$\begin{aligned} (\mathbf{p}_1^{world})_{homo} &= \mathbf{P}_1(\mathbf{p}_1^{camera})_{homo} \\ &= \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \frac{d_1(x_1^{pixel}-c_1^x)}{f_1^x} \\ \frac{d_1(y_1^{pixel}-c_1^y)}{f_1^y} \\ d_1 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} x_1^{world} \\ y_1^{world} \\ z_1^{world} \\ 1 \end{pmatrix}. \end{aligned} \tag{14}$$

A 3D point's coordinate in the world coordinate system can be projected onto an image plane from any camera perspective. Therefore, $\mathbf{p}_1^{world}$ is subsequently projected onto the image plane of Camera_2. To achieve this, $(\mathbf{p}_1^{world})_{homo}$ is first converted to the camera coordinate system of Camera_2 by multiplying with the inverse of the pose matrix $\mathbf{P}_2$ of Camera_2:

$$\begin{aligned} (\mathbf{p}_2^{camera})_{homo} &= \mathbf{P}_2^{-1}(\mathbf{p}_1^{world})_{homo} \\ &= \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_1^{world} \\ y_1^{world} \\ z_1^{world} \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} x_2^{camera} \\ y_2^{camera} \\ z_2^{camera} \\ 1 \end{pmatrix}. \end{aligned} \tag{15}$$

Then the resulting homogeneous coordinate is converted to Euclidean coordinate and projected onto the imaging plane at a distance of 1 from the Camera_2's origin:

$$\begin{aligned} \mathbf{p}_2^{camera} &= \begin{pmatrix} x_2^{camera} \\ y_2^{camera} \\ z_2^{camera} \end{pmatrix} \\ &= z_2^{camera} \begin{pmatrix} \frac{x_2^{camera}}{z_2^{camera}} \\ \frac{y_2^{camera}}{z_2^{camera}} \\ 1 \end{pmatrix}. \end{aligned} \tag{16}$$

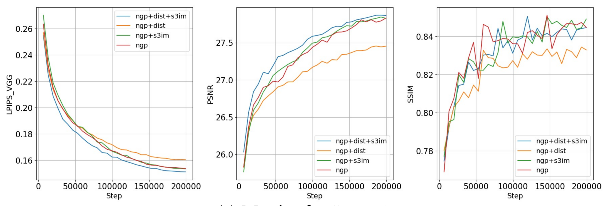
Finally, the pixel coordinate $\mathbf{p}_2^{pixel} = (x_2^{pixel}, y_2^{pixel})$ is obtained by multiplying the point on the imaging plane with the intrinsic matrix $\mathbf{K}_2$ of Camera_2:

$$\begin{aligned} (\mathbf{p}_2^{pixel})_{homo} &= \frac{1}{z_2^{camera}}\mathbf{K}_2\mathbf{p}_2^{camera} \\ &= \begin{bmatrix} f_2^x & 0 & c_2^x \\ 0 & f_2^y & c_2^y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \frac{x_2^{camera}}{z_2^{camera}} \\ \frac{y_2^{camera}}{z_2^{camera}} \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{f_2^x x_2^{camera}}{z_2^{camera}} + c_2^x \\ \frac{f_2^y y_2^{camera}}{z_2^{camera}} + c_2^y \\ 1 \end{pmatrix}, \end{aligned} \tag{17}$$

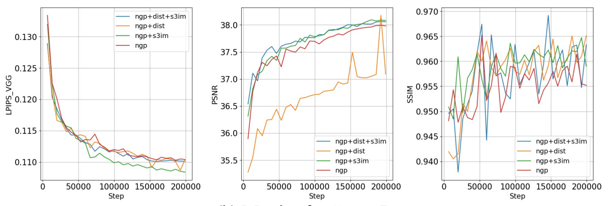
and the pixel coordinate of $\mathbf{p}_2^{pixel}$ is obtained by converting the homogeneous coordinate to Euclidean coordinate:

$$\mathbf{p}_2^{pixel} = \begin{pmatrix} \frac{f_2^x x_2^{camera}}{z_2^{camera}} + c_2^x \\ \frac{f_2^y y_2^{camera}}{z_2^{camera}} + c_2^y \end{pmatrix}. \tag{18}$$

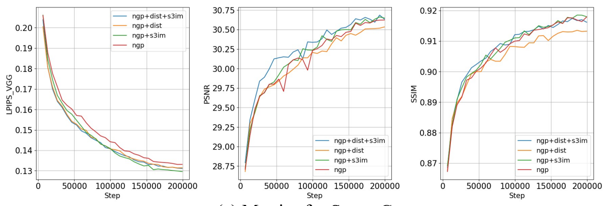
Now, the semantic label corresponding to the point $\mathbf{p}_2^{pixel}$ in the labeled Image_2 can be selected as the proxy label for $\mathbf{p}_1^{pixel}$ in the unlabeled Image_1. Based on this, the projected semantic loss for unlabeled images can be calculated in a semi-supervised manner.



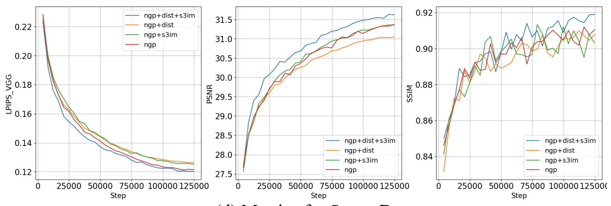
(a) Metrics for Scene A



(b) Metrics for Scene B

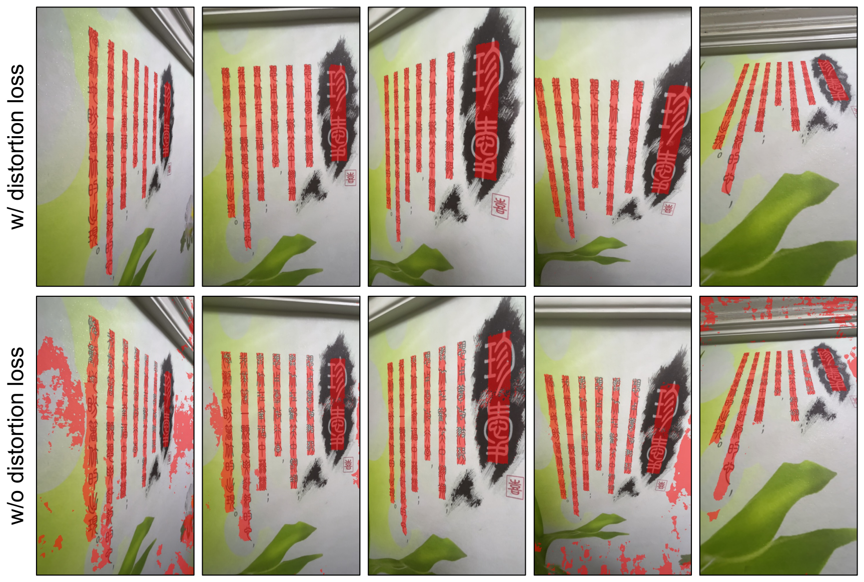


(c) Metrics for Scene C

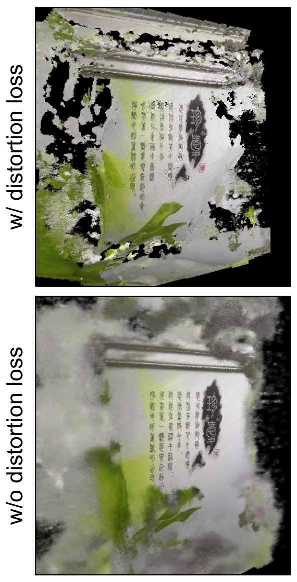


(d) Metrics for Scene D

Figure 6. Evaluation metrics for scene modeling under different training configurations.



(a) semantic results rendered from different views



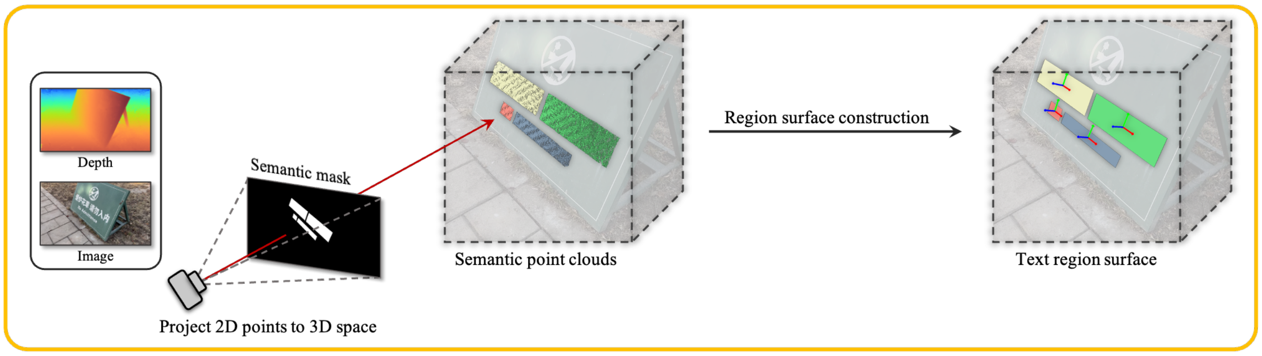
(b) reconstruction results

Figure 7. Effect of distortion loss. (a) displays the rendering results of the semantic mask from multiple views. (b) presents a visual comparison of the scene reconstruction effect of the model under two different conditions.

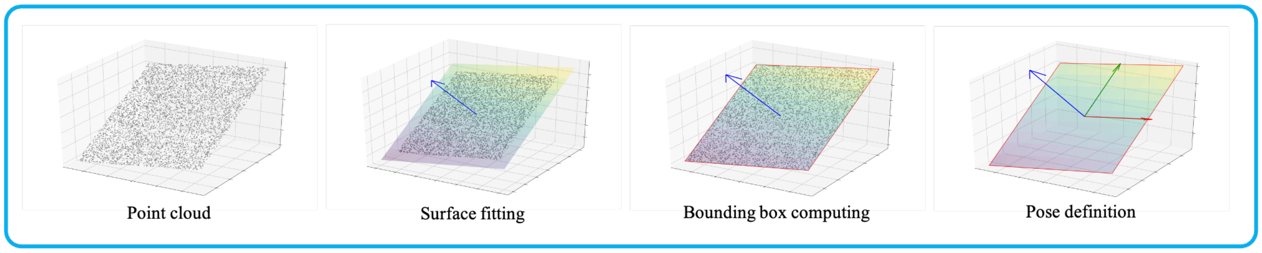
7. Enhanced Analysis of Scene Modeling with Neural Radiance Fields

In this section, we delve deeper into the nuances of scene modeling, specifically focusing on the training of Neural Radiance Fields [19] (NeRFs) during the scene modeling phase. We conducted a series of ablation experiments to assess the impact of various training enhancements on the quality of the rendered scenes. A subset of scenes was chosen for this experimental evaluation, where we compared the results of using Instant-NGP [20] alone, Instant-NGP with added distortion loss [2] as a regularization term, Instant-NGP trained with the S3IM loss [31] and Instant-NGP trained with the distortion and S3IM loss.

As with Sec 4.1, we selected 120 images per scene and divided them into training and testing sets at an 85% to 15% ratio. Each scene was trained for 30 epochs, and the PSNR, SSIM, and LPIPS metrics were recorded for each epoch on the test images. Fig. 6 illustrates the variations in



(a) Text region modeling process



(b) Details about surface construction

Figure 8. The elaborate process of text region surface modeling: (a) illustrating the general procedure of text region modeling, and (b) elucidating the details of text surface construction in 3D space.

test metrics for four distinct scenes under various configuration conditions. From these metrics, it is observable that the inclusion of distortion loss slightly degrades the rendering performance of NeRF. However, the utilization of the S3IM training paradigm not only fosters faster convergence but also mitigates the adverse effects of distortion loss, resulting in a slight enhancement across all metrics. This validates the efficacy of our configuration for scene modeling.

The rationale behind incorporating distortion loss, despite its mild negative impact on synthetic image quality, was further validated through comparative analysis. We examined the significance of distortion loss for accurate scene geometry construction and its subsequent influence on semi-supervised semantic learning. Fig. 7 (a) displays the effect of two settings on the semantic learning of text regions within the NeRF training outcomes. Fig. 7 (b) showcases the learning results of scene geometry from a distant viewpoint under both settings. We observed that omitting distortion loss led to the emergence of many cloud-like floaters around the scene. While these floaters could enhance the image rendering effects from specific viewpoints, they proved to be detrimental from others. More specifically, these floaters introduced incorrect scene geometry, which significantly increased the error in coordinate projection and reduced the effectiveness of semantic learning for unlabeled text, potentially causing overfitting to labeled images. Consequently, incorporating distortion loss serves to regulate the learning process of the radiance field. It ensures that voxels with higher density are more accurately concentrated on the actual surfaces of the scene, which enhances geometric reconstruction, reduces coordinate projection errors, and amplifies the impact of semi-supervised learning on text regions.

8. Precise Text Surface Modeling

In this section, we present a detailed account of text surface modeling, supplementing the description provided in Sec 3.2. The overall workflow for modeling the surface of text regions is illustrated in Fig. 8 (a). Initially, we leverage the scene geometry information (depth maps) constructed by NeRF to project the 2D semantic masks of text regions from various training viewpoints into 3D space, resulting in a point cloud representation of the text areas. Subsequently, we fit a continuous surface to each text region's point cloud, delineate the contours, and generate the pose in 3D space. For rendering images from novel viewpoints, the constructed text region annotations are simply projected from 3D space into the corresponding 2D views, enabling the annotation of scene text content for that viewpoint.

Fig. 8 (b) details the construction process of the text surface. Upon obtaining the 3D point cloud of the text surface, we first filter out obvious outliers from the cloud. Then,

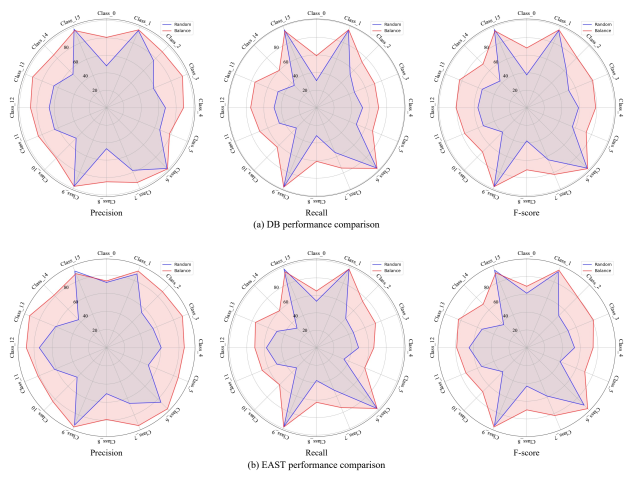


Figure 9. Performance comparison of detectors trained on the two datasets across various text pose categories.

a smooth plane is fitted from these points using the least squares method, representing the surface where the text region resides. Next, we calculate the projection points of the cloud onto this fitted plane and determine the text region's contour bounding box based on these projections. Subsequently, the normal vector of the text surface is used as the z-axis component of the rotation matrix in the text region's pose, while the directions along the longer and shorter sides of the bounding box represent the x and y-axis components, respectively. The center of the text contour is calculated to serve as the translation vector for the pose, thus completing the construction process of the text region's surface pose.

9. More experiment results on robustness evaluation.

This section demonstrates the unique advantages of our scene text image synthesis method in excavating model biases and appraising performance. To this end, we conducted a text pose robustness evaluation of the text detectors DB [15] and EAST [38] mentioned in multi-view evaluation part of Sec 4.2. Text instances from our test set were categorized into pose categories as analyzed in Sec 4.1, and the detectors' precision, recall, and F-score were computed for each category. As depicted in Fig. 9 (a) and (b), we observed a pronounced bias in detectors trained on the randomly sampled dataset, with disparities in F-scores across pose categories peaking at 59.93 and 58.02 for DB and EAST, respectively. Conversely, detectors trained on the viewpoint-balanced dataset exhibited greater robustness, narrowing the F-score discrepancy to 30.05 and 27.69 across different pose categories. We noted that text with poses perpendicular to the camera's capture angle achieved superior detection performance, while poses with significant perspective distortion posed greater challenges. Moreover, a comparative analysis of both detectors across identical pose categories revealed that models trained with a viewpoint-balanced training set outperformed those trained on randomly sampled data. This was particularly evident for text instances with intense perspective distortion, which are inherently more difficult to detect. These experiments further corroborate the efficacy of our viewpoint-balancing strategy in synthesizing quality data and training models with enhanced robustness against text pose variations.

10. Additional Experimental Subjects

Previous research primarily utilized EAST, due to its straightforward architecture, rapid detection capabilities, and high accuracy, establishing it as a time-tested model. In our work, we went beyond prior conventions and incorporated another detector DB, which holds a more prominent influence compared to EAST and also serves as a fundamental model in numerous open-source OCR frameworks, such as OpenCV, PaddleOCR, and WeChatOCR engine. Meanwhile, we also have conducted additional experiments using two recent text detectors, TextFuseNet and MixNet. The results can be found in Table 3. It should be noted that the TextFuseNet pipeline involves instance segmentation at the character level, so we modified the original implementation to avoid using character-level annotation during training. And for MixNet, we also adapted the original code, since we found the model with the midline prediction branch is hard to converge and the authors did not perform experiments on the benchmarks we utilized. Overall, the performance of these two detectors is superior; however, they exhibit more complex structures and necessitate extensive data processing operations.

Train data	TextFuseNet			MixNet		
	IC13	IC15	MLT17	IC13	IC15	MLT17
VISD-10K	67.41	62.05	43.14	72.41	64.90	44.21
ST3D-10K	64.13	65.31	48.13	71.58	67.12	49.16
UT-10K	68.44	59.74	50.60	74.24	60.84	52.04
Ours-10K	**70.16**	**67.28**	**51.01**	**75.30**	**68.39**	**54.28**
Real	54.42	79.46	60.31	55.31	82.09	62.76
VISD-10K + Real	72.30	84.02	62.12	81.41	84.03	64.78
ST3D-10K + Real	71.98	83.58	61.73	80.63	85.44	64.07
UT-10K + Real	74.16	85.47	63.17	81.87	86.18	65.84
Ours-10K + Real	**76.01**	**85.93**	**64.24**	**82.07**	**87.52**	**66.17**

Table 3. Results (F1-score) of TextFuseNet and MixNet pretrained on synthetic and finetuned on real datasets.

11. More synthesized results

In this section, we provide additional examples of the synthesized scene text images in Fig. 10 and 11.

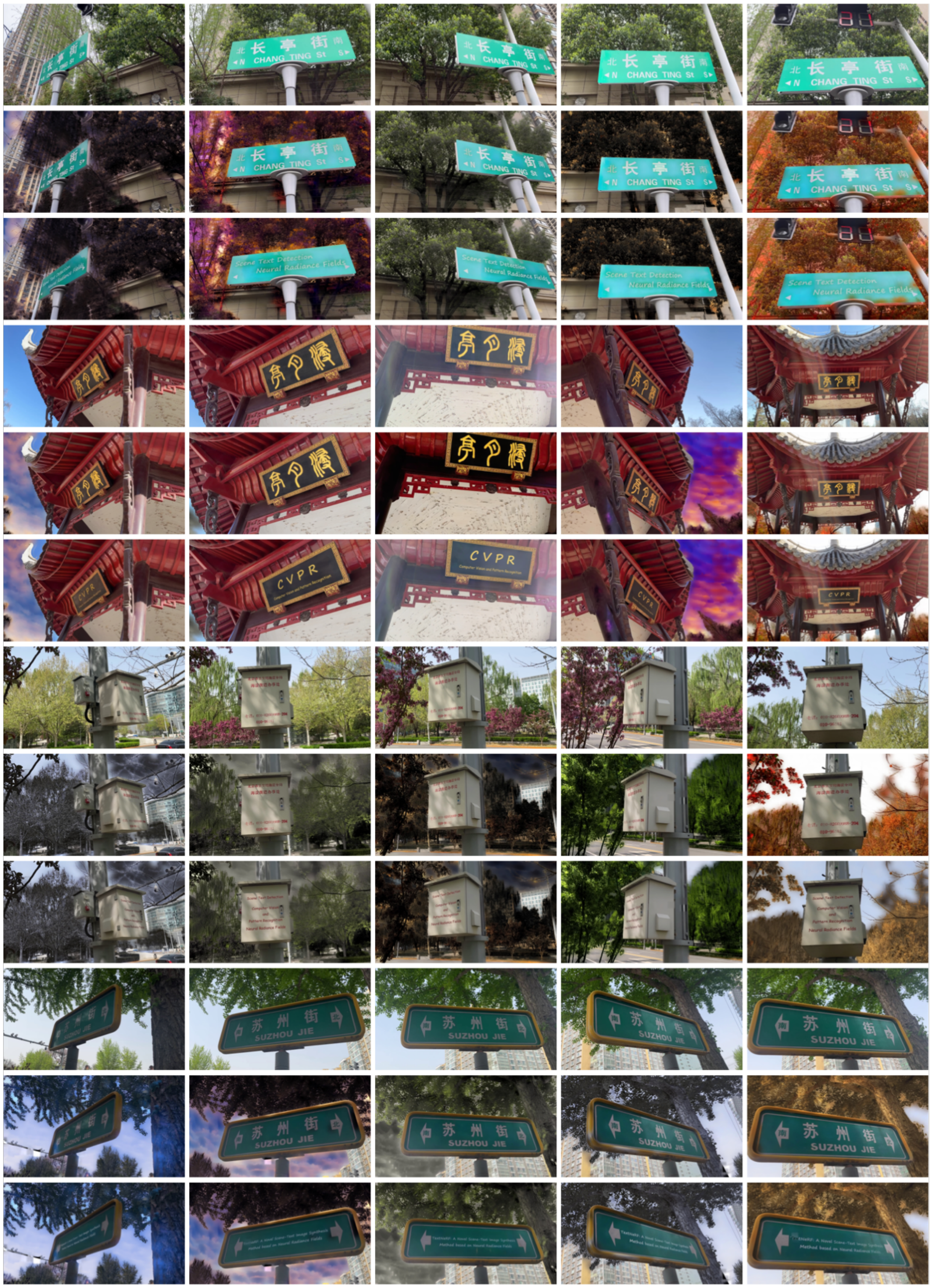

北 长 亭 街 南
◄N CHANG TING St S►
Scene Text Detection
Neural Radiance Fields
CVPR
Computer Vision and Pattern Recognition
Scene Text Detection
Computer Vision and Pattern Recognition
Neural Radiance Fields
苏 州 街
SUZHOU JIE
TextNeRF: A Novel Scene-Text Image Synthesis Method based on Neural Radiance Fields

Figure 10. Results of generated scene text images

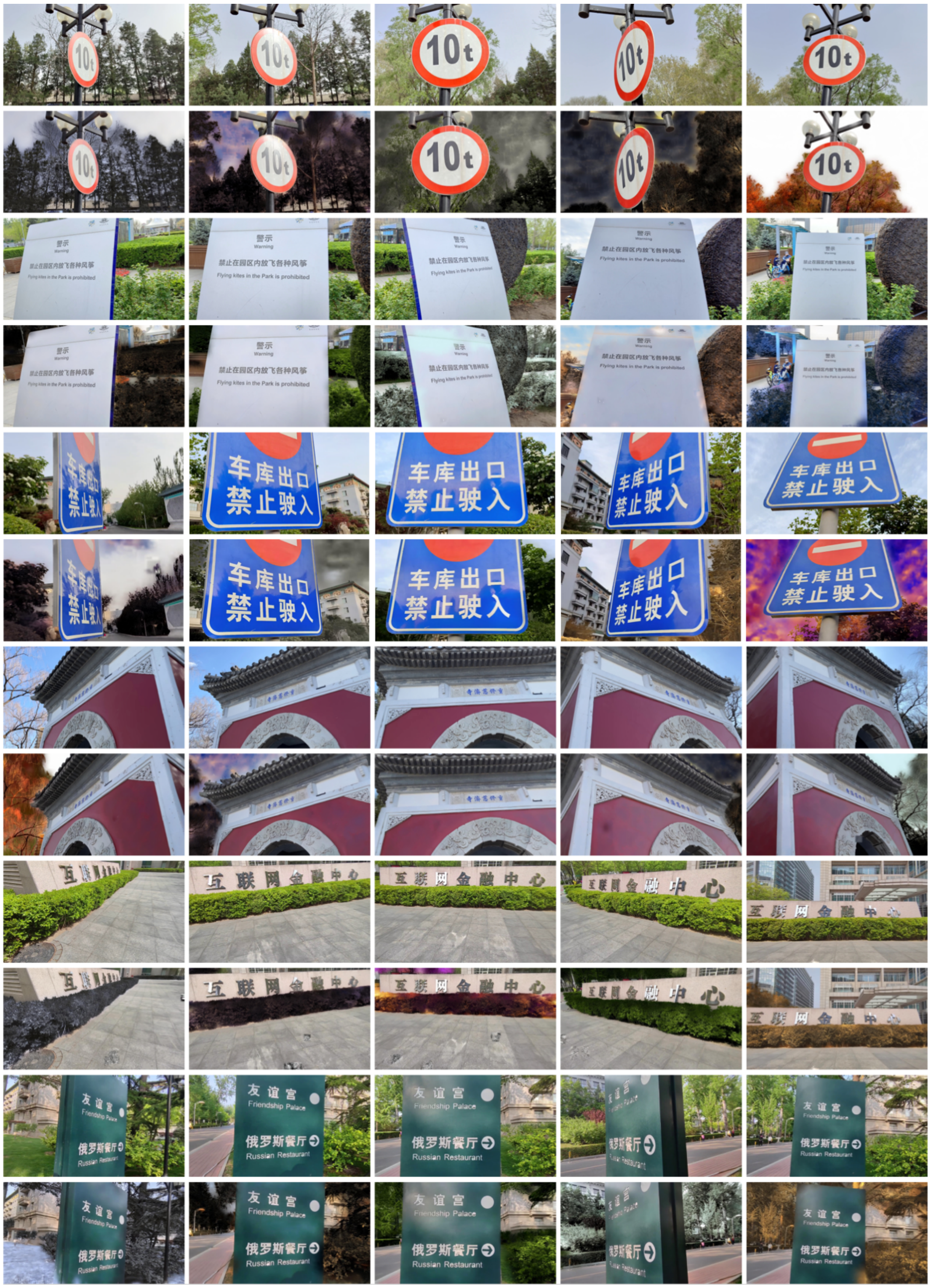


Figure 11. Results of generated scene text images