GO-NeRF: Generating Virtual Objects in Neural Radiance Fields

Peng Dai[1] Feitong Tan[2] Xin Yu[1] Yinda Zhang[2] Xiaojuan Qi[1]
[1]The University of Hong Kong [2] Google

Abstract

Despite advances in 3D generation, the direct creation of 3D objects within an existing 3D scene represented as NeRF remains underexplored. This process requires not only high-quality 3D object generation but also seamless composition of the generated 3D content into the existing NeRF. To this end, we propose a new method, GO-NeRF, capable of utilizing scene context for high-quality and harmonious 3D object generation within an existing NeRF. Our method employs a compositional rendering formulation that allows the generated 3D objects to be seamlessly composited into the scene utilizing learned 3D-aware opacity maps without introducing unintended scene modification. Moreover, we also develop tailored optimization objectives and training strategies to enhance the model's ability to exploit scene context and mitigate artifacts, such as floaters, originating from 3D object generation within a scene. Extensive experiments on both feed-forward and 360^o scenes show the superior performance of our proposed GO-NeRF in generating objects harmoniously composited with surrounding scenes and synthesizing high-quality novel view images. Project page at https://daipengwa.github.io/GO-NeRF/.

1. Introduction

In recent years, tremendous progress has been made for re-renderable real-world environment reconstruction using the neural radiance field (NeRF) [3–5, 7, 23, 27, 40, 43]. Concurrently, text-guided object generation [13, 18, 20, 29, 39, 44] has demonstrated significant potential in creating novel 3D contents. In this work, we investigate a novel problem: generate 3D objects that harmonize with given 3D real-world scenes. Such capability is critical for novel scene creation and editing, demanding the seamless composition of generated content into the environment, and ensuring a highly immersive experience in downstream applications.

Recently, Gordon *et al*. [8] leveraged a CLIP-based text-image matching loss for 3D object generation and introduced a 3D blending pipeline for compositing synthesized 3D objects into a NeRF. However, the approach is limited by the model's generative capacity and the lack of utilizing scene contextual information, resulting in suboptimal, low-quality outcomes that do not blend seamlessly with the NeRF (see Fig. 3 second row: the fruits are flying in the sky). On the other hand, text-guided image inpainting models can create harmonized scenes with desired objects by filling specified masks [24, 25]. However, generating view-consistent images with synthesized objects for subsequent NeRF model training [24, 25] remains challenging. Consequently, these techniques are susceptible to large view changes and unintended scene content modifications due to inaccurate inpainting masks (see Fig. 3 bottom right: the given mask does not match the object's silhouette).

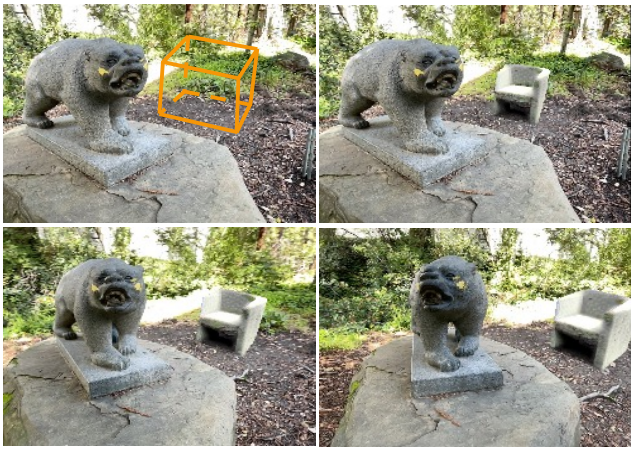

Figure 1. Given a 3D bounding box and textual prompts describing virtual objects, we aim to generate virtual objects directly into pre-trained neural radiance fields and require the results to coordinate seamlessly with the surrounding scene. We show the generated stone chair observed from different camera views.

In this paper, we introduce a novel pipeline with an easy-to-use interface, namely GO-NeRF, which generates text-prompt-controlled 3D virtual objects at user-specified locations within a given NeRF-based environment, resulting in a harmonized 3D scene (see Fig. 1, 9, 3, where shallow shadows and reflections can be observed). Our approach is grounded in two key aspects: (1) a compositional rendering formulation that facilitates seamless composition of generated 3D objects into an existing scene while prevent-

ing unintended scene modifications, and (2) meticulously designed context-aware learning objectives that utilize 2D inpainting priors to optimize the 3D objects, ensuring their high quality and seamless fusion with the scene context.

Specifically, we design a user-friendly interface that, given a NeRF model, allows users to determine the desired 3D location for object generation by selecting three points from a rendered image (see Fig. 2) and utilizing depth information to map them into a 3D box. Then, for virtual object generation in the scene, we additionally create a new object NeRF in the 3D box and render it separately from the existing scene NeRF. We optimize the occupancy values of query points within the identified 3D box and associate them with the existing scene to compute a 3D-aware opacity map, which is used to composite the rendered object into the existing scene. Such separation and 3D-aware composition preserve unchanged scene content beyond the desired editing space, handle occlusions, and enable compatibility with any existing NeRFs of arbitrary representations (e.g., InstantNGP [26] and NeRF [23].).

Further, to generate a 3D object according to a text description and ensure compatibility with the scene context, we distill 2D text-guided image inpainting priors from diffusion models [34] using score distillation sampling (SDS) [29]. The inpainting prior allows us to better leverage scene context from the existing NeRF, facilitating the synthesis of scene-compatible objects and their surroundings (e.g., shadows). However, results obtained using SDS have the over-saturated problem (see Fig. 5). To address this, we introduce a regularizer to harmonize the saturation channel of synthesized objects in HSV color space with the scene's overall saturation tone. Moreover, to remove floater artifacts caused by compositing the object and scene for optimization (see Fig. 6), we randomly replace the scene context as a random background with a certain ratio for optimizing objects without distractions from the scene context. Lastly, we also propose a reference-guided feature space loss, enabling users to control the generated style using a reference from the scene context or new images.

To demonstrate the effectiveness of our proposed method, we perform extensive experiments on public datasets encompassing both feed-forward and 360° scenes [4, 10, 22], where our approach exhibits superior performance in both quantitative and qualitative evaluations. In summary, our primary contributions are as follows:

- We introduce GO-NeRF, a novel pipeline that enables the generation of context-compatible 3D virtual objects from text prompts and seamlessly composites them into a pre-trained neural radiance field while preserving unchanged scene context and maintaining compatibility with arbitrary NeRF representations for existing scenes.
- We develop learning objectives and regularizers that take into account scene context, enabling high-quality, floater-free 3D synthesis and scene-harmonious composition to create new 3D scenes.
- Experimental results showcase our approach's ability to leverage scene context for virtual object generation, outperforming previous methods on both feed-forward and 360° datasets.

2. Related Work

Neural radiance field editing. NeRF [23] is primarily designed for novel view synthesis and has attracted significant attention due to its efficacy in reproducing our world [3–7, 26, 30, 40]. Considering the editing needs in NeRF, early works [19, 38] attempted to modify the appearance and geometry of object-level NeRF using latent codes extracted from text prompts or RGB images. After this, Huang *et al*. [12] proposed to stylize NeRF using pre-stylized 2D images for NeRF fine-tuning. Kobayashi *et al*. [15] made the editing semantic-driven by distilling semantic features into NeRF. To enable NeRF-based animation editing, recent works [2, 41, 45] distilled NeRF onto the mesh and used it to animate the NeRF. Instead of editing the appearance and pose of content already in NeRF, Mirzaei *et al*. [24, 25] proposed the inpainting NeRF task, which substituted regions of interest (ROI) of the NeRF with new content. Specifically, they adopted a 2D image inpainting model [35] to fill masked regions. Subsequently, the inpainted content was distilled into NeRF by using inpainted images for fine-tuning. Although impressive results are displayed, their approach is unsuitable for scenes with large view changes. Unlike previous methods, our goal is to generate virtual objects directly in NeRFs and not be limited by view changes.

3D generation. 3D content is highly desirable for a variety of applications, but difficult to obtain. In the past, most high-quality 3D models were manually created by experienced persons, however, this creation process is still challenging and time-consuming. Recently, Dreamfuison [29] proposed a text-guided 3D generation framework that optimized an object NeRF using the novel Score Distillation Sampling (SDS) loss, which distills the generation capability of 2D diffusion model [34] into 3D generation. This method greatly reduced consumption in high-quality 3D content creation and inspired numerous subsequent works [16–18, 32, 37, 39, 44]. Different from previous methods focusing on object-level generation, Ryan *et al*. [28] simultaneously optimized multiple pre-defined 3D bounding boxes with different text prompts and generated a scene containing multiple objects. Moreover, Gordon *et al*. [8] proposed to blend the generated objects into pre-trained NeRFs using a distance-based blending scheme. Even though they successfully generated objects in the NeRF, the results were unrealistic and did not harmonize with the surrounding scene. In this paper, we focus on gen-

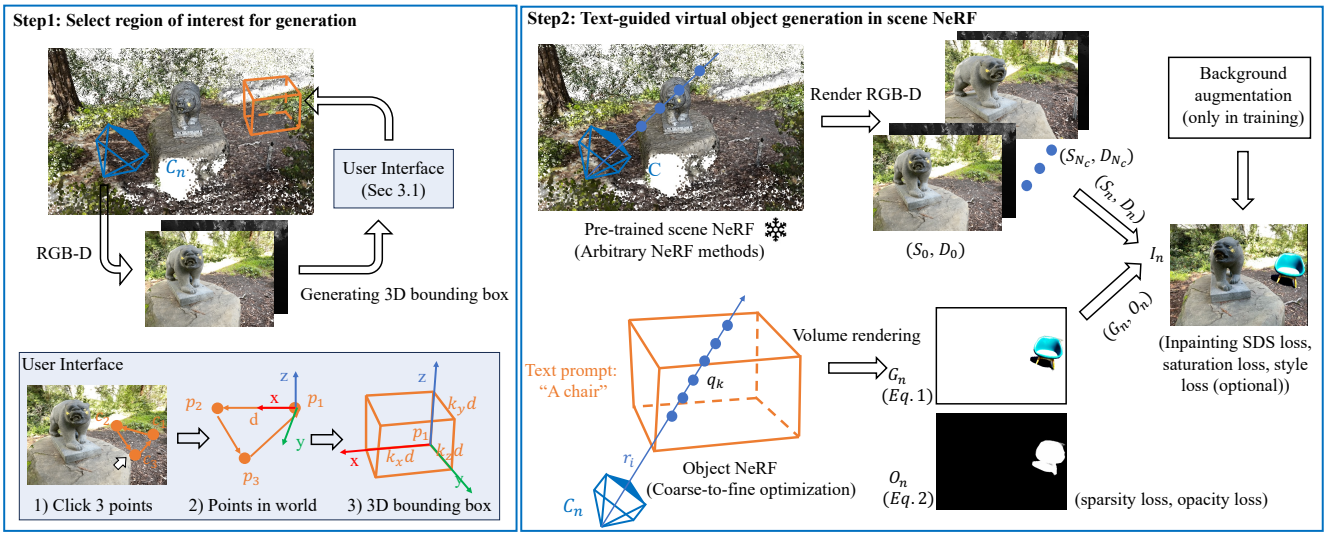

Figure 2. **Our pipeline for generating virtual objects in neural radiance fields.** Left: we provide a user-friendly interface for selecting generation regions in the pre-trained scene NeRF. Specifically, users can simply click three points on the image plane, then our interface will return a 3D bounding box in the scene by employing perspective projection and cross-product operations. Right: we separate scene rendering (up) and object generation (down), then composite them in NeRF's output space. The scene rendering renders scene NeRF into RGB-D images (S, D) using pre-defined cameras C. The object generation optimizes a neural radiance field within the 3D box and produces RGB images S_n (Eq. (1)) and opacity maps O_n (Eq. (2)) through volume rendering. Subsequently, the scene and generated content are blended with the help of optimized opacity maps to produce the final output I_n. During optimization, we design loss functions and training strategies to ensure high-quality and scene-harmonic results.

erating high-quality and scene-harmonic virtual objects in NeRFs while taking scene context into account.

3. Method

Overview. This section presents our pipeline for generating a virtual object within a given 3D NeRF-based scene from an input text prompt. It begins with creating a 3D bounding box to define the modification region. This process can be automated by clicking three points on the rendered images using our interface (Sec. 3.1). Next, we introduce a compositional rendering pipeline (Sec. 3.2) for object generation in the scene radiance field. Specifically, we render the scene radiance field into RGB-D images from predefined perspectives; the novel object is generated by optimizing the object NeRF in the 3D box. To obtain final renderings, the object and scene are composited conditioning on pre-rendered RGB-D images and optimized opacity maps. Finally, we describe our elaborately designed loss functions and training strategies (Sec. 3.3), which guide the optimization process and ensure harmonious object generation in the scene. An overview of our method is shown in Fig. 2.

3.1. Interface

We create a simple and intuitive interface that enables users to effortlessly define the position of objects in the 3D scene for generation. This tool eliminates the need for complex 3D user interfaces by allowing users to conveniently click on 2D images to automatically generate corresponding 3D bounding boxes, as illustrated in Fig. 2 left. To achieve this, we begin by clicking three points $\{c_1, c_2, c_3\}$ and back-projecting these three points onto the 3D scene, resulting in points $\{p_1, p_2, p_3\}$. A plane P can be constructed using these points. Next, we define the 3D bounding box's coordinate system by setting the x axis as $p_1 - p_2$ and we set the z axis to be perpendicular to plane P, y axis can be computed using cross-product operations. Finally, the size of each 3D bounding box (i.e., length of each axis) is manually set as a ratio $\{k_x, k_y, k_z\}$ of the distance d between p_1 and p_2. Note that the 3D bounding box does not exactly match the object's shape. Previous work [24] generates the object within a 3D region relying on 2D inpainting, which introduces unintended scene modifications. Instead, we consider this 3D box as an initialization and adaptively optimize it to fit the desired object in Sec. 3.2.

3.2. Compositional Rendering

As illustrated in the right panel of Fig. 2, a separate NeRF is introduced to represent the object, parameterized by θ. During training, the generation process is to learn these parameters to synthesize the object in accordance with the input text prompt and the scene context. This learning is achieved by optimizing the rendering output generated by combining the contributions from both the scene and object.

For scene rendering, we render RGB-D images from the

pre-trained scene NeRF. Note that to alleviate the computational burden associated with repeated volume rendering during training, we eliminate the need to query and render the unchanged content of the large-scale scene at each iteration by pre-rendering the scene into a collection of RGB-D image sequences $\{(S_0, D_0), ..., (S_{N_c}, D_{N_c})\}$ from a set of predefined N_c camera views $\{C_0, ..., C_{N_c}\}$. In practice, we utilize the same camera views employed for training the scene NeRF.

For object generation, we render RGB images $\{G_0, ..., G_{N_c}\}$ and opacity maps $\{O_0, ..., O_{N_c}\}$ from the object NeRF in the 3D bounding box, whose representation can differ from the scene representation. Specifically, we cast rays from a camera C_n and sample query points inside the 3D box. The corresponding RGB value $G_n(i)$ and opacity value $O_n(i)$ of a ray r_i are volume rendered following Eq. (1) and Eq. (2):

$$\begin{aligned} \tau_k &= \exp(-\sum_{t=1}^{k-1} \delta_t \Delta_t), \\ G_n(i) &= \sum_{k=1}^{K} \tau_k (1 - \exp(-\delta_k \Delta_k)) c_k; \end{aligned} \tag{1}$$

$$O_n(i) = \begin{cases} \sum_{k=1}^{K} \tau_k (1 - \exp(-\delta_k \Delta_k)), \\ 0, \quad if \quad r_i \notin \text{box} | D_{\text{box}}(i) > D_n(i). \end{cases} \tag{2}$$

The δ_k and c_k are volume density and RGB values at query point q_k, and Δ_k represents the distance between two adjacent query points along the ray. Additionally, we set the opacity value as 0, which means using the original scene content (Eq. 3), when the ray has no intersection with the 3D box ($r_i \notin$ box) or query points are occluded by the scene foreground ($D_{\text{box}}(i) > D_n(i)$).

To render the final output I_n, we leverage the opacity map O_n to seamlessly composite the object and scene renderings following Eq. 3:

$$I_n = G_n \cdot O_n + (1 - O_n) \cdot S_n. \tag{3}$$

Since the optimized opacity map can precisely identify regions affected by the generated objects (i.e., white regions in Fig. 2 bottom right), using it for guiding composition is beneficial for unchanged scene context preservation. Moreover, the composition process occurs on the outputs of NeRF, which means that our rendering pipeline is not affected by NeRF methods used for scene pre-training, making our rendering pipeline a plug-and-play and compatible with NeRF of arbitrary representations.

3.3. Optimization

In this part, we first describe loss objectives for scene-harmonic object generation, followed by introducing optimization strategies that improve the generation quality. To simplify the discussion of the loss, unless specifically stated, we omit the subscript n from $\{I_n, G_n, O_n, S_n\}$.

Inpainting SDS loss. To generate a 3D object, we employ a pre-trained 2D diffusion model, denoted as ϵ_ϕ, to provide generative priors. We optimize our 3D representation by supervising 2D images I rendered from our 3D representation through score distillation sampling (SDS). However, direct utilization of a text-to-image model similar to DreamFusion [29] can not ensure harmony between the generated object and background scenes. To this end, we propose replacing the pure text-to-image generative model with a diffusion-based inpainting model [34] for score distillation. Specifically, given mask M and masked image I_M, our SDS loss from the inpainting-based diffusion model is defined as follows:

$$\nabla_\Theta \mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\epsilon,C} \left[\omega(t) \cdot [\epsilon_\phi(I_t; y, I_M, M, t) - \epsilon] \cdot \frac{\partial I}{\partial \Theta} \right], \tag{4}$$

where t represents the time-step randomly sampled within the diffusion process, ϵ is a randomly sampled Gaussian noise, and the mask M is derived by projecting the 3D box into a corresponding camera view. I_t is the noise-perturbed image rendered from our 3D scenes. This objective function encourages the rendered images to reside in high-density areas [29], conditioned on both the text prompt y and the scene information (provided by M and I_M). Consequently, it ensures the harmonious composition of the optimized object NeRF into the environment.

Geometry loss. Following [13], we adopt the sparsity loss and opacity loss to assist the object's geometry optimization. 1) The sparsity loss requires the rendered opacity map to be sparse following Eq. 5:

$$\mathcal{L}_{\text{S}} = \frac{1}{N} \sum_{i=1}^{N} O(i), \tag{5}$$

where N is the number of rays intersecting with the 3D box. By doing this, the generated object will be concrete thus suppressing floaters. 2) The opacity loss aims to avoid translucent effects by encouraging opacity values to be 0 or 1 using Eq. (6).

$$\begin{aligned} \mathcal{L}_{\text{O}} = -\frac{1}{N} \sum_{i=1}^{N} & O(i) \cdot \log(O(i)) \\ & + (1 - O(i)) \cdot \log(1 - O(i)). \end{aligned} \tag{6}$$

Saturation loss. While the SDS loss boosts the generation of the 3D object, it suffers from color over-saturation issues [39], hindering the seamless composition of generated object NeRFs into the existing scene. To alleviate this, we introduce a reference-guided saturation loss. Specifically, we regulate the saturation values of generated content

within the HSV space as defined in Eq. 7:

$$\mathcal{L}_{\text{SAT}} = \left(\overline{G}_s - \overline{R}_s\right)^2 + \left(\hat{G}_s - \hat{R}_s\right)^2, \tag{7}$$

where $\overline{G}_s$ and $\hat{G}_s$ denote the mean and variance of the saturation values for the generated content, masked using the opacity map O. Similarly, $\overline{R}_s$ and $\hat{R}_s$ represent the mean and variance of saturation values for the reference image R. Unless explicitly specified, we employ the scene image as the reference ($R := S$) for computing the saturation loss.

Style loss. We further incorporate a style loss to improve the color and style coherence between generated objects and a given reference, enabling users to specify the desired style of a synthesized object using a reference image. Unlike the saturation loss which solely focuses on addressing the over-saturation issues, this loss captures the feature-level information from the reference to constrain the generated object:

$$\begin{aligned}
\mathcal{L}_{\text{STY}}^{\text{global}} &= \left(\overline{G}_{\text{vgg}} - \overline{R}_{\text{vgg}}\right)^2 + \left(\hat{G}_{\text{vgg}} - \hat{R}_{\text{vgg}}\right)^2, \\
\mathcal{L}_{\text{STY}}^{\text{local}} &= \frac{1}{N}\sum_{i=1}^{N} \min_{i'} \left(G_{\text{vgg}}(i) - R_{\text{vgg}}(i')\right)^2, \\
\mathcal{L}_{\text{STY}} &= \mathcal{L}_{\text{STY}}^{\text{global}} + \mathcal{L}_{\text{STY}}^{\text{local}}.
\end{aligned} \tag{8}$$

This style loss $\mathcal{L}_{\text{STY}}$ in Eq. 8 consists of $\mathcal{L}_{\text{STY}}^{\text{global}}$ and $\mathcal{L}_{\text{STY}}^{\text{local}}$. Similar to saturation loss, the $\mathcal{L}_{\text{STY}}^{\text{global}}$ calculates the statistical loss in VGG feature space [14], and the $\mathcal{L}_{\text{STY}}^{\text{local}}$ is a contextual loss [21] that searches the closest feature for measuring the difference.

Overall loss. Finally, the overall loss Eq. 9 is formulated as a weighted combination of all loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{SDS}} + \lambda_{\text{S}} \cdot \mathcal{L}_{\text{S}} + \lambda_{\text{O}} \cdot \mathcal{L}_{\text{O}} + \lambda_{\text{R}} \cdot \mathcal{L}_{\text{SAT or STY}}. \tag{9}$$

Note that the generated shadow regions are excluded using an empirically defined intensity threshold (< 0.2) and are not used for calculating saturation or style loss.

Coarse-to-fine optimization. Our object NeRF adopts the hash grid representation for efficient 3D content generation [26]. Instead of optimizing a high-resolution hash grid at the beginning, which tends to overfit training views, we start with a low-resolution hash grid and gradually increase its resolution to facilitate generating concrete objects with view-consistent renderings.

Background augmentation. Since our method composites the generated content G_n and scene context S_n for optimization, the generated content usually contains floaters that look similar to the scene background (Fig. 6 first column), making them difficult to observe and remove. To solve this, we augment the background with pure white or black during optimization, where floaters will be significant and easy to remove. It works collaboratively with the sparsity loss (Eq. 5) to generate objects with a clean background (see Fig. 6).

4. Experiments

4.1. Implementation Details

Network and training. To obtain pre-trained scene neural radiance fields and valid the compatibility of different NeRF methods, we use the PyTorch implementation of NeRF [42] for feed-forward scenes, and NeRFstudio [36] for 360^o scenes. Following the 3D object generation pipeline [9, 29], we optimize a hash grid [26] in the 3D bounding box for object generation. During training, λ_S and λ_O are determined using a cosine schedule with values ranging from 30 to 300, and λ_R is set as 500 or 1000. We optimize our generation model on a single Nvidia 3090 GPU for a total of 20000 iterations and 30% iterations are trained with background augmentation.

Datasets. We conduct experiments on the publicly available datasets, including both feed-forward scenes and 360^o scenes from LLFF [22], Instruct-NeRF2NeRF [10], and Mip-NeRF 360^o [4]. The specific scenes we use are "Bear", "Garden", "Benchflower", and "Pond".

Baselines. *1) Manual placement.* We manually place the virtual object, generated using SDS loss, into the scene. *2) Blended-NeRF* [8]. It optimizes an object NeRF within a 3D bounding box using CLIP-based loss [31]. Subsequently, the generated object is blended into the scene NeRF based on its distance to the box center. Note that there is no scene context used for guiding optimization and blending. *3) Spin-NeRF** [25]. To modify the ROI of scene NeRF, it employs a 2D image inpainting model, i.e., LaMa [35], to fill masked regions in image space. Subsequently, the scene NeRF is fine-tuned using inpainted images. Here, we replace the LaMa [35] with the stable diffusion inpainting model [34] to make this process text-guided and obtain multi-view inpainted images by gradually warping and filling the first inpainted image [24].

4.2. Qualitative Comparisons

We compare our method with baselines on both feed-forward and 360^o scenes. To validate that the scene context has an influence on the generation process, we additionally stylized the feed-forward scene into an oil painting style using ARF [46] but experimented with the same text prompt. The comparison results are displayed in Fig. 3, where the first row visualizes the 3D bounding box and its corresponding mask region in the 2D image plane.

Without using the scene context for guidance, the generated objects of Blened-NeRF are not harmonized with the

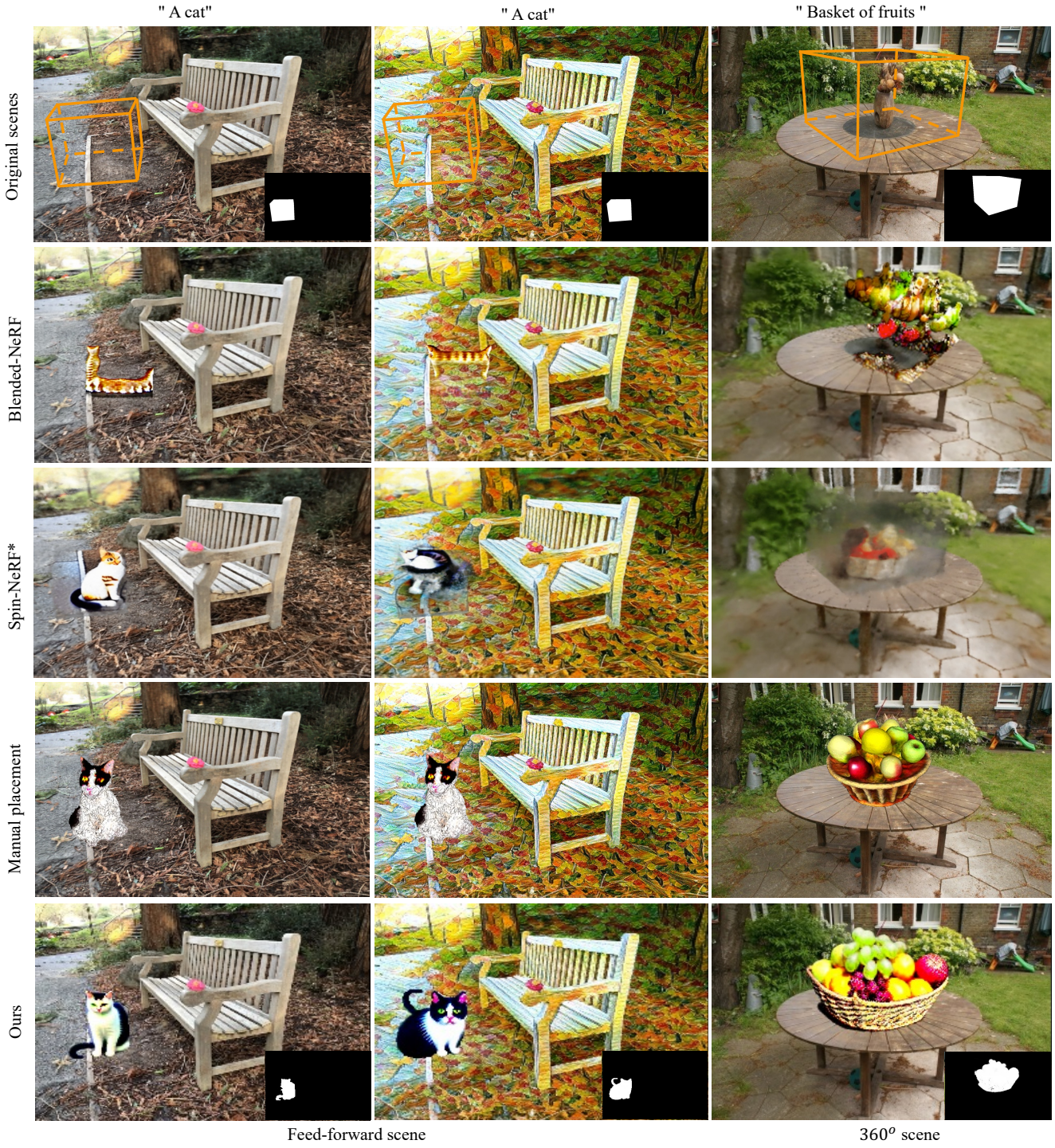

Figure 3. **Qualitative comparisons**. We compare our method with other baselines on feed-forward and 360^o scenes. We show the 3D box and corresponding 2D mask in the first row, and other rows display the results of different methods. Blended-NeRF tends to produce unrealistic and disharmonious results, such as fruits flying in the sky. Spin-NeRF* failed in stylized scenes and 360^o scenes with large view changes. Similarly, manual placement is tedious and also has the problem of utilizing scene context. Our method performs well in all scenes, generating cats with different styles and fruits on the table with shadows to promote harmony. At the bottom right of the last row, we visualize the optimized opacity maps that accurately describe the silhouette of generated content.

scene (Fig. 3 second row). For example, the fruits are flying in the sky instead of being on the table. Moreover, its quality remains low due to the limited capability of the model using CLIP-based loss. By leveraging the 2D inpainting model [34] to fill the masked region (first-row bottom right), the Spin-NeRF* succeeds in utilizing the scene context and generates reasonable results (Fig. 3 third row), but still has the following weaknesses: 1) the inpainting-based scheme inevitably destroy scene content (e.g., the extended white line) due to mismatch between mask and object silhouette; 2) The rendering fidelity heavily relies on the quality of inpainted images which are sensitive to estimated depth [33], thus this method is not suitable for scenes with large view changes or unsuitable for depth estimation. This is demonstrated in stylized and 360^o scenes where Spin-NeRF*' results are bad. Similarly, generating virtual objects first and then manually placing them into the scene is a tedious process requiring users to have experience in 3D software operation, and there is also the problem of ignoring the scene context. Moreover, extracting maneuverable object mesh from object NeRF for placement usually deteriorates the object quality, such as the cat in Fig. 3 fourth row.

Our method can leverage scene context for object generation and achieve good performance on both feed-forward and 360^o scenes. As shown in Fig. 3 last row, the generated cat has a different style in different scenes where the cat is more like a painting in the stylized scene. The generated fruits are correctly placed on the table and the shallow shadow around the base of the basket makes it scene-harmonic. When compared with manual placement, our results are more photo-realistic, such as the cat and fruits. We suspect that the photo-realistic scene context has influences on generation reality. Additionally, our approach composites generated objects into the scene employing optimized opacity maps (last-row bottom right), which accurately describe the silhouette of generated objects and thus preserve scene context unintended to be modified. Please refer to the supplementary material for video results.

4.3. Quantitative Comparisons

We use CLIP score [11] to measure the match between generated objects and given text prompts, and report different methods' average CLIP scores for three scenes (Fig. 3) in Table. 1. To obtain the CLIP score, we randomly render 10 views for each scene and remove the background effects by cropping the bounding box region in the image plane (Fig. 3 first-row bottom right) for assessment. From Table. 1, the average CLIP score of our proposed GO-NeRF is 74.6 significantly outperforms other baselines by at least 20%.

4.4. Ablation Study

Inpainting SDS loss. To validate the efficacy of inpainting SDS loss, we conduct experiments using the original SDS loss. From the results in Fig. 4, the inpainting SDS loss generates a cat with a complete body and proper pose, while the SDS only generates the head of a cat that does not harmonize with the scene. In other words, the inpainting SDS loss can better utilize scene context for generation.

	Blended-NeRF [8]	Spin-NeRF* [25]	GO-NeRF
CLIP score [11] ↑	63.0	60.8	**74.6**

Table 1. **Quantitative comparisons.** This table shows CLIP scores indicating the match between generated content and texts.

Figure 4. **The efficacy of inpainting SDS loss**. Compared with the original SDS loss, the inpainting SDS loss can better utilize the scene context and generate a cat with the proper shape and pose.

Figure 5. **The efficacy of saturation loss**. We show RGB images and corresponding saturation values with and without using the saturation loss. Without constraining the saturation value (bottom left), the generated objects tend to be over-saturated.

Saturation loss. In Fig. 5, we compare results with and without using saturation loss. Without using saturation loss, the generated objects tend to be over-saturated like the backpack in the scene, more clear in the visualized saturation channel in HSV color space. After regularizing the saturation value to be close to the scene saturation, the generated objects become harmonious with the scene.

Sparsity loss and background augmentation. Simply compositing the generated object with the scene for optimization tends to introduce floaters having a similar appearance to the scene background, as shown in Fig. 6 baseline results, where the green floaters are difficult to recognize in the scene with trees and grass as background. By introducing the sparsity loss that encourages the generated object to be concrete, the floaters are reduced to a certain extent but still present (Fig. 6 second column). To further

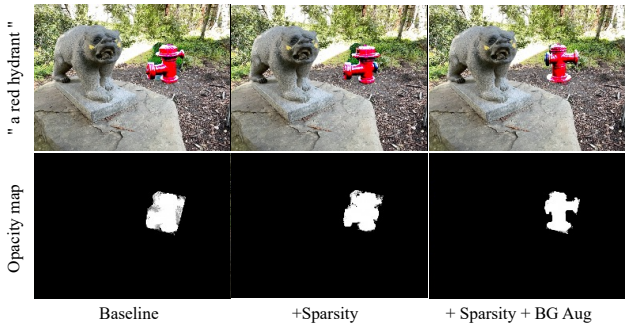

Figure 6. **Sparsity loss and background augmentation.** The floaters appear similar to the scene background, making them difficult to remove. The best results are obtained while using both sparsity loss and background augmentation. The opacity maps in the second row provide a better view.

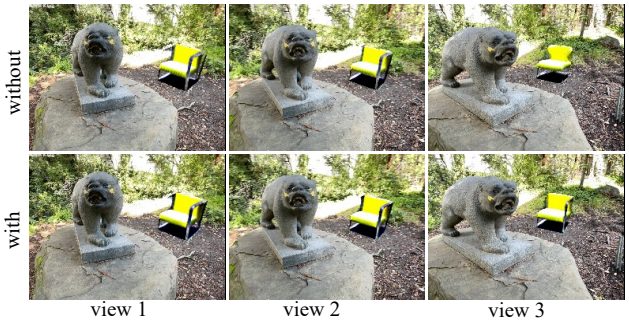

Figure 7. **Coarse-to-fine optimization.** Using coarse-to-fine optimization benefits the view consistency. Please note the shape of the chair across different camera views.

remove floaters, we augment the background with pure white or black during training to make floaters obvious. Eventually, the results are clean as displayed in Fig. 6 last column. The corresponding opacity map is visualized in Fig. 6 second row, where floaters can be clearly observed in the baseline results.

Coarse-to-fine optimization. Instead of gradually increasing the hash grid resolution, we start training with a high-resolution hash grid and show results in Fig. 7 first row. From the results, the generated chair is not consistent across different views because the capability of the high-resolution hash grid is too strong to overfit predefined camera views. On the contrary, training using a coarse-to-fine scheme benefits the view consistency as shown in Fig. 7 bottom.

4.5. Additional Experiments

Style adaptation. In addition to text prompts, we also use reference images as conditions. As shown in Fig. 8 left, the reference image is a stone that is used to guide the generation of "a stone chair", and the corresponding result is shown in Fig. 8 right. Compared with results without using reference (Fig. 8 middle), the generated chair has a closer style to the reference image. Note that the generated chair has its own illumination distribution, such as low-light regions under the chair, which does not exactly follow the reference image. More results can be found in the supplementary material.

Figure 8. **Style adaptation.** With reference-guided generation, the virtual object has a closer appearance to the reference image.

Figure 9. **Reflective surface.** For challenging scenes, such as the reflective water surface, our method can generate wooden boats with reflections to a certain extent.

Reflective surface. We conduct experiments on more challenging scenarios, such as the reflective water surface. We tried to generate a wooden boat in the pond shown in Fig. 9 left. Thanks to the guidance from the stale diffusion model which is pre-trained on large-scale datasets, our method successfully generated wooden boats with reflections in the water, shown in Fig. 9 right. However, the reflection is not perfectly complete yet. We infer that the 3D bounding box's range limits the optimization of reflections because the exact region belonging to the reflection is uncertain. In other words, the reflections may appear outside the 3D box.

5. Conclusion

This paper presents GO-NeRF, a novel method that takes a step forward to directly generate text-controlled 3D objects in an existing scene-level NeRF. To achieve this goal, we employ a compositional rendering formulation associated with tailored optimization objectives and training strategies for synthesizing 3D objects that are seamlessly composited into existing scenes. Our approach leverages image priors from pre-trained text-guided image inpainting networks to facilitate the harmonious generation of objects and their surroundings. Experimental results show the superiority of our approach in the feed-forward and 360^o datasets. We hope our investigation will inspire further work in this domain.

References

[1] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *ACM Transactions on Graphics (TOG)*, 42 (4):1–11, 2023. 11

[2] Ziqian Bai, Feitong Tan, Zeng Huang, Kripasindhu Sarkar, Danhang Tang, Di Qiu, Abhimitra Meka, Ruofei Du, Mingsong Dou, Sergio Orts-Escolano, et al. Learning personalized high quality volumetric head avatars from monocular rgb videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16890–16900, 2023. 2

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 1, 2

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2, 5

[5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *arXiv preprint arXiv:2304.06706*, 2023. 1

[6] Peng Dai, Yinda Zhang, Xin Yu, Xiaoyang Lyu, and Xiaojuan Qi. Hybrid neural rendering for large-scale scenes with motion blur. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 154–164, 2023.

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 1, 2

[8] Ori Gordon, Omri Avrahami, and Dani Lischinski. Blended-nerf: Zero-shot object generation and blending in existing neural radiance fields. *arXiv preprint arXiv:2306.12760*, 2023. 1, 2, 5, 7

[9] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. https://github.com/threestudio-project/threestudio, 2023. 5

[10] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. *arXiv preprint arXiv:2303.12789*, 2023. 2, 5

[11] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 7

[12] Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. Stylizednerf: consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18342–18352, 2022. 2

[13] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022. 1, 4

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016. 5

[15] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems*, 35:23311–23330, 2022. 2

[16] Nikos Kolotouros, Thiemo Alldieck, Andrei Zanfir, Eduard Gabriel Bazavan, Mihai Fieraru, and Cristian Sminchisescu. Dreamhuman: Animatable 3d avatars from text. *arXiv preprint arXiv:2306.09329*, 2023. 2

[17] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023.

[18] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 1, 2

[19] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5773–5783, 2021. 2

[20] Zhengzhe Liu, Peng Dai, Ruihui Li, Xiaojuan Qi, and Chi-Wing Fu. Iss: Image as stetting stone for text-guided 3d shape generation. *arXiv preprint arXiv:2209.04145*, 2022. 1

[21] Roey Mechrez, Itamar Talmi, and Lihi Zelnik-Manor. The contextual loss for image transformation with non-aligned data. In *Proceedings of the European conference on computer vision (ECCV)*, pages 768–783, 2018. 5

[22] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 2, 5

[23] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2

[24] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Marcus A Brubaker, Jonathan Kelly, Alex Levinshtein, Konstantinos G Derpanis, and Igor Gilitschenski. Reference-guided controllable inpainting of neural radiance fields. *arXiv preprint arXiv:2304.09677*, 2023. 1, 2, 3, 5

[25] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor

Gilitschenski, and Alex Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20669–20679, 2023. 1, 2, 5, 7

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2, 5, 11

[27] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 1

[28] Ryan Po and Gordon Wetzstein. Compositional 3d scene generation using locally conditioned diffusion. *arXiv preprint arXiv:2303.12218*, 2023. 2

[29] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 1, 2, 4, 5

[30] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 5

[32] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. Dreambooth3d: Subject-driven text-to-3d generation. *arXiv preprint arXiv:2303.13508*, 2023. 2

[33] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021. 7

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 2, 4, 5, 7, 11

[35] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. *arXiv preprint arXiv:2109.07161*, 2021. 2, 5

[36] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023. 5

[37] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. *arXiv preprint arXiv:2303.14184*, 2023. 2

[38] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 2

[39] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023. 1, 2, 4

[40] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 1, 2

[41] Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision*, pages 597–614. Springer, 2022. 2

[42] Lin Yen-Chen. Nerf-pytorch. https://github.com/yenchenlin/nerf-pytorch/, 2020. 5

[43] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 1

[44] Xin Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Song-Hai Zhang, and Xiaojuan Qi. Text-to-3d with classifier score distillation. *arXiv preprint arXiv:2310.19415*, 2023. 1, 2

[45] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 2

[46] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. 5

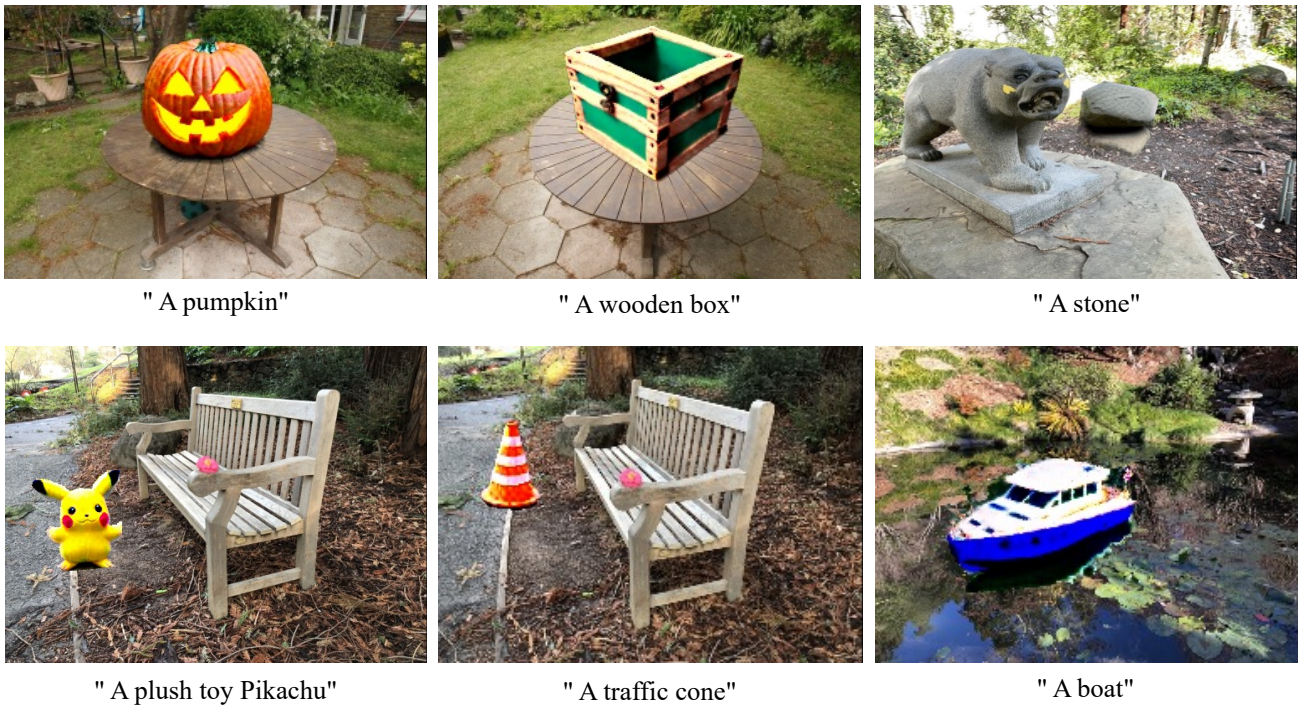

Figure 10. **More results.** We provide more results using different text prompts.

Appendix

In the supplementary file, we provide more implementation details and more results not elaborated in our paper due to the paper length limit:

- Sec. 6: results with more diverse prompts.
- Sec. 7: style adaptation using downloaded images.
- Sec. 8: comparisons with 2D in-painting methods.
- Sec. 9: more implementation details.
- Sec. 10: limitation analysis.

6. More Results

To evaluate the effectiveness of GO-NeRF, we put it to the test using more text prompts, with the results displayed in Fig. 10. GO-NeRF successfully creates vibrant virtual objects within scenes, such as an adorable Pikachu and a Halloween-themed pumpkin. All generated objects possess well-defined shapes and are situated in reasonable poses (e.g., on the ground or a table). Additionally, self-occlusion effects are noticeable; take note of the areas beneath the stone, pumpkin, and Pikachu. For a more comprehensive understanding of the multi-view results, please refer to our video results and the HTML file attached in the supplementary material.

7. Style Adaptation

In the main paper, we present results using a reference image from the scene. In contrast, here, we utilize a reference image sourced from the Internet. As illustrated in Fig. 11, Go-NeRF can leverage user-provided references to generate virtual objects whose texture style aligns with the user-specified references.

8. Applications on Image Inpainting

Our approach can also be applied to 2D image inpainting by optimizing a single view. We compare our method with diffusion-based 2D image inpainting techniques [1, 34] and display the results in Fig. 12. Unlike previous 2D inpainting methods that alter all pixels in masked areas (i.e., the white region in Fig. 12 in the first image), leading to unintended scene changes such as the white line and road surface, our approach progressively optimizes the masked area and only modifies regions impacted by the generated object (i.e., the white region in Fig. 12 in the last image).

9. More Implementation Details

Due to its efficiency, we employ Instant-NGP [26] as the object generation backbone. This paper's multi-level hash grid representation consists of 16 levels with a base resolution of 16. During the coarse-to-fine optimization process, we begin at level 2 and increment the level by one every

Figure 11. **Style adaptation**. The generated object's style follows the reference when using the reference image as guidance.

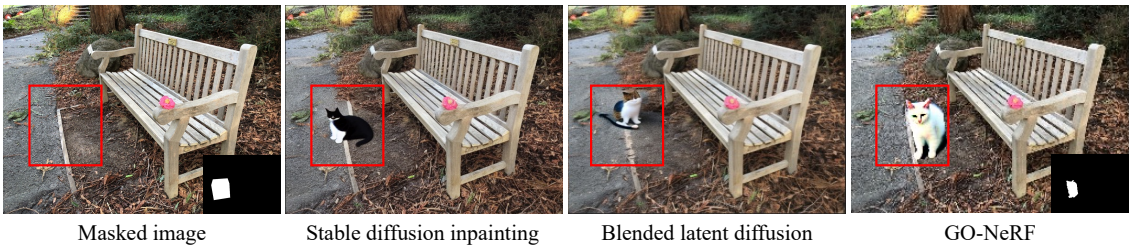

Figure 12. **Compare with 2D image inpainitng.** Our approach generates desired objects in the masked region without introducing unintended scene modifications. Other image inpainting approaches change the scene content, such as the white line and road surface.

1000 steps. Given that the diffusion model is pre-trained on square images, we randomly crop a square patch encompassing the masked region to mitigate the impact of the image aspect ratio when computing the inpainting SDS loss.

10. Limitations

As illustrated in Fig. 9 of the main paper, if the specified 3D box does not encompass the area affected by the object, such as reflections, our approach may not be capable of modifying these areas. In the future, we can investigate dynamic methods for adjusting user-specified boxes. Additionally, since our method fundamentally relies on SDS loss, it may be subject to its limitations, such as the Janus problem.