Blending-NeRF: Text-Driven Localized Editing in Neural Radiance Fields

Hyeonseop Song[1*] Seokhun Choi[1*] Hoseok Do[1] Chul Lee[1] Taehyeong Kim[2†]
[1]AI Lab, CTO Division, LG Electronics, Republic of Korea
[2]Dept. of Biosystems Engineering, Seoul National University, Republic of Korea
{hyeonseop.song, seokhun.choi, hoseok.do, clee.lee}@lge.com taehyeong.kim@snu.ac.kr

Abstract

Text-driven localized editing of 3D objects is particularly difficult as locally mixing the original 3D object with the intended new object and style effects without distorting the object's form is not a straightforward process. To address this issue, we propose a novel NeRF-based model, Blending-NeRF, which consists of two NeRF networks: pretrained NeRF and editable NeRF. Additionally, we introduce new blending operations that allow Blending-NeRF to properly edit target regions which are localized by text. By using a pretrained vision-language aligned model, CLIP, we guide Blending-NeRF to add new objects with varying colors and densities, modify textures, and remove parts of the original object. Our extensive experiments demonstrate that Blending-NeRF produces naturally and locally edited 3D objects from various text prompts.

1. Introduction

3D image synthesis and related technologies are greatly impacting industries such as art, product design, and animation. While recent 3D image synthesis techniques like Neural Radiance Field (NeRF) [22] have opened up new applications for 3D content production [8, 14, 26] at scale, their ability to enable precise and localized editing of object shapes and colors remains a challenge for broader adoption. Often time, a more localized and granular editing of 3D objects, especially attaching or removing certain objects of certain styles, is still difficult and costly in spite of several recent attempts at 3D object editing [4, 18, 21, 35, 40].

Previous attempts, such as EditNeRF [18] and NeRF-Editing [40], only offer limited and non-versatile editing options, while Text2Mesh [21] and TANGO [4] allow only simple texture and shallow shape transformations of entire 3D objects. CLIP-NeRF [35] propose a generative method with disentangled conditional NeRF for object editing but it requires a large volume of training data for the targeted editing category and is hard to edit only the desired part of objects locally. They present an additional approach, fine-tuning a single NeRF per scene with a CLIP-driven objective, which can edit object appearance but not shape well.

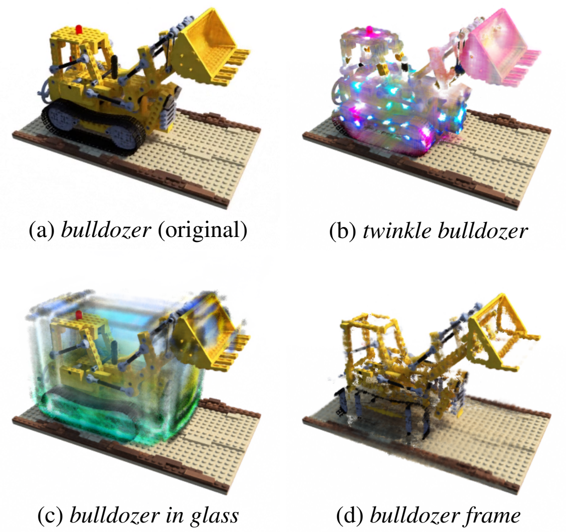
(a) *bulldozer* (original) (b) *twinkle bulldozer*
(c) *bulldozer in glass* (d) *bulldozer frame*

Figure 1. Representative results of text-driven localized object editing using our method. (a) Bulldozer is the original object, and each editing is performed by (b) color change, (c) density addition, and (d) density removal operations.

To achieve effective and practical localized editing of 3D objects by any text prompts at scale, it is necessary to apply style changes to specific portions of the object, including selectively changing color and locally adding and removing densities, as shown in Figure 1. In this study, we propose a novel method for localized object editing that allows modification of 3D objects by text prompts, enabling full stylization including density-based localized editing. We believe that relying on the simple fine-tuning of a single NeRF to generate new densities in the low initial density area or to alter existing densities through a CLIP-driven objective is inadequate for achieving complete stylization of

[*] Equal contribution.
[†] Corresponding author. Partially conducted at LG Electronics.

shapes and colors. Instead, our approach involves parameterizing specific regions in the implicit 3D volumetric representations and blending the original 3D object representation with an *editable NeRF* architecture specifically trained to render the blended image naturally. We use a pretrained vision-language method like CLIPSeg [19] to specify the area to be modified in the text input workflow.

The proposed method is based on a novel layered NeRF architecture, called Blending-NeRF, which includes a *pretrained NeRF* and an *editable NeRF*. There are some studies that employ multiple NeRFs and train them simultaneously to individually reconstruct the static and dynamic components of a dynamic scene [7, 33, 37, 39]. On the other hand, our approach introduces an additional NeRF to facilitate text-based modifications in specific regions of a pretrained static scene. These modifications encompass various editing operations, including color changes, density addition, and density removal. By blending density and color from the two NeRFs, we can achieve fine-grained localized editing of 3D objects. In summary, our contributions include:

- We propose the novel Blending-NeRF architecture that combines a *pretrained NeRF* with an *editable NeRF* using various objectives and training techniques. This approach allows to naturally edit the specific regions of 3D objects while preserving their original appearance.
- We introduce new blending operations that capture the degree of density addition, density removal, and color alteration. Thanks to these blending operations, our method allows for precisely targeting the specific regions for localized editing and constraining the degree of object editing.
- We conduct several experiments involving text-guided 3D object editing, such as editing of shape and color, and compare our approach to previous attempts and their simple extensions, showing that Blending-NeRF is both qualitatively and quantitatively superior.

2. Related Work

Text-Guided 3D Object Generation This task aims to create 3D objects from natural language descriptions. Recent advancements in joint embedding of images and text [28, 41], text-to-image generation [29, 30], and neural rendering [3, 22, 23, 32, 34] have made it possible to generate 3D objects without 3D supervision, using only textual guidance. CLIP-Forge [31] uses an auto-encoder and a contrastive language-image pretraining (CLIP) [28] embedding to generate multiple object geometries for a given text query without paired text and 3D data. It is not that efficient, though, requiring a large unlabeled 3D dataset to train its autoencoder and to learn a latent space for shapes. DreamField [8] optimizes NeRF from multiple camera views to produce high-quality objects so that the CLIP embeddings of the rendered image and target text are similar. It improves the fidelity and visual quality of generated objects using simple geometric priors. DreamFusion [26] uses a pretrained 2D text-to-image diffusion model [30] and NeRF to perform text-to-3D synthesis, while CLIP-Mesh [11] optimizes texture, normal, and vertices position of the mesh using a differentiable renderer and CLIP. Our work utilizes NeRF and CLIP to generate 3D objects but focuses on localized editing of objects based on textual guidance, which is different from previous studies.

3D Object Editing Preserving the original object structure while meeting user intent is crucial in object editing tasks. Liu *et al.* [18] proposed a conditional radiance field that enables color and shape editing by learning disentangled volumetric representation and propagating sparse 2D user scribbles over the 3D region. NeRF-Editing [40] establishes the correspondence between explicit mesh representation and implicit volume representation, allowing for controllable shape deformation such as increasing or decreasing the size of 3D objects. Our method also aims at localized editing but differs in using texts as input and focusing on reshaping and restyling rather than simple modifications [18] or geometric transformations [40].

Text-driven 3D object editing methods have also been studied. Text2Mesh [21] and TANGO [4] edit the style of 3D objects with the supervision of CLIP. Text2Mesh stylizes a 3D mesh by predicting color and local geometries for a given target text prompt. TANGO enables photorealistic 3D style transfer by automatically predicting reflectance effects according to a text prompt without task-specific training. CLIP-NeRF [35] allows for control of global structure and appearance individually by leveraging disentangled latent representations from conditional generative models, but it requires a significant amount of 3D dataset (e.g., 150k chair images [24] that include sofas and wood chairs for training). While CLIP-NeRF also presents a single NeRF-based editing method per scene that can edit an object's color, it has a limitation in that it cannot edit its density well. That is, it fails to achieve satisfying results while editing the shape of a single NeRF by a text prompt. Our approach overcomes such limitations, allowing full stylization to the specific regions as demonstrated in our experiments.

3. Background

3.1. Neural Radiance Field

Neural Radiance Field (NeRF) [22] implicitly represents a 3D scene with a multi-layer perceptron (MLP) which produces a density and a color for a queried ray point sample. Specifically, given a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ passing through a image pixel, depth $t \in [t_{\text{near}}, t_{\text{far}}]$, and camera center $\mathbf{o}$, it takes as input a 3D position $\mathbf{x}$ and viewing direction

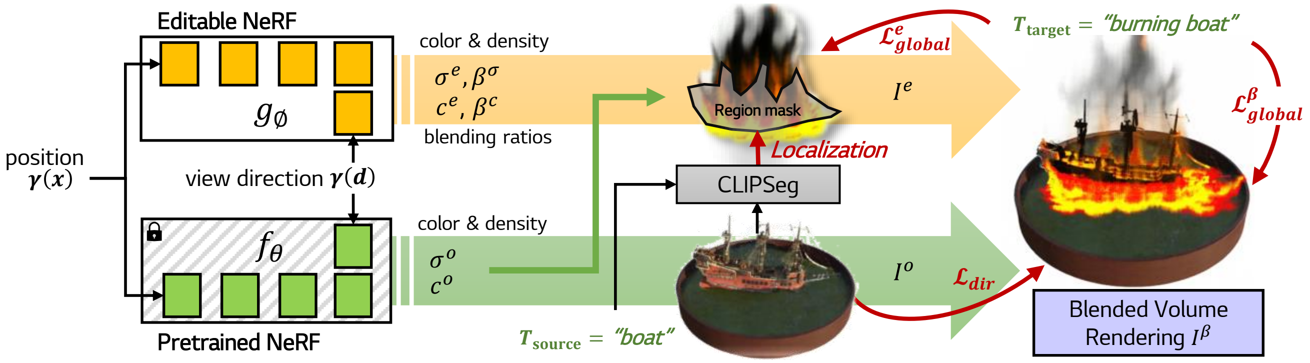

Figure 2. Overall architecture and main objectives. The target editing region is specified by the source text T_{source} in the original rendered image I^o, and the editable NeRF g is trained to render a blended image I^β that matches the target text T_{target}. The CLIP encoders and other localized editing objectives are omitted for simplicity.

$\mathbf{d}$ to produce a density $\sigma^o \in [0, \infty)$ and a color $\mathbf{c}^o \in [0, 1]^3$:

$$(\sigma^o, \mathbf{c}^o) = f_\theta(\gamma(\mathbf{x}), \gamma(\mathbf{d})), \tag{1}$$

where θ is the parameterized network weights, and γ is a positional encoding. Then, NeRF estimates the expected color of a ray by using quadrature with K sampled points:

$$\hat{\mathbf{C}}^o(\mathbf{r}) = \sum_{k=1}^{K} T_k^o \alpha_k^o \mathbf{c}_k^o, \tag{2}$$

where $T_k^o = \prod_{k'=1}^{k-1}(1 - \alpha_{k'}^o)$ is a transmittance [27] with an alpha value $\alpha_k^o = 1 - \exp(-\sigma_k^o \delta_k)$ and $\delta_k = t_{k+1} - t_k$ is a distance between two adjacent sampled points on a ray. An accumulated opacity of a ray also can be estimated as:

$$\hat{E}^o_{\text{acc}}(\mathbf{r}) = \sum_{k=1}^{K} T_k^o \alpha_k^o. \tag{3}$$

3.2. Connecting Text and Images

There have been various studies on vision-text joint representation learning methods [10, 17, 41] including CLIP [28]. CLIP's image and text encoders are pretrained on large dataset to ensure that the representation vectors of image-text pairs match well. Based on these aligned representations, CLIP losses (*i.e.*, global [25] and directional [6] CLIP loss) are widely used in text-guided image editing [1, 2, 6, 12, 25, 35]. The global CLIP loss $\mathcal{L}_{\text{global}}(I, T)$ minimizes the cosine distance between an image I and a text T in the CLIP embedding space:

$$\mathcal{L}_{\text{global}}(I, T) = \mathcal{D}_{\cos}(E_{\text{img}}(I), E_{\text{txt}}(T)), \tag{4}$$

where $E_{\text{img}}(\cdot)$ and $E_{\text{txt}}(\cdot)$ are the image and text encoder of CLIP, and $\mathcal{D}_{\cos}$ is the cosine distance. In another way, the directional CLIP loss $\mathcal{L}_{\text{dir}}$ controls the direction of change for the image embedding vector. This method is known to prevent mode-collapsed problems [6], which is defined as:

$$\mathcal{L}_{\text{dir}}(I_{\text{target}}, T_{\text{target}}, I_{\text{source}}, T_{\text{source}}) = \mathcal{D}_{\cos}(\triangle I, \triangle T), \tag{5}$$

where $\triangle I = E_{\text{img}}(I_{\text{target}}) - E_{\text{img}}(I_{\text{source}})$ and $\triangle T = E_{\text{txt}}(T_{\text{target}}) - E_{\text{txt}}(T_{\text{source}})$. Here, I_{target} and T_{target} are target edited image and its text description, and I_{source} and T_{source} are original image and its text description. We use both CLIP losses for text-driven object editing.

In addition, CLIP is also used for text or image-driven segmentation tasks in a zero-shot manner [5, 16, 19, 36, 38]. We utilize CLIPSeg [19] to get the target image region for a queried text for localized object editing.

4. Method

Our goal is to locally edit the pretrained NeRF model with the natural language as guidance. To this end, we propose Blending-NeRF, which consists of *pretrained NeRF* f_θ for the original 3D model and *editable NeRF* g_ϕ for object editing. The weight parameter θ is frozen, and ϕ is learnable. The edited scene is synthesized by blending the volumetric information of two NeRFs (Section 4.1). We use two kinds of natural language prompts: *source text* and *target text*, describing the original and edited 3D model, respectively. Blending-NeRF performs text-driven editing using the CLIP losses with both prompts (Section 4.2). However, using only the CLIP losses is not sufficient for localized editing as it does not serve to specify the target region. Thus, during training, we specify the editing region in the original rendered scene using the source text. Simultaneously, the editable NeRF is trained to edit the target region under the guidance of localized editing objective (Section 4.3). An overview of the proposed method is depicted in Figure 2. Note that Blending-NeRF is trained in an end-to-end manner.

4.1. Blended Volume Rendering

Editable NeRF The *editable NeRF* extends NeRF to produce two blending ratios $\beta^c \in [0,1]$ and $\beta^\sigma \in [0,1]$ in addition to a density $\sigma^e \in [0,\infty)$ and a color $\mathbf{c}^e \in [0,1]^3$ for seamlessly blending the ray points of two networks:

$$\begin{aligned} (\sigma^e, \mathbf{c}^e, \beta^\sigma, \beta^c) &= g_\phi(\gamma(\mathbf{x}), \gamma(\mathbf{d})) \\ \sigma^{o'} &= (1-\beta^\sigma)\sigma^o \\ \mathbf{c}^{o'} &= (1-\beta^c)\mathbf{c}^o + \beta^c \mathbf{c}^e. \end{aligned} \tag{6}$$

The density blending ratio β^σ determines the amount of density σ^o in the pretrained NeRF that is removed for object modification. Consequently, the modified original density $\sigma^{o'}$ contributes to the dominance of the editable NeRF density σ^e. Similarly, the color blending ratio β^c controls the amount of color $\mathbf{c}^o$ modified in the pretrained NeRF. In this case, to prevent the modified original color $\mathbf{c}^{o'}$ from changing to a specific color (*e.g*., black or white), it is determined by mixing the editable color $\mathbf{c}^e$ by the proportion of β^c. Finally, we get $\sigma^{o'}$, $\mathbf{c}^{o'}$, σ^e, and $\mathbf{c}^e$ to blend the two NeRFs. Using these values, partial addition and removal of density, and change of color are performed on the original scene by the following blending operations.

Blending Operations A previous work [20] introduces a method for augmenting the static part with the transient part of the NeRF outputs on volume rendering to disentangle the static and transient components. Likewise, the blended color $\hat{\mathbf{C}}^\beta(\mathbf{r})$ of a ray can be calculated as:

$$\begin{aligned} \hat{\mathbf{C}}^\beta(\mathbf{r}) &= \sum_{k=1}^{K} T_k^\beta (\alpha_k^{o'} \mathbf{c}_k^{o'} + \alpha_k^e \mathbf{c}_k^e) \\ T_k^\beta &= \prod_{k'=1}^{k-1} (1-\alpha_{k'}^\beta) \\ \alpha_k^\beta &= 1-\exp(-\sigma_k^\beta \delta_k), \;\; \sigma_k^\beta = \sigma_k^{o'} + \sigma_k^e, \end{aligned} \tag{7}$$

where $\alpha_k^{o'} = 1-\exp(-\sigma_k^{o'}\delta_k)$ and $\alpha_k^e = 1-\exp(-\sigma_k^e\delta_k)$.

In parallel, the color $\hat{\mathbf{C}}^e(\mathbf{r})$ of the ray for the editable NeRF is calculated as:

$$\hat{\mathbf{C}}^e(\mathbf{r}) = \sum_{k=1}^{K} T_k^\beta (\alpha_k^{o'} \beta_k^c + \alpha_k^e) \mathbf{c}_k^e. \tag{8}$$

The colors $\hat{\mathbf{C}}^o$, $\hat{\mathbf{C}}^e$, and $\hat{\mathbf{C}}^\beta$ are later used to render the three images: original, editable, and blended images.

We also define three types of accumulated opacity for the ray: $\hat{E}_{\text{acc}}^{\text{add}}$, $\hat{E}_{\text{acc}}^{\text{remove}}$, and $\hat{E}_{\text{acc}}^{\text{change}}$. The accumulated opacities are calculated as follows:

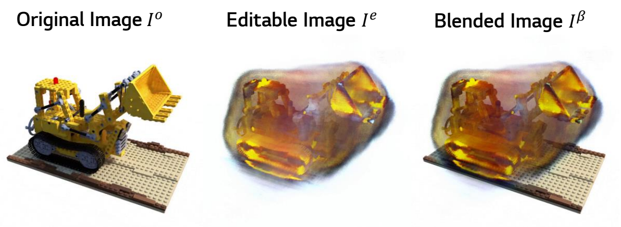

Figure 3. The rendered images of the *'bulldozer'* object edited by the target text *'bulldozer amber'*. Given a sampled camera pose, Blending-NeRF renders three types of images for training.

$$\begin{aligned} \hat{E}_{\text{acc}}^{\text{add}}(\mathbf{r}) &= \sum_{k=1}^{K} T_k^\beta \alpha_k^e \\ \hat{E}_{\text{acc}}^{\text{remove}}(\mathbf{r}) &= \sum_{k=1}^{K} (T_k^{o'} - T_k^o)\alpha_k^o \Big/ \sum_{k=1}^{K} \alpha_k^o \\ \hat{E}_{\text{acc}}^{\text{change}}(\mathbf{r}) &= \sum_{k=1}^{K} T_k^\beta \alpha_k^{o'} \beta_k^c, \end{aligned} \tag{9}$$

where $T_k^{o'} = \prod_{k'=1}^{k-1}(1-\alpha_{k'}^{o'})$. Each accumulated opacity denotes the degree of adding density, removing density, and changing color for the rendered pixel by the blending operations. Specifically, $\hat{E}_{\text{acc}}^{\text{add}}$ represents the amount of density added by the editable NeRF, and $\hat{E}_{\text{acc}}^{\text{remove}}$ represents the amount of density removed from the pre-trained NeRF by the blending ratio β_k^σ. The last opacity $\hat{E}_{\text{acc}}^{\text{change}}$ means the amount of original color $\mathbf{c}_k^o$ changed by the blending operations. Note that the modifications to the object's parts that are occluded in a specific viewpoint are ignored in this operation. These accumulated opacities for the ray are used to limit the region and amount of the object editing, guided by the source text. This method, which plays an important role in localized object editing, is described in Section 4.3.

Volume Rendering There are three types of images that are rendered with Blending-NeRF, namely original, editable, and blended images, during our localized object editing process. To train our model, we first sample a camera pose to generate these images from the sampled viewpoint. For 360° bounded scenes, we set a uniform distribution over the upper hemisphere with bounded radius [9] and sample a camera pose each training iteration.

Given the sampled camera pose $\mathbf{p}$, the rays are also sampled at even intervals to make an image patch of size S covering the entire extent of the image plane (refer to Appendix for details). Then we can obtain $S \times S$ image patches $I^o(\theta, \mathbf{p})$, $I^e(\theta, \phi, \mathbf{p})$, and $I^\beta(\theta, \phi, \mathbf{p})$ for original, editable, and blended images by using Eq. (2), Eq. (8), and Eq. (7). Likewise, for our localized editing, the three types of opacity patches $\hat{E}_{\text{acc}}^{\text{add}}(\theta, \phi, \mathbf{p})$, $\hat{E}_{\text{acc}}^{\text{remove}}(\theta, \phi, \mathbf{p})$, and

$\hat{E}^{\text{change}}_{\text{acc}}(\theta, \phi, \mathbf{p})$ of $S\times S$ size are also obtained using Eq. (9). Once trained, Blending-NeRF can render over an entire image without pixel strides at any camera pose. Examples of rendered images are shown in Figure 3.

4.2. Text-Driven Objective

We leverage the pretrained CLIP model for text-driven object editing on Blending-NeRF. For image patches $I^o(\theta, \mathbf{p})$, $I^e(\theta, \phi, \mathbf{p})$, and $I^\beta(\theta, \phi, \mathbf{p})$ rendered in the previous step, we apply the global and directional CLIP losses of Eq. (4) and Eq. (5): $\mathcal{L}^e_{\text{global}}(I^e(\theta, \phi, \mathbf{p}), T_{\text{target}})$, $\mathcal{L}^\beta_{\text{global}}(I^\beta(\theta, \phi, \mathbf{p}), T_{\text{target}})$ and $\mathcal{L}_{\text{dir}}(I^\beta(\theta, \phi, \mathbf{p}), T_{\text{target}}, I^o(\theta, \mathbf{p}), T_{\text{source}})$. The global CLIP losses $\mathcal{L}^e_{\text{global}}$ and $\mathcal{L}^\beta_{\text{global}}$ make CLIP embeddings of both editable image I^e and blended image I^β close to that of target text T_{target}. The directional CLIP loss ensures that the direction of representation vector from the source image I^o to the blended image I^β is similar to the direction from the source text T_{source} to the target text T_{target}. The total text-driven objective is defined as:

$$\mathcal{L}_{\text{clip}} = \mathcal{L}_{\text{dir}} + \lambda_{\text{global}}\mathcal{L}_{\text{global}}, \tag{10}$$

where $\mathcal{L}_{\text{global}} = \mathcal{L}^\beta_{\text{global}} + \mathcal{L}^e_{\text{global}}$ is the global CLIP loss, and λ_{global} is a hyperparameter for balancing the directional and global CLIP losses.

Image and Text Augmentation Before feeding image patches and text prompts to CLIP encoders, we apply image and text augmentations. Previous work [14] shows that applying 2D image-based augmentations can prevent adversarial generation problems when using CLIP guidance. Similarly, we augment image patches in the order of differential [42] and random perspective augmentations. We use the text templates [2] to augment T_{source} and T_{target}.

4.3. Localized Editing Objective

The text-driven objective can guide Blending-NeRF to edit the original object to match the meaning of a given target text T_{target}. However, with the CLIP losses alone, it is challenging to specify the region and amount of editing. Thus, we employ a text-guided semantic segmentation method and the opacity patches $\hat{E}^{\text{add}}_{\text{acc}}(\theta, \phi, \mathbf{p})$, $\hat{E}^{\text{remove}}_{\text{acc}}(\theta, \phi, \mathbf{p})$, and $\hat{E}^{\text{change}}_{\text{acc}}(\theta, \phi, \mathbf{p})$. Note that precisely targeting the region for localized editing and constraining the degree of object editing is well handled by these three accumulated opacities, which capture the extent of density addition, density removal, and color alteration through our blending operations. In addition, joint optimization of localizing target region and constraining editable amount to maintain the high-fidelity results of the pre-trained NeRF is a vital factor in producing localized editing that is less prone to noise, as demonstrated throughout our experimental results in Section 5.4.

Localizing Target Region We use CLIPSeg [19] to guide the region to be edited only with a user text prompt T_{source}. Specifically, we leverage zero-shot segmentation $h(I, T)$ to produce a probability map of the pixels in an image I associated with the input text T. We first estimate region M, which is more likely to be T_{source} than the text *'photo'* in the source image $I^o(\theta, \mathbf{p})$, as follows:

$$M = \mathbb{1}(h(I^o(\theta, \mathbf{p}), T_{\text{source}}) - h(I^o(\theta, \mathbf{p}), \textit{`photo'})), \tag{11}$$

where function $\mathbb{1}(\cdot)$ pixel-wisely outputs 1 if its input is positive, and 0 otherwise. After applying N_f dilation operations to M, we get the positive target region M_+ which specifies the region of interest to edit. Additionally, we specify the negative target region M_- which designates the region of non-interest by applying N_f dilatation operations to M_+ and element-wise *not* operation. Then the loss to localize the target region is:

$$\begin{aligned}\mathcal{L}_{\text{region}} = \text{MSE}([0]_{S\times S}, M_- \odot E_{\text{sum}}) + \\ \lambda_+ \text{MSE}([1]_{S\times S}, M_+ \odot E_{\text{sum}}),\end{aligned} \tag{12}$$

where $E_{\text{sum}} = \sum_{x=\{\text{add,remove,change}\}} \hat{E}^x_{\text{acc}}(\theta, \phi, \mathbf{p})$ is the pixel-wise sum of the three accumulated opacities, $\odot$ denotes pixel-wise multiplication, and λ_+ is a hyperparameter for balancing the two terms. The first term prevents modification outside the target region, while the second encourages editing within the target region.

Constraining Editable Amount To limit the amount of area being modified, we use an opacity loss similar to the transmittance loss in the previous work [8]. The opacity loss $\mathcal{L}_{\text{opacity}}$ is defined using the opacity patches $\hat{E}^{\text{add}}_{\text{acc}}(\theta, \phi, \mathbf{p})$, $\hat{E}^{\text{remove}}_{\text{acc}}(\theta, \phi, \mathbf{p})$, and $\hat{E}^{\text{change}}_{\text{acc}}(\theta, \phi, \mathbf{p})$ as follows:

$$\mathcal{L}_{\text{opacity}} = \sum_x \max(\tau^x, \text{mean}(\hat{E}^x_{\text{acc}}(\theta, \phi, \mathbf{p}))), \tag{13}$$

where $x = \{\text{add, remove, change}\}$ and $\{\tau^x\}$ are the thresholds to limit the amount of addition and removal of density, and change of color. These thresholds are annealed for stable learning.

We also apply the regularization loss $\mathcal{L}_{\text{reg}}$ to the opacity patch $\hat{E}^{\text{add}}_{\text{acc}}(\theta, \phi, \mathbf{p})$ to avoid adding ambiguous densities:

$$\mathcal{L}_{\text{reg}} = -\text{mean}(F(\hat{E}^{\text{add}}_{\text{acc}}(\theta, \phi, \mathbf{p}))), \tag{14}$$

where $F(z) = z \log_2 z + (1-z)\log_2(1-z)$ is the binary entropy function. Note that we use stop gradients to ensure that localized editing losses do not indiscriminately affect training. In particular, the losses for opacity patch $\hat{E}^{\text{add}}_{\text{acc}}(\theta, \phi, \mathbf{p})$ to add density are only concerned with the backpropagation by editable density σ^e. Likewise, the losses by opacity patches $\hat{E}^{\text{remove}}_{\text{acc}}(\theta, \phi, \mathbf{p})$ and $\hat{E}^{\text{change}}_{\text{acc}}(\theta, \phi, \mathbf{p})$ propagate only to β^σ and β^c, respectively.

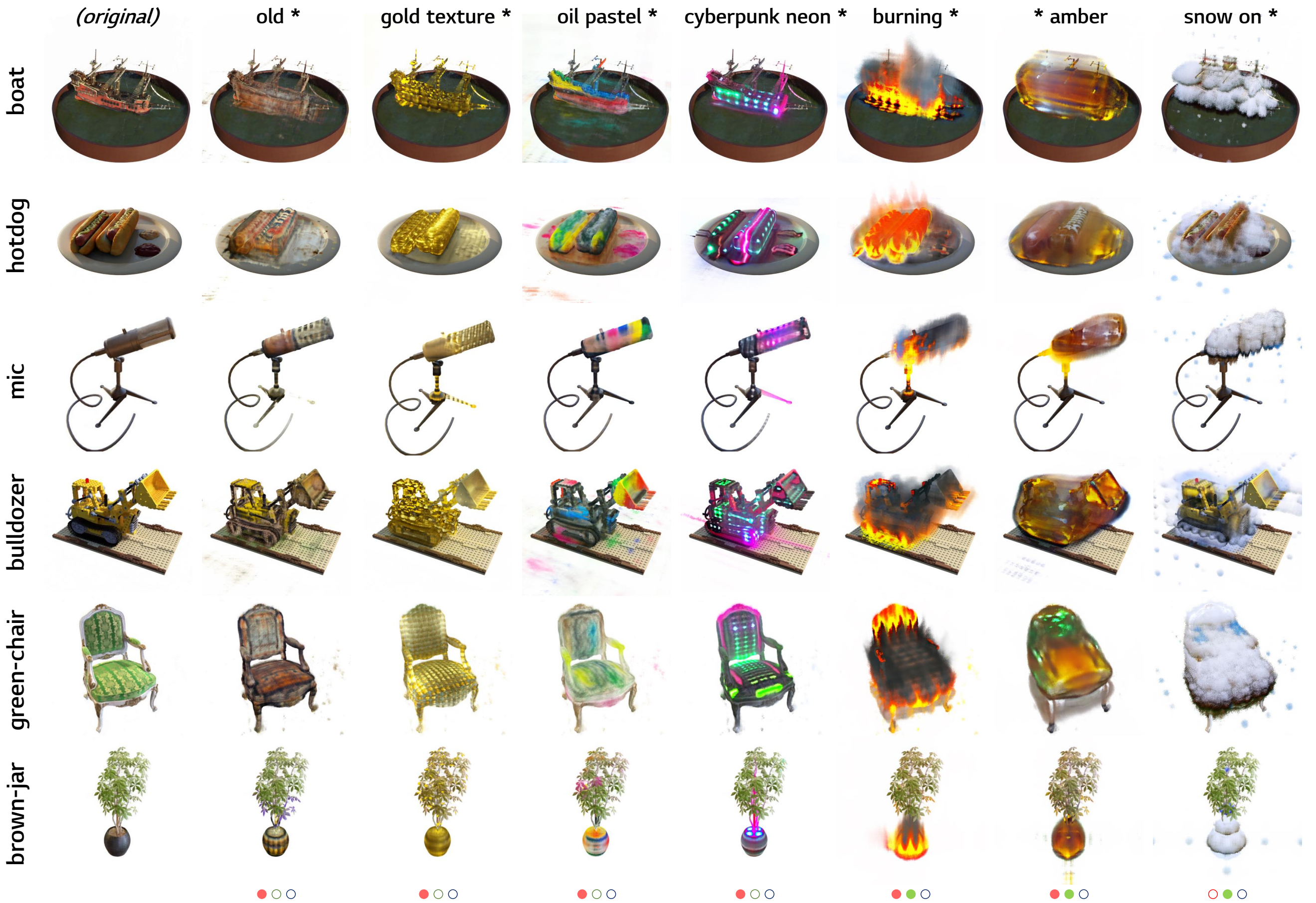

Figure 4. Examples of editing in which the common target text templates are applied to various 3D objects. The images in the first column are the original objects. The object names are listed on the left side of the figure, and the target text templates are listed on the top of the figure. For example, the second image in the first row is an edited result with *'old boat'*, which combines *'boat'* and *'old *'*. We denote the ● as changing colors, ● as adding densities, and ● as removing densities of the object. If two or more dots exist, the editing is performed with the corresponding case together. This notation is common to all figures.

● : changing colors ● : adding densities ● : removing densities.

Finally, our total objective $\mathcal{L}_{\text{total}}$ for text-driven localized object editing is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{clip}} + \lambda_1 \mathcal{L}_{\text{region}} + \lambda_2 \mathcal{L}_{\text{opacity}} + \lambda_3 \mathcal{L}_{\text{reg}}, \tag{15}$$

where λ_1, λ_2, and λ_3 are hyperparameters to balance losses.

5. Experiments and Results

5.1. Implementation Details

The pretrained NeRF consists of an 8-layer MLP of 256 hidden units with ReLU activations as in the architecture of the originally proposed NeRF [22]. For the editable NeRF, we partially modified the original NeRF using residual blocks (see Appendix for details). We followed the same procedures in the hierarchical volume sampling of NeRF as well, but we did the importance sampling based on $T_k^\beta \alpha_k^\beta$ instead of $T_k^o \alpha_k^o$. The patch size for all images and accumulated opacities is $S = 72$. We used Adam Optimizer and the learning rate is linearly decayed from 5×10^{-4} to 10^{-4} for the first 1k iteration steps and stays at 10^{-4} for the remaining steps. We included the regularization loss component only after the first 1k iteration steps, once the density of the newly added object has reached a certain level of form. Regarding the hyperparameter values, we employed the following: $\lambda_{\text{global}} = 0.5$, $\lambda_1 = 1$, $\lambda_2 = 2$, and $\lambda_3 = 0.2$. We evaluated our method using a variety of target texts and six 3D objects (ship, hotdog, mic, lego, chair, and ficus) from the Realistic Synthetic 360° dataset [22].

5.2. Localized Editing

To investigate the performance of our method for localized object editing, we performed a variety of experiments such as addition or removal of densities, and color changes to the original objects. Figure 4 shows the edited results obtained by applying the same target text templates to all source objects. Our method clearly achieves a detailed edit-

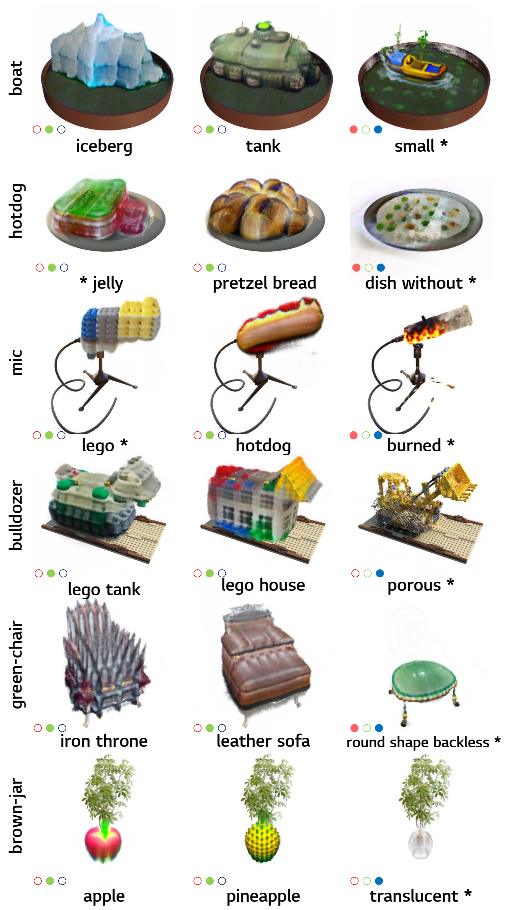

Figure 5. Examples of editing in which the target-specific text descriptions are applied to 3D objects. The texts on the left of the images refer to the editing target on the original objects.

ing of the target object while preserving the source structure. For example, when given a source text *'boat'* and a target text *'gold texture boat'*, a well-stylized boat with a gold texture was rendered while preserving the background. The localized editing also worked well when simultaneously adding densities to an object and changing its colors. In the examples of *'burning bulldozer'* or *'snow on mic'*, the appropriate ambient effect appeared naturally along with editing the target object.

We extended our experiments to a more diverse set of target texts, as shown in Figure 5. In particular, we performed object editing tasks to remove densities. For instance, given a source text *'green-chair'* and a target text *'round shape backless green-chair'*, the back and armrests of the chair were removed to achieve the editing goal.

5.3. Comparison with Baselines

Baselines To demonstrate the effectiveness of our approach, we compared it against three different variants of CLIP-NeRF. Wang *et al.* [35] present a single NeRF-based editing method per scene (let's call it CLIP-NeRF-*c*). We compared our method against CLIP-NeRF-*c*, which only fine-tunes its color-related layers, using officially released code. We also evaluated our method against CLIP-NeRF-*f*, which fine-tunes all layers instead of just the color-related ones. Additionally, we compared our method against another variant, CLIP-NeRF-*D*, which uses distilled feature fields [13] as a localization module using official code that fine-tunes all layers.

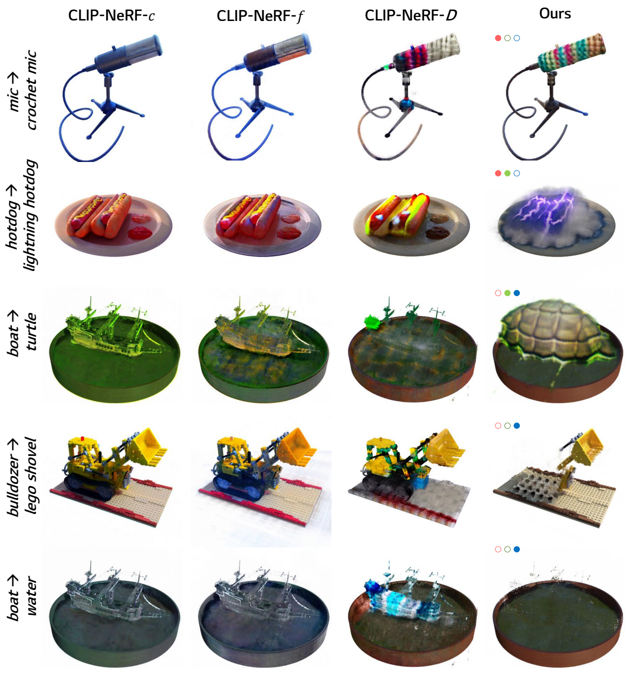

Figure 6. Comparison with baselines. Our method demonstrates superior ability, particularly in editing density, as evidenced by the third row where we added densities, as well as the fourth and fifth rows where we removed densities. In contrast, the competing methods fail to achieve these tasks.

	CLIP-NeRF-*c*	CLIP-NeRF-*f*	CLIP-NeRF-*D*	Ours
$D_{L_1} \downarrow$	**.029**	.041	.047	.051
$S_{\text{CLIP}} \uparrow$	.065	.081	.084	**.128**
$\text{MP}_{\text{CLIP}} \uparrow$	.063	.077	.080	**.121**

Table 1. Quantitative comparison with baseline models. We measured the preservation of the original appearance (D_{L_1}) and the alignment with the target text (S_{CLIP}). We also measured the manipulative precision (MP_{CLIP}) to consider them both.

Evaluation Metric We evaluated the quality of text-driven object editing using the manipulative precision (MP) metric [15]. The MP metric takes into account two aspects: the preservation of the original appearance, which is measured as the L1 normalized pixel distance (D_{L_1}) between the original and edited image, and the alignment with the target text, which is measured by the CLIP score (S_{CLIP})

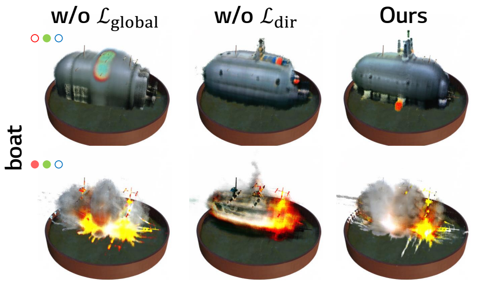
w/o $\mathcal{L}_{\text{global}}$
w/o $\mathcal{L}_{\text{dir}}$
Ours
boat

Figure 7. Ablation study on the text-driven objectives. The *'boat'* object is edited with *'submarine'* (upper row) and *'exploding boat'* (bottom row) as the target texts.

	w/o $\mathcal{L}_{\text{region}}$	w/o $\mathcal{L}_{\text{opacity}}$	w/o $\mathcal{L}_{\text{reg}}$	w/o $\mathcal{L}_{\text{global}}$	w/o $\mathcal{L}_{\text{dir}}$	Ours
D_{L_1} ↓	**.049**	.085	.053	.053	.049	.051
S_{CLIP} ↑	.121	.126	.125	.122	.108	**.128**
MP_{CLIP} ↑	.115	.115	.119	.115	.103	**.121**

Table 2. Quantitative comparison with ablation models. We ablate the proposed localized editing (w/o $\mathcal{L}_{\text{region}}$, w/o $\mathcal{L}_{\text{opacity}}$, and w/o $\mathcal{L}_{\text{reg}}$) and text-driven objectives (w/o $\mathcal{L}_{\text{global}}$ and w/o $\mathcal{L}_{\text{dir}}$).

between the edited image and target text. The CLIP score is obtained by averaging CLIP similarity and Directional CLIP similarity [12]. For a fair comparison, we used CLIP ViT-L/14 to calculate the CLIP score instead of CLIP ViT-B/32 used to train Blending-NeRF and the baselines. Finally, the CLIP based MP metric is defined as $\text{MP}_{\text{CLIP}} = (1 - D_{L_1}) \times S_{\text{CLIP}}$. We calculated all these metrics using 60 different scenes for each model.

Comparisons We qualitatively and quantitatively compared the performance of Blending-NeRF with three variants of CLIP-NeRF. Our method outperformed all baselines, as shown in Figure 6. The CLIP-NeRF variants were able to perform the color change task (*mic* → *crochet mic*), but struggled with the density change tasks. Although the localizing module helped CLIP-NeRF-*D* to edit the target region well, it still had difficulty in editing the densities of the target object. As indicated by Wang *et al.* [35], this demonstrates that relying on the simple fine-tuning of a single NeRF to generate new densities in the low initial density area or to alter existing densities through a CLIP-driven objective is not sufficient for achieving complete localized object editing of shapes. Instead, we found in our experiments that using our novel dual NeRF architecture to blend volumetric information from two independent NeRFs, namely, pretrained NeRF capturing the original 3D model and editable NeRF capturing object editing information, and specifying the positive and negative regions to help the blended editing focus on the target regions results in more natural localized object editing. These qualitative results in Figure 6 are consistent with the superior quantitative performance of our model, measured using MP metric, as shown in Table 1.

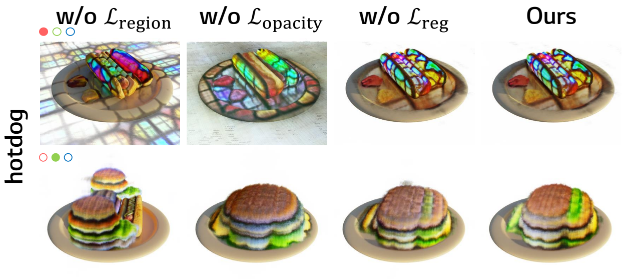
w/o $\mathcal{L}_{\text{region}}$
w/o $\mathcal{L}_{\text{opacity}}$
w/o $\mathcal{L}_{\text{reg}}$
Ours
hotdog

Figure 8. Ablation study on the localized editing objectives. The *'hotdog'* object is edited with *'stained glass hotdog'* (upper row) and *'hamburger'* (bottom row) as the target texts.

5.4. Ablation Study

To validate the effect of our text-driven and localized editing losses, we compared the performance qualitatively as well as quantitatively when each loss term was removed from the total objective. We first performed an ablation study on text-driven losses, as shown in Figure 7. In the *'submarine'* case, when global CLIP loss was not used (w/o $\mathcal{L}_{\text{global}}$), the result was blurry with degraded quality. Similarly, in the *'exploding boat'* case, when only global CLIP loss was used (w/o $\mathcal{L}_{\text{dir}}$), the explosion effect was not well expressed, resulting in a poor editing performance. These results are also consistent with the poor CLIP scores and MP metrics, as shown in Table 2. That is, using both global and directional losses enhances the overall editing quality.

We further analyzed the effect of localized editing losses on localizing the target region. As shown in Figure 8, when each localized editing loss was excluded (*i.e.*, w/o $\mathcal{L}_{\text{region}}$ or w/o $\mathcal{L}_{\text{opacity}}$), the editing regions were not well targeted or adequately constrained overall. As shown in Table 2, these results are also consistent with the low MP metric (w/o $\mathcal{L}_{\text{region}}$) and the poor preservation score (w/o $\mathcal{L}_{\text{opacity}}$). Additionally, for the w/o $\mathcal{L}_{\text{reg}}$ in the *'hamburger'* case, the edited object has ambiguous boundaries. In contrast, our method edits objects with clear boundaries and less noise. This result implies that the regularization loss guides Blending-NeRF to add density distinctly, improving the MP metric as shown in Table 2. That is, our method can locally and naturally edit the target object in the original scene with only minor modifications to the regions of non-interest.

5.5. Extendability of Blending-NeRF

We investigated the extendability of the proposed method using Instant-NGP [23] which utilizes hash grid encoding to represent a 3D scene with low computational cost. The localized editing results on real scenes [22] in Figure 9 demonstrate that our method can be integrated with other 3D scene representation methods such as Instant-NGP. In this experiment, Blending-NeRF was able to inherit the advantages of Instant-NGP over the originally proposed

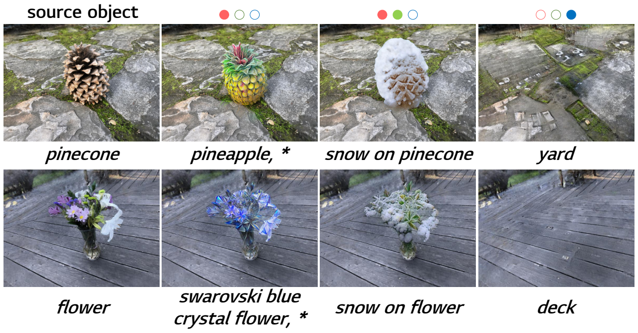

Figure 9. Examples of localized editing on the *pinecone* and *vasedeck* scenes. Blending-NeRF with Instant-NGP was used for these results. (* : *trending on artstation*)

NeRF [22] on memory efficiency and training time. For more results and implementation details, refer to Section E of Appendix.

5.6. Editing Operations

Our approach explicitly distinguishes editing operations, such as adding and removing density, and changing color. The editing results can vary even for the same scene and text, depending on the manually specified editing operations. By choosing different combinations of the editing operations, users have the ability to achieve desired editing, as shown in Figure 10.

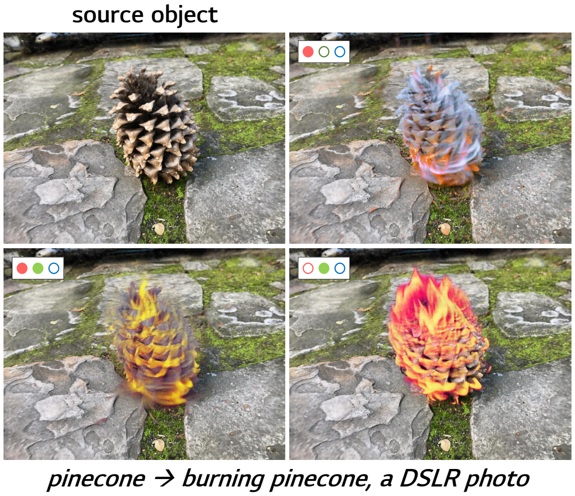

Figure 10. Experiments on using different editing operations. The source object (*'pinecone'*) is edited into each object using the same target text (*'burning pinecone, a DSLR photo'*) with different combinations of editing operations. Note that Blending-NeRF with Instant-NGP was used for these results.

5.7. Limitations

Our work has limitations in that the overall performance can be affected by the two off-the-shelf models, CLIPSeg and CLIP. For instance, if the segmentation of the target area by CLIPSeg is not appropriate, unedited parts may remain. This performance degradation can be mitigated by using advanced segmentation models or a potential solution described in Appendix (*i.e.*, user-provided mask).

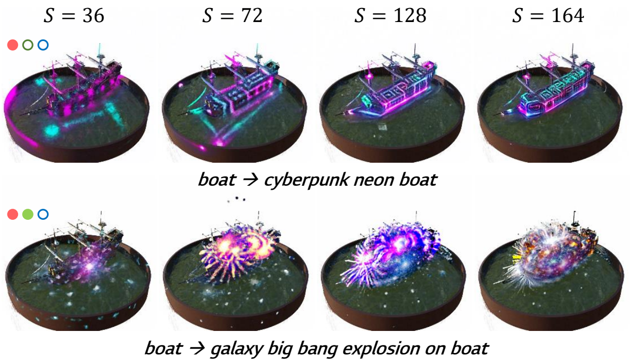
$S = 36$
$S = 72$
$S = 128$
$S = 164$
●○○
boat → cyberpunk neon boat
●●○
boat → galaxy big bang explosion on boat

Figure 11. Experiments on object editing with various patch sizes. The numbers at the top denote each patch size used. The source object (*'boat'*) is edited into each object using two target texts: *'cyberpunk neon boat'* (top row) and *'galaxy big bang explosion on boat'* (bottom row), respectively. Note that Blending-NeRF with Instant-NGP was used for these results.

Additionally, we found that the limited patch size input to CLIP's image encoder can make edited results blurry. The input size of the CLIP encoder is 224, but we used a patch size of 72 due to our computational resources when we used the originally proposed NeRF [22] as a backbone. However, this issue can be alleviated by using a memory-efficient backbone (*i.e.*, Instant-NGP). As shown in Figure 11, where we applied the proposed method to Instant-NGP as described in Section 5.5, the blurry results were improved as the patch sizes increased. Considering the trade-off between the quality improvement and the increase in computational time, we set the patch size as 128 for our experiments using Instant-NGP.

6. Conclusion

For text-driven localized 3D object editing, we propose Blending-NeRF, which consists of pretrained NeRF and editable NeRF. The target region for editing is specified by the source text and the original object in the pretrained NeRF. Blending-NeRF renders blended images of two NeRFs suitable for the target text by freezing the pretrained NeRF and training the editable NeRF to locally edit the original object while maintaining the overall appearance. Especially, we define three types of editing operations (*i.e.*, adding or removing density, changing color) and use them to perform various 3D object editing. Empirical results show that our approach is superior to text-driven localized object editing. We firmly believe that the proposed method and localized object editing tasks hold practical value in neural rendering.

References

[1] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022. 3

[2] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. *arXiv preprint arXiv:2204.02491*, 2022. 3, 5

[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. 2

[4] Yongwei Chen, Rui Chen, Jiabao Lei, Yabin Zhang, and Kui Jia. Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. *arXiv preprint arXiv:2210.11277*, 2022. 1, 2

[5] Jian Ding, Nan Xue, Gui-Song Xia, and Dengxin Dai. Decoupling zero-shot semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11583–11592, 2022. 3

[6] Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022. 3

[7] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 2

[8] 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, 2, 5

[9] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021. 4

[10] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021. 3

[11] Nasir Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. *ACM Transactions on Graphics (TOG), Proc. SIGGRAPH Asia*, 2022. 2

[12] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2426–2435, 2022. 3, 8

[13] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *arXiv preprint arXiv:2205.15585*, 2022. 7

[14] Han-Hung Lee and Angel X Chang. Understanding pure clip guidance for voxel grid nerf models. *arXiv preprint arXiv:2209.15172*, 2022. 1, 5

[15] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. Manigan: Text-guided image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7880–7889, 2020. 7

[16] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *arXiv preprint arXiv:2201.03546*, 2022. 3

[17] Yangguang Li, Feng Liang, Lichen Zhao, Yufeng Cui, Wanli Ouyang, Jing Shao, Fengwei Yu, and Junjie Yan. Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm. *arXiv preprint arXiv:2110.05208*, 2021. 3

[18] 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. 1, 2

[19] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7086–7096, June 2022. 2, 3, 5

[20] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 4

[21] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022. 1, 2

[22] 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, 6, 8, 9

[23] 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, 8

[24] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *ACM Trans. Graph.*, 37(6), Nov. 2018. 2

[25] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094, 2021. 3

[26] 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

[27] Thomas Porter and Tom Duff. Compositing digital images. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, page 253–259, New York, NY, USA, 1984. Association for Computing Machinery. 3

[28] 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. 2, 3

[29] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 2

[30] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 2

[31] Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang, Chin-Yi Cheng, Marco Fumero, and Kamal Rahimi Malekshan. Clip-forge: Towards zero-shot text-to-shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18603–18613, 2022. 2

[32] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019. 2

[33] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. 2

[34] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 2

[35] 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. 1, 2, 3, 7, 8

[36] Zhaoqing Wang, Yu Lu, Qiang Li, Xunqiang Tao, Yandong Guo, Mingming Gong, and Tongliang Liu. Cris: Clip-driven referring image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11686–11695, 2022. 3

[37] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. Dˆ 2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *Advances in Neural Information Processing Systems*, 35:32653–32666, 2022. 2

[38] Mengde Xu, Zheng Zhang, Fangyun Wei, Yutong Lin, Yue Cao, Han Hu, and Xiang Bai. A simple baseline for zero-shot semantic segmentation with pre-trained vision-language model. *arXiv preprint arXiv:2112.14757*, 2021. 3

[39] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13144–13152, 2021. 2

[40] 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. 1, 2

[41] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18123–18133, 2022. 2, 3

[42] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 5