DreamEditor: Text-Driven 3D Scene Editing with Neural Fields

Jingyu Zhuang*
zhuangjy6@mail2.sysu.edu.cn
Sun Yat-sen University
China

Chen Wang*
cw.chenwang@outlook.com
University of Pennsylvania
USA

Liang Lin†
linliang@ieee.org
Sun Yat-sen University
China

Lingjie Liu†
lingjie6@seas.upenn.edu
University of Pennsylvania
USA

Guanbin Li†
liguanbin@mail.sysu.edu.cn
Sun Yat-sen University
China

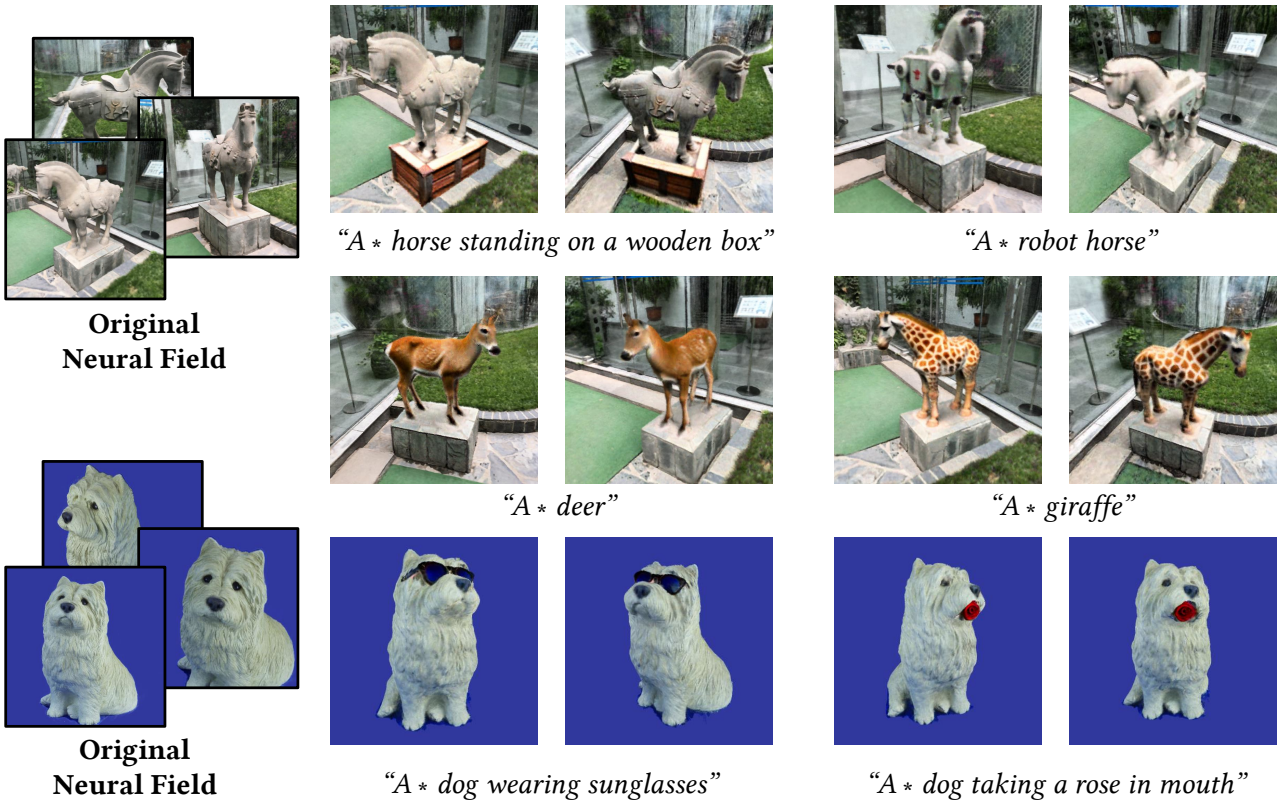

Figure 1: Our approach DreamEditor allows users to edit 3D scenes with text prompts. DreamEditor achieves precise and high-quality editing that maintains irrelevant regions unchanged.

ABSTRACT

Neural fields have achieved impressive advancements in view synthesis and scene reconstruction. However, editing these neural fields remains challenging due to the implicit encoding of geometry and texture information. In this paper, we propose DreamEditor, a novel framework that enables users to perform controlled editing of neural fields using text prompts. By representing scenes as mesh-based neural fields, DreamEditor allows localized editing within specific regions. DreamEditor utilizes the text encoder of a pretrained text-to-Image diffusion model to automatically identify the regions to be edited based on the semantics of the text prompts. Subsequently, DreamEditor optimizes the editing region and aligns

*Both authors contributed equally to this research.
†Corresponding authors. Welcome to *Code* and *Project page*

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0315-7/23/12…$15.00
https://doi.org/10.1145/3610548.3618190

its geometry and texture with the text prompts through score distillation sampling [Poole et al. 2022]. Extensive experiments have demonstrated that DreamEditor can accurately edit neural fields of real-world scenes according to the given text prompts while ensuring consistency in irrelevant areas. DreamEditor generates highly realistic textures and geometry, significantly surpassing previous works in both quantitative and qualitative evaluations.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Neural networks**.

ACM Reference Format:
Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. 2023. DreamEditor: Text-Driven 3D Scene Editing with Neural Fields. In *SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers '23), December 12–15, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3610548.3618190

1 INTRODUCTION

Neural radiance fields [Mildenhall et al. 2021], NeuS [Wang et al. 2021] and subsequent research [Liu et al. 2020; Müller et al. 2022; Wang et al. 2022c] (collectively referred to as *neural fields*) have made significant progress in scene reconstruction and novel view synthesis. By capturing multi-view images of a 3D scene and using off-the-shelf structure-from-motion models to estimate camera poses, one can train neural networks to learn neural fields that implicitly represent the geometry and texture of the scene. Compared to the traditional pipeline involving tedious 3D matching and complex postprocessing steps, neural fields offer a more efficient and accessible approach for reconstructing real-world objects and scenes into Computer Graphics assets for general users.

However, editing neural fields is not a straightforward task since the shape and texture information is implicitly encoded within high-dimensional neural network features. Conventional 3D modeling techniques are ineffective for manual sculpting and re-texturing since explicit geometry is not available. Previous research has explored techniques for neural fields editing, such as moving objects in a scene [Chen et al. 2021], modifying textures [Xiang et al. 2021], and altering object shape [Yang et al. 2022]. However, these editing procedures still require extensive user input. While recent work has enabled NeRF editing with text prompts [Haque et al. 2023], it struggles to achieve precise and high-quality editing due to a restricted diversity of instructions. Consequently, further research is needed to develop *easy-to-use* and *accurate* 3D editing methods, enabling improved re-creation of existing 3D assets.

In this paper, we present DreamEditor, a framework that allows users to intuitively and conveniently modify neural fields using text prompts. As illustrated in Fig. 1, for a given scene represented by a neural field, *e.g.,* a dog or a complex outdoor environment, text descriptions can be used to achieve various object-centric editing, including re-texturing, object replacement, and object insertion, while simultaneously preserving irrelevant regions to the text prompts. This is made possible through two key designs in our method: (1) a mesh-based neural field representation, and (2) a stepwise framework that leverages pretrained diffusion models for 3D editing. Compared to an implicit representation, an explicit mesh-based neural field enables the efficient conversion of 2D editing masks into 3D editing regions through back projection, facilitating precise local editing by only modifying the masked regions. Additionally, the mesh representation disentangles geometry and texture, preventing unnecessary geometry deformation when only appearance changes are expected. Leveraging the advantages of the mesh representation, we propose a stepwise *finetune-localization-optimization* framework that efficiently and accurately edits 3D scenes using simple text prompts, achieved by score distillation sampling within the masked region.

We extensively evaluate DreamEditor on various synthetic and real-world scenes, including animals, human faces and outdoor scenes. Unlike methods that operate on the entire image, our editing approach enables precise local deformations while naturally preserving irrelevant areas. For example, in Fig. 1, only the dog's mouth is modified when holding a rose in its mouth. Furthermore, as the edit can be accomplished with a simple text prompt, the procedure is user-friendly and significantly simplifies the editing of neural fields, showing its great potential for practical applications. Both qualitative and quantitative comparisons demonstrate the superiority of DreamEditor over previous methods in terms of editing precision, visual fidelity and user satisfaction.

The contributions of this paper can be summarized as follows: (1) We introduce a novel framework for text-guided 3D scene editing, which achieves highly realistic editing results for a wide range of real-world scenes; (2) We propose to use a mesh-based neural field to enable local modification of the scene and decouple texture and geometric features for flexible editing; (3) We devise a stepwise editing framework that first identifies the specific regions requiring editing according to text prompts and then performs modifications exclusively within the selected regions. This systematic procedure ensures precise 3D editing while minimizing unnecessary modifications in irrelevant regions.

2 RELATED WORKS

2.1 Text-guided image generation and editing

The denoising diffusion probabilistic model [Ho et al. 2020; Song et al. 2020] has drawn great attention for its ability to generate high-quality images. Later, diffusion models [Ramesh et al. 2022; Rombach et al. 2022; Saharia et al. 2022] trained on large-scale image-text paired datasets demonstrated astonishing performance in understanding complex semantics from text prompts (including nouns, adjectives, verbs, etc.) and generating corresponding high-quality images. Due to the rich semantics and high controllability of pretrained text-to-image diffusion models, a series of studies [Avrahami et al. 2022; Couairon et al. 2022; Hertz et al. 2022; Kawar et al. 2022] have employed them to text-guided image editing. Most related to our work is subject-driven generation with text-to-image diffusion models [Gal et al. 2022a; Ruiz et al. 2022], which enables users to personalize their image generation for specific subjects and concepts given. DreamBooth [Ruiz et al. 2022] expands the language-vision dictionary using rare tokens and finetunes the model with a preservation loss for regularization. Similarly, Textual Inversion [Gal et al. 2022a] optimizes a new "word" in the embedding space of the pre-trained diffusion model to represent the input objects. These works generate images with novel concepts, but it is non-trivial to extend these 2D methods to 3D.

2.2 Text-to-3D generation

With the development of text-to-image generation models, there has been a growing interest in text-to-3D generation. Some works use the CLIP model to optimize mesh [Chen et al. 2022; Michel et al. 2022; Mohammad Khalid et al. 2022] or neural fields [Jain et al. 2022]. The seminar work DreamFusion [Poole et al. 2022] first proposes score distillation sampling (SDS) loss to distill the knowledge in pretrained 2D Text-to-Image diffusion models for text-to-3D generation. A series of works [Chen et al. 2023; Lin et al. 2022; Metzer et al. 2022; Raj et al. 2023] based on SDS loss, further improve the generation results by introducing geometry prior or changing 3D representation. Score Jacobian Chaining [Wang et al. 2022b] arrives at a similar training objective from the perspective of approximating 3D score with the 2D score. However, all these methods lack the ability to edit existing 3D scenes. One of the main reasons is the difficulty in fully aligning an existing 3D scene with text, resulting in these methods tending to generate a new scene and breaking the consistency before and after editing. To overcome this limitation, we propose a novel text-guided 3D editing framework that can edit an existing 3D scene based on text prompts.

2.3 Neural Field Editing

Editing neural fields is inherently difficult due to its entangled shape and appearance. EditNeRF [Liu et al. 2021] is the first work to support editing the shape and color of neural fields conditioned on latent codes. Some works [Bao et al. 2023; Gao et al. 2023; Wang et al. 2022a, 2023] further leverage a CLIP model to allow editing with text prompts or reference images. Another line of work uses pre-defined template models or skeletons to support re-posing or re-rendering [Noguchi et al. 2021; Peng et al. 2021], but is constrained in a specific category. 3D editing can also be achieved by combining 2D image manipulation such as inpainting with neural fields training [Kobayashi et al. 2022; Liu et al. 2022]. Geometry-based methods [Li et al. 2022; Xu and Harada 2022; Yang et al. 2022; Yuan et al. 2022] export neural fields to mesh and synchronize the deformation of the mesh back to implicit fields. TEXTure [Richardson et al. 2023] uses a text prompt to generate the textures of the mesh using an iterative diffusion-based process.

The most similar work to ours is Instruct-NeRF2NeRF [Haque et al. 2023] and Vox-E [Sella et al. 2023], which edit a neural field freely text prompts. Instruct-NeRF2NeRF uses image-based diffusion model [Brooks et al. 2022] to edit the input image with instructions for optimizing the neural field. Nonetheless, since it manipulates the entire image, usually undesired regions will also be changed. Vox-E adopts SDS loss and performs local editing in 3D space by 2D cross-attention maps. However, due to the constraints inherent of Vox-E's volumetric representation, the editing quality of real scenes remains suboptimal.

3 BACKGROUND

Optimizing Neural Fields with SDS Loss. DreamFusion [Poole et al. 2022] proposed the score distillation sampling (SDS) loss to distill the priors Text-to-Image (T2I) diffusion models for 3D generation. It first adds random Gaussian noise at level t to a random rendered view $\hat{I}$ to get $\hat{I}_t$. The pretrained diffusion model ϕ is used to predict the added noise given $\hat{I}_t$ and the input text condition y. The SDS loss is calculated as the per-pixel gradient as follows:

$$\nabla_\theta \mathcal{L}_{SDS}(\phi, \hat{I} = g(\theta)) = \mathbb{E}_{\epsilon,t}\left[w(t)(\epsilon_\phi(\hat{I}_t; y, t) - \epsilon)\frac{\partial \hat{I}}{\partial \theta}\right], \quad (1)$$

where $w(t)$ is a weighting function that depends on noise level t, θ is the parameters of neural field and g is the rendering process. During training, the diffusion model is frozen and gradients are back-propagated to θ, enforcing the neural field's renderings to resemble the images generated by the diffusion model with the text condition y.

DreamBooth. DreamBooth [Ruiz et al. 2022] is a subject-driven image generation method based on T2I models. Given a few images of the same subject, DreamBooth embeds the subject into a T2I diffusion model by binding it to a unique identifier (denoted as $*$). It uses an L2 reconstruction loss to fine-tune the diffusion model on the input images and a class prior-preserving loss to prevent overfitting. The details of its training can be found in Ruiz et al [2022]. In this paper, we also adopt DreamBooth to fine-tune the T2I diffusion models for expressing a specific scene.

4 METHOD

4.1 Overview

The inputs of our method are a set of posed images of a 3D scene to be edited and a text prompt for editing. Our goal is to change the shape and appearance of the object of interest in the original 3D scene according to the text prompt. Fig. 3 gives an example of turning a horse sculpture into a real giraffe. This task requires keeping the 3D contents irrelevant to the text prompt unchanged before and after editing.

The framework of DreamEditor is shown in Fig. 3, which consists of three stages. We first transform the original neural radiance field into a mesh-based neural field (Section 4.2), which enables us to achieve spatially-selective editing. In Section 4.3, we customize the T2I model to the input scene and use the cross-attention maps of it to locate the editing area in the 3D space according to the keywords in the text prompts. Finally, we edit the target object in the neural field under the control of text prompts through the T2I diffusion model (Section 4.4).

4.2 Distilling Neural Fields

Inspired by [Yang et al. 2022], we first learn a neural radiance field from input images and decompose it into many local implicit fields organized in an explicit mesh, where the mesh is extracted from the neural radiance field using marching cubes [Lorensen and Cline 1987]. Representing a scene as a mesh-based neural field introduces two benefits. First, a mesh-based neural field enables precise editing of specific regions in the scene. The regions, such as background and irrelevant objects, can remain unchanged during editing by fixing the specific implicit fields. Second, the extracted mesh can explicitly represent the surface and outline of the objects in the scene. Compared with other explicit representations such as voxels [Liu et al. 2020] and point clouds [Ost et al. 2022], it is more convenient to determine the range of editing area with mesh. Combining the attention scheme of the diffusion model, we further propose a method to automatically determine the editing area,

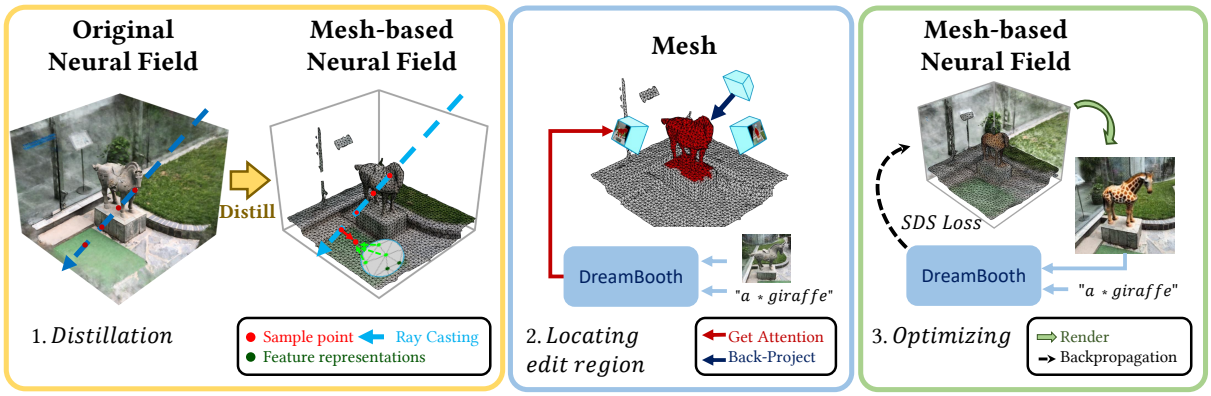

Figure 2: The overview of our method. Our method edits a 3D scene by optimizing an existing neural field to conform with a target text prompt. The editing process involves three steps: (1) The original neural field is distilled into a mesh-based one. (2) Based on the text prompts, our method automatically identifies the editing region of the mesh-based neural field. (3) Our method utilizes the SDS loss to optimize the color feature f_c, geometry feature f_g, and vertex positions v of the editing region, thereby altering the texture and geometry of the respective region. Best viewed in color.

which can accurately locate the editing area in the mesh according to the input text.

Specifically, after the neural radiance field is obtained, we adopt a teacher-student based training framework to perform distillation, where the neural radiance field is taken as the teacher model to guide the student model, i.e., the mesh-based neural field. We define the mesh-based neural field by assigning each mesh vertex $\mathbf{v}$ a color feature f_c and a geometry feature f_g to represent the local shape and texture information near $\mathbf{v}$, respectively. During the volume rendering process, for a sampled point x, we first obtain the aggregated features $\tilde{f}_c$ and $\tilde{f}_g$ by interpolating the features of the top K nearest vertices of x weighted by the inverse distance $(\mathbf{v}_k - x)$ [Qi et al. 2017]:

$$\tilde{f}_t(x) = \frac{\sum_{k=1}^{K} w_k f_{t,k}}{\sum_{k=1}^{K} w_k}, w_k = \frac{1}{||\mathbf{v}_k - x||}, t \in \{g, c\} \tag{2}$$

Then, $\tilde{f}_g$ and $\tilde{f}_c$ are decoded to the s-density s and color c of x:

$$s = D_G(\tilde{f}_g, \tilde{h}), \quad c = D_C(\tilde{f}_c, \tilde{h}, \mathbf{d}, \nabla_x s) \tag{3}$$

where D_G and D_C are the geometry decoder and color decoder respectively, $\tilde{h}$ is the interpolated signed distance of x to $\mathbf{v}_k$, $\mathbf{d}$ is the ray direction and $\nabla_x s$ is the gradient of s-density s at point x. The framework of the network is shown in Fig. 9.

During the distillation process, we randomly sample rays r in the scene and use the output of the teacher model given r as the ground truth, including the rendered pixel color $\hat{C}(r)$, s-density $\hat{s}_i$ and point color $\hat{c}_i$ of each sampling point x on this ray. The distillation loss is computed as:

$$\mathcal{L}_{dis} = \sum_{r\in R}\sum_{i\in N}(\|\hat{s}_i - s_i\| + \|\hat{c}_i - c_i\|) + \sum_{r\in R}\left\|\hat{C}(r) - C(r)\right\|_2^2, \tag{4}$$

where the volume rendering formulation of teacher and student models (*i.e.,* $\hat{C}$ and C) is the same as NeuS [Wang et al. 2021]. Besides, we add Eikonal loss [Gropp et al. 2020] on the sampled points to regularize the norm of the spatial gradients with weight $\lambda_{reg} = 0.01$

$$\mathcal{L}_{reg} = \sum_{r\in R}\sum_{i\in N}\left\|\|\nabla_{x_i} s_i\| - 1\right\|_2^2. \tag{5}$$

In our framework, all camera pose sampling is based on the spherical coordinate system. We transform the target object to the origin and make the y-axis point upwards. We confine the scope of sampled views by setting the range of the elevation and azimuth angles in the following locating and optimizing step, thereby improving editing efficiency.

4.3 Locating Editing Regions

As illustrated in the middle part of Fig 2, given text prompts, DreamEditor first determines the target editing area in a rendered view. As a preparation step, we first fine-tune the Stable Diffusion model with DreamBooth with the sampled views, which adapts the model's knowledge to the specific scene. Then, we utilize the fine-tuned diffusion model to obtain a 2D mask for each rendered view. Finally, we obtain the 3D editing region by back-projecting the masked target region from different views onto the mesh.

The locating is motivated by the fact that cross-attention layers in T2I diffusion models control the relationship between the layout of the generated images and each word [Hertz et al. 2022]: $M = \text{Softmax}(QK^T/\sqrt{q})$, where Q is the query features projected from the spatial features of the noisy image with dimension q, K is the key matrix projected from the textual embedding, M is the attention map that defines the weight of a token for each pixel. Therefore, M indicates the probability that a pixel corresponds to a word in the text prompt and can be utilized to locate the editing area.

Specifically, the noisy image $\hat{I}_t$ of a rendered view and the text prompt are fed into the diffusion model for denoising. We select the keyword that represents the intended editing results (e.g., "apron", "giraffe", "hat" as in Fig. 3) and extracts all its attention maps produced during the generation process. In practice, the backbone of the diffusion model usually consists of L convolutional blocks, which are equipped with H multi-headed attention layers [Vaswani

et al. 2017]. Therefore, after T rounds of denoising, the final set of attention maps $\mathbf{M}$ can be represented as $\{M_{t,l,h}\}$, where t, l, h represent the index of the time step, convolution block, attention head, respectively. We resize all attention maps to the same resolution by bilinear interpolation and aggregate them to obtain the aggregated attention map $\overline{M}$. $\overline{M}$ are further normalized to [0,1] and binarized with a threshold $\tau = 0.75$, where the area with a value of 1 is the editing area. We back-project all the pixels belonging to the editing area in the mask onto the mesh and mark the intersected mesh faces as the editing region. It is worth highlighting that the keywords are not restricted to the objects in the initial scene, as the attention maps of a keyword delineate regions in the generated image where the likelihood of keyword presence is highly probable. As shown in Fig. 6), even though "sunglasses" is not part of the original scene, it remains feasible to identify the reasonable region on the scene mesh.

In this stage, we traverse all elevation and azimuth angles at 45° intervals within the scope of sampled views to ensure the coverage of all potential editing regions. Subsequently, we get the masks of all sampled views and back-project them onto the mesh. After merging the results of back-projection, we employ two steps to refine the masked region: (1) Discard: we discard the small pieces within the editing region where the number of faces is less than 10% of the total projected area, which typically emerges from inaccuracy 2D masks (*e.g.,* masks larger than target object is projected outside the object); (2) Fill: we use breadth-first search to fill in the "holes" in the editing region, *i.e.,* a non-editing region surrounded by editing regions. Such "holes" usually come from occluded (*e.g.,* the bottom of a horse) or concave areas. By integrating these regions into the editing area, we enhance the completeness of the editing area. We denote the final editing region as $\mathbf{V} = \{v_e\}_{e=1}^{E}$.

4.4 Optimizing Editing Regions

In this step, we adopt the SDS Loss from DreamFusion [Poole et al. 2022] to guide the optimization of the editing region in the neural field with the T2I diffusion model, making the scene conforms to the text prompt. By feeding random rendered views and the text prompt to the T2I diffusion model, we calculate the SDS Loss and backpropagate the gradients to the neural field. Since the Imagen [Saharia et al. 2022] in DreamFusion is proprietary, we compute the SDS Loss in the latent space with Stable Diffusion [Rombach et al. 2022] as follows:

$$\nabla_{\omega}\mathcal{L}_{SDS}(\phi, g(\omega)) = \mathbb{E}_{\epsilon,t}\left[w(t)(\epsilon_{\phi}(z_t; y, t) - \epsilon)\frac{\partial z}{\partial \hat{I}}\frac{\partial \hat{I}}{\partial \omega}\right], \quad (6)$$

where $\omega = \{f_{g,k}, f_{c,k}, \mathbf{v}_k\}_k$ are the set of geometry features, color features and positions for all mesh vertices in $\mathbf{V}$, z_t denotes the noisy latent, and z is the original latent generated by the encoder of the Stable Diffusion model.

We can see from Equation 6 that during training, apart from optimization of the color feature f_c and geometry feature f_g of the vertices in the editing region, the positions of the vertices are also included. This implies that the structure of the mesh is also dynamically adjusted during the optimization, which is a critical part of our approach. In local implicit fields, geometry features mainly represent shape details near the vertices. The smoothness of the object's surface will be disrupted if there are significant changes in the s-density of the points situated away from the vertices. Hence, we propose a complementary optimization approach, which simultaneously optimizes the vertex position and geometry features. The optimization of the vertex position ensures that the overall shape of the mesh conforms to the text prompt, while the optimization of the geometry features refines the local geometry of the object. This optimization approach enables DreamEditor to generate complex shapes, such as rose petals. Our ablation study in Section 5.4 demonstrates the necessity of the joint optimization of the vertex position and geometry features.

To maintain a smooth surface and encourage natural deformation during vertex position optimization, we introduce widely-used mesh regularization terms, including the Laplacian loss and ARAP (as-rigid-as-possible) loss [Sumner et al. 2007]:

$$\mathcal{L}_{lap} = \frac{1}{E}\sum_{i=1}^{E}\left\|\mathbf{v}_i - \frac{1}{|N_i|}\sum_{j\in N_i}\mathbf{v}_j\right\|^2, \quad (7)$$

$$\mathcal{L}_{ARAP} = \sum_{i=1}^{E}\sum_{j\in N_i}\left|||\mathbf{v}_i - \mathbf{v}_j||_2 - ||\mathbf{v}'_i - \mathbf{v}'_j||_2\right|, \quad (8)$$

where N_i is the set of one-ring neighbours for vertex v_i, v' indicates the vertex position in the last iteration. We set $\lambda_{lap} = 10^{-4}$ and $\lambda_{ARAP} = 10^{-4}$ to balance them respectively.

We perform both the SDS Loss and mesh regularization terms during optimization in each iteration. We found that optimizing the SDS and regularization terms separately achieves better results empirically. Given a rendered view, we first optimize f_c, f_g, $\mathbf{v}$ of the editing region with the SDS loss. Then, f_c and f_g are fixed, and only $\mathbf{v}$ is optimized with the mesh regularization terms.

5 EXPERIMENTS

5.1 Experimental Setup

Dataset. To verify the effectiveness of our method in various scenes, we select six scenes with different levels of complexity from four datasets: DTU [Jensen et al. 2014], BlendedMVS [Yao et al. 2020], Co3D [Reizenstein et al. 2021], and GL3D [Shen et al. 2018]. These scenes include objects in simple backgrounds, human faces, and outdoor scenes with complex backgrounds. We use high-resolution images and the corresponding camera poses from the respective datasets to learn the original neural fields. Then, we edit the original scenes based on text prompts.

Baselines. We compare with three baselines. (1) D-DreamFusion*: as pointed out by Instruct-N2N, DreamFusion fails to edit a neural field due to the difficulty of finding an exact textual description that matches a scene. To learn a better neural representation of a specific scene, we combine Stable-DreamFusion with DreamBooth [Ruiz et al. 2022] as another baseline. (2) Instruct-NeRF2NeRF (Instruct-N2N): we also compare with a recent work Instruct-NeRF2NeRF and use the text instructions provided by the paper [Haque et al. 2023] to edit a 3D scene. (3) NeRF-Art [Wang et al. 2023]: Since NeRF-Art only supports stylized editing, we compare it in the stylization task.

Evaluation Criteria. Following [Haque et al. 2023], we use the CLIP Text-Image directional similarity to evaluate the degree of alignment between the change in both the images and text prompts

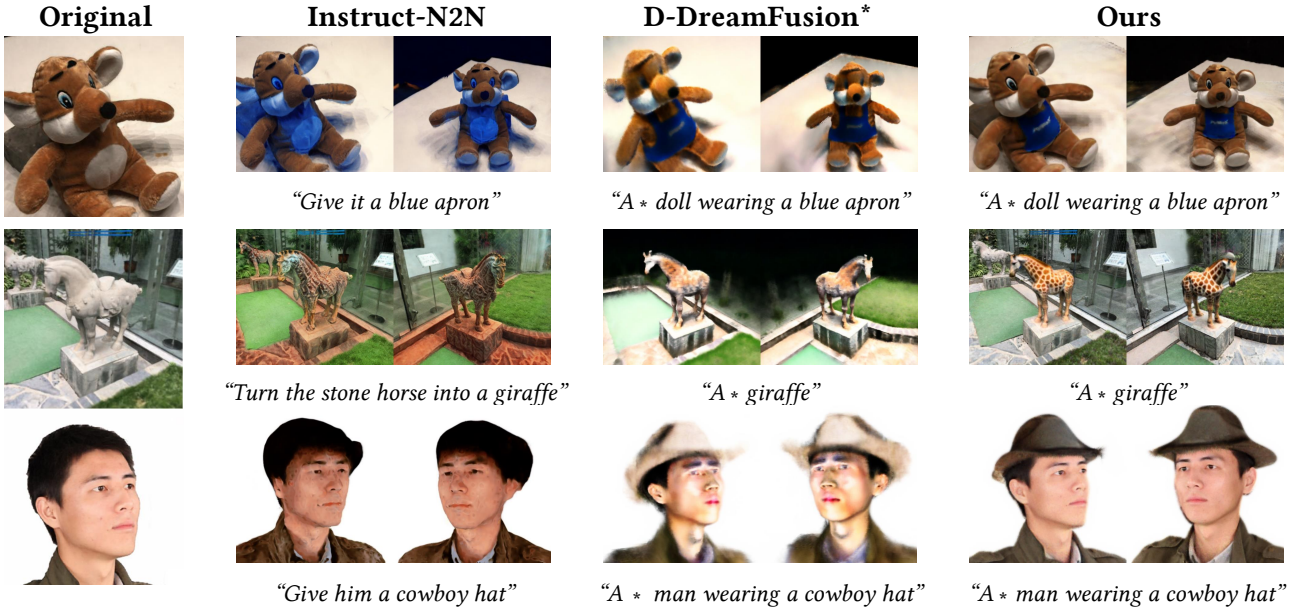

Figure 3: Visual results of our method compared with two baselines on three different scenes. The results clearly show that DreamEditor can precisely locate the relevant region, perform faithful editing to the text, and prevent undesirable modifications, which are difficult to be achieved by the baseline methods.

and its detailed definition can be found in [Gal et al. 2022b]. For each editing result, we uniformly sample 50 viewpoints around the editing region and take the mean value as the result. Since the CLIP directional similarity can only approximately evaluate the editing quality, we additionally conduct user studies to obtain human evaluations. We distribute 50 copies of questionnaires, presenting rotation video results of all methods side by side and asking users to choose the best editing result. The voting rates are calculated for each method. We compare our method with the aforementioned baselines in four selected scenes, covering a total of 20 distinct editing operations. We exclude NeRF-Art in the quantitative comparison due to it only supports stylized editing.

Implementation Details. In our experiments, we adopt NeuS to learn the original neural field. The training parameters can be found in [Wang et al. 2021]. As for the diffusion model, we use the public pretrained Stable Diffusion model V2. For each original neural field, we use the rendered images from the locating step, applying DreamBooth to fine-tune the Stable Diffusion model over 500 iterations. In the distilling step, we use the Adam optimizer with $lr = 10^{-4}$ to optimize the local fields for 100K iterations. In the optimizing step, the size of the rendered images is gradually increased from 96×96 to 192×192. We set the Adam optimizer with $lr = 10^{-2}$ to optimize the $f_c, f_g, \mathbf{v}$ of vertices in the editing region for 2K iterations. We implement our editing framework in Pytorch.

5.2 Qualitative Results

Results of Editing 3D Scenes. We provide qualitative results of our method in Fig.1 and Fig. 10. Results demonstrate that our method can effectively perform targeted editing of neural fields in various scenes. As depicted in the middle row of Fig.1, even in complex scenes such as outdoor gardens, our method can accurately determine the horse sculpture as the editing region, subsequently turning it into a deer or giraffe with high-quality textures and geometry. Moreover, our method is capable of local editing, such as wearing sunglasses for the dog in the bottom of Fig. 1. Notably, as shown in Fig. 7, the editing results produced by our method demonstrate excellent consistency in 3D geometry, as can be intuitively observed in the extracted mesh.

Fig.3 presents a comparison of the results of our method with baselines. Instruct-N2N has difficulties in executing abstract operations (e.g. give an apron to a doll) and generates suboptimal results in some scenes. This is largely attributed to the fact that the Instruct-Pix2Pix model is not always reliable, and it operates on the full image. Therefore, Instruct-N2N changes the entire scene and may underperform when executing the instructions beyond the Instruct-Pix2Pix training set. The DreamBooth finetuning in D-DreamFusion* enables the T2I diffusion model to roughly learn the representation of the input objects, such as the toy in the first row and the man in the third. However, due to the complexity and diversity of real-world scenes, D-DreamFusion* cannot accurately represent a specific scene, leading the irrelevant regions of the scenes edited by D-DreamFusion* to change significantly, such as the deformation of the doll in the first row, the background in the second row. Moreover, all compared baselines can not guarantee the consistency of the scenes before and after editing in complex scenes (such as the garden in the second row), and their editing process may change the entire scene. In contrast, our method has more details and faithfully generates the content of the text prompts, while successfully maintaining the consistency of the input objects and scenes before and after editing.

Table 1: Results of the CLIP Text-Image Direction Loss and user studies.

Method	CLIP Text-Image Direction Similarity ↑	Editing performance voting percentage ↑
D-DreamFusion*	12.43	12.1%
Instruct-N2N	10.86	6.8%
Ours	**18.49**	**81.1%**

Results of stylization task. As shown in Fig.8, we compare our method with NeRF-Art and Instruct-N2N. In this task, we omit the locating step to stylize the whole scene. Since stylization editing is a subjective task, we only provide the qualitative results as a reference and do not conduct quantitative analysis.

Results of locating editing region. In Fig.6, we also show our method's results of locating editing regions. We can see that our method can locate reasonable editing regions precisely.

5.3 Quantitative Results

In Table 1, we present the results of the CLIP text-to-image directional loss. The results clearly demonstrate that our method achieves significantly higher scores, indicating that our method generates shapes and textures that are clearer and more aligned with the edited text prompts. Additionally, our method receives over 81.1% of the votes, surpassing the other methods by a significant margin. This further demonstrates DreamEditor can achieve much higher user satisfaction across various scenes.

5.4 Ablation Study

Effectiveness of locating step. To demonstrate the necessity of locating step, we design two variants: (1) w/o locating: We omit the locating step and optimize all local implicit fields on the mesh. (3) Our method: we determine the editing region through locating step, and fix the non-editing region in optimization. As illustrated in Fig.4 (1), editing without the locating step will inadvertently change irrelevant regions of the scene, such as shortening the doll's arm, which destroys the consistency of the object. In contrast, the locating step allows our framework to optimize exclusively the region of interest.

Effectiveness of optimizing approach. To evaluate whether our optimizing approach can generate more detailed 3D shapes during optimization, we ablate with three variants of DreamEditor as follows: (1) Fixing $\mathbf{v}$: fixing the mesh structure during the updating process, only optimizing the geometry features. (2) Fixingf_g: only changing the mesh structure without optimizing the geometry feature. (3) Our method: $\mathbf{v}$ and f_g are optimized simultaneously. We select a challenging scene to evaluate: *generating a rose on a cup.*

We present the rendered images of the generated results and the extracted 3D shape using the marching cubes algorithm in Fig. 5. Fig.5 (1) displays the rose generated by fixing vertex positions, which are full of spikes. This is because, in regions far from the mesh surface, constraining the smoothness of the s-density of the sampling points across implicit fields is quite challenging. Fixing geometry features, as shown in Fig.5 (2), can generate a rough shape

Original (1) w/o locating (2) Ours

Figure 4: Ablation study of locating step. Editing without the locating step will deform the doll.

(1) Fixing $\mathbf{v}$ (2) Fixing f_g (3) Ours

Figure 5: Ablation study of optimizing approach. Obviously, our method generates red roses with more detailed and realistic 3D shapes.

but lacks details. In contrast, our method simultaneously optimizes both the geometric features and vertex positions, which eliminates the spikes as well as generates more detailed buds and petals.

6 CONCLUSION AND LIMITATIONS

In this paper, we present DreamEditor, a text-driven framework for editing 3D scenes represented by neural fields. Given a neural field and text prompts describing the desired edits, DreamEditor automatically identifies the editing region within the scene and modifies its geometry and texture accordingly. Experiments across a diverse range of scenes, including faces, objects, and large outdoor scenes, showcase the robust editing capabilities of DreamEditor to generate high-quality textures and shapes compared with other baselines while ensuring the edited scene remains consistent with the input text prompts.

Limitations of DreamEditor include the Janus problem, an issue inherited from DreamFusion, where the generated object appears as a front view from different viewpoints. Furthermore, DreamEditor does not directly model environmental lighting, which limits control over the lighting condition. While DreamEditor generally works well, due to the dependence of rendered views in editing, its performance may suffer in the presence of significant self-occlusions in the scene, consequently impacting the final synthesis results. Considering that NeuS faces difficulties in effectively reconstructing backgrounds in unbounded scenes, our current focus lies on object-centric editing in the foreground of the scene. In the future work, by combining recent unbounded real-world scene mesh reconstruction methods, such as BakedSDF [Yariv et al. 2023], our method can be extended to the whole scene editing.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (NO. 62322608, 61976250), in part by the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No.VRLAB2023A01), and in part by the Guangdong Basic and Applied Basic Research Foundation (NO. 2020B1515020048).

REFERENCES

Omri Avrahami, Dani Lischinski, and Ohad Fried. 2022. Blended diffusion for text-driven editing of natural images. In *CVPR 2022*. 18208–18218.

Chong Bao, Yinda Zhang, and Bangbang et al. Yang. 2023. Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In *CVPR 2023*. 20919–20929.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. 2022. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800* (2022).

Jianchuan Chen, Ying Zhang, Di Kang, Xuefei Zhe, Linchao Bao, Xu Jia, and Huchuan Lu. 2021. Animatable neural radiance fields from monocular rgb videos. *arXiv preprint arXiv:2106.13629* (2021).

Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. 2023. Fantasia3D: Disentangling Geometry and Appearance for High-quality Text-to-3D Content Creation. *arXiv preprint arXiv:2303.13873* (2023).

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

Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. 2022. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427* (2022).

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. 2022a. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618* (2022).

Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. 2022b. StyleGAN-NADA: CLIP-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.

William Gao, Noam Aigerman, Thibault Groueix, Vladimir G Kim, and Rana Hanocka. 2023. TextDeformer: Geometry Manipulation using Text Guidance. *arXiv preprint arXiv:2304.13348* (2023).

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099* (2020).

Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. 2023. Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions. *arXiv preprint arXiv:2303.12789* (2023).

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. 2022. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626* (2022).

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *NeurIPS 2020* 33 (2020), 6840–6851.

Ajay Jain, Ben Mildenhall, Jonathan T Barron, and et al. 2022. Zero-shot text-guided object generation with dream fields. In *CVPR 2022*. 867–876.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. 2014. Large scale multi-view stereopsis evaluation. In *CVPR 2014*. 406–413.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. 2022. Imagic: Text-based real image editing with diffusion models. *arXiv preprint arXiv:2210.09276* (2022).

Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. 2022. Decomposing nerf for editing via feature field distillation. *arXiv preprint arXiv:2205.15585* (2022).

Yuan Li, Zhi-Hao Lin, David Forsyth, Jia-Bin Huang, and Shenlong Wang. 2022. ClimateNeRF: Physically-based Neural Rendering for Extreme Climate Synthesis. *arXiv e-prints* (2022), arXiv–2211.

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. 2022. Magic3D: High-Resolution Text-to-3D Content Creation. *arXiv preprint arXiv:2211.10440* (2022).

Hao-Kang Liu, I Shen, Bing-Yu Chen, et al. 2022. NeRF-In: Free-form NeRF inpainting with RGB-D priors. *arXiv preprint arXiv:2206.04901* (2022).

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, and et al. 2020. Neural sparse voxel fields. *NeurIPS 2020* 33 (2020), 15651–15663.

Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. 2021. Editing conditional radiance fields. In *ICCV 2021*. 5773–5783.

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics* 21, 4 (1987), 163–169.

Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. 2022. Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures. *arXiv preprint arXiv:2211.07600* (2022).

Oscar Michel, Roi Bar-On, Richard Liu, and et al. 2022. Text2mesh: Text-driven neural stylization for meshes. In *CVPR 2022*. 13492–13502.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. 2022. CLIP-Mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022*. 1–8.

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

Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. 2021. Neural articulated radiance field. In *ICCV 2021*. 5762–5772.

Julian Ost, Issam Laradji, Alejandro Newell, and et al. 2022. Neural point light fields. In *CVPR 2022*. 18419–18429.

Sida Peng, Yuanqing Zhang, Yinghao Xu, and et al. 2021. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *CVPR 2021*. 9054–9063.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. 2022. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988* (2022).

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR 2017*. 652–660.

Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. 2023. DreamBooth3D: Subject-Driven Text-to-3D Generation. *arXiv preprint arXiv:2303.13508* (2023).

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* (2022).

Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, and et al. 2021. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *ICCV 2021*. 10901–10911.

Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. 2023. Texture: Text-guided texturing of 3d shapes. *arXiv preprint arXiv:2302.01721* (2023).

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *CVPR 2022*. 10684–10695.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2022. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *arXiv preprint arXiv:2208.12242* (2022).

Chitwan Saharia, William Chan, and Saurabh et al. Saxena. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS 2022* 35 (2022), 36479–36494.

Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. 2023. Vox-E: Text-guided Voxel Editing of 3D Objects. *arXiv preprint arXiv:2303.12048* (2023).

Tianwei Shen, Zixin Luo, Lei Zhou, and et al. 2018. Matchable image retrieval by learning from surface reconstruction. In *ACCV 2018*. Springer, 415–431.

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* (2020).

Robert W Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded deformation for shape manipulation. In *ACM siggraph 2007 papers*. 80–es.

Ashish Vaswani, Noam Shazeer, Niki Parmar, and et al. 2017. Attention is all you need. *NeurIPS 2017* 30 (2017).

Can Wang, Menglei Chai, Mingming He, and et al. 2022a. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *CVPR 2022*. 3835–3844.

Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. 2023. Nerf-art: Text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics* (2023).

Chen Wang, Xian Wu, Yuan-Chen Guo, and et al. 2022c. NeRF-SR: High Quality Neural Radiance Fields using Supersampling. In *ACM MM 2022*. 6445–6454.

Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. 2022b. Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation. *arXiv preprint arXiv:2212.00774* (2022).

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689* (2021).

Fanbo Xiang, Zexiang Xu, Milos Hasan, and et al. 2021. Neutex: Neural texture mapping for volumetric neural rendering. In *CVPR 2021*. 7119–7128.

Tianhan Xu and Tatsuya Harada. 2022. Deforming radiance fields with cages. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*. Springer, 159–175.

Bangbang Yang, Chong Bao, and Junyi et al. Zeng. 2022. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *ECCV 2022*. Springer, 597–614.

Yao Yao, Zixin Luo, Shiwei Li, and et al. 2020. BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks. In *CVPR 2020*.

Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. 2023. BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. *arXiv preprint arXiv:2302.14859* (2023).

Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, and et al. 2022. NeRF-editing: geometry editing of neural radiance fields. In *CVPR 2022*. 18353–18364.

Figure 6: Visualization of the editing region, where the bold words indicate keywords and the red area on the mesh represents the editing region.

Figure 7: Visualization of the extracted mesh from our editing results.

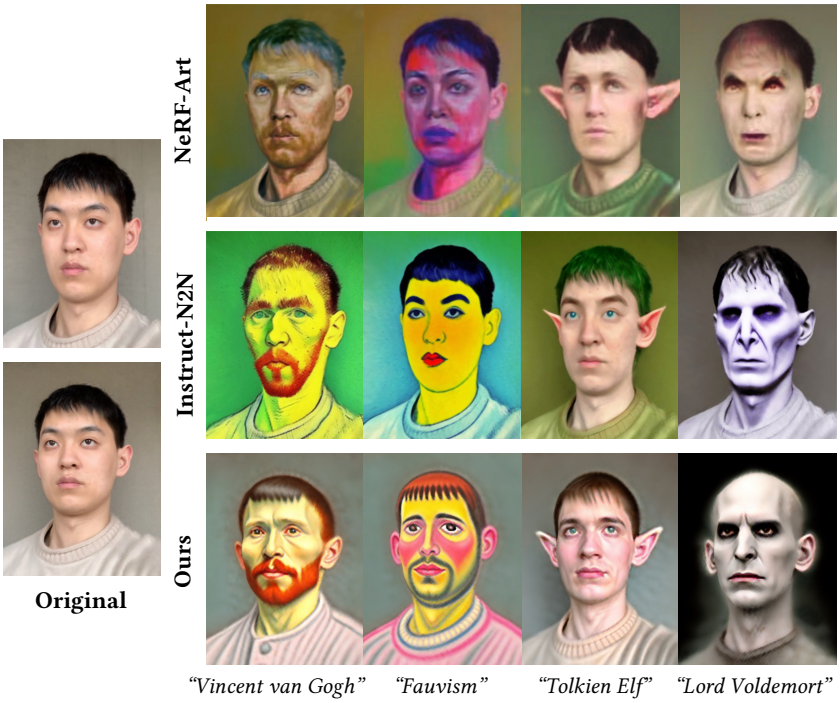

Figure 8: Visualization of the stylization editing results, we compare with NeRF-Art and Instruct-NeRF2NeRF.

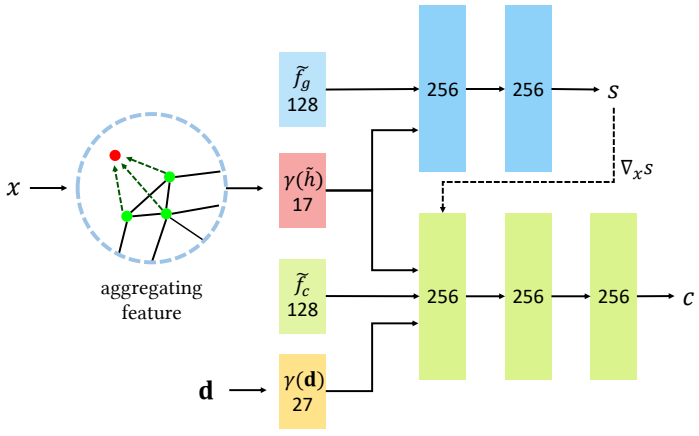

Figure 9: The network of the mesh-based neural fields. It takes the sampled point x and the ray direction d as input, output the s-density s and color c. $\gamma(\cdot)$ denotes positional encoding adopted in NeRF [Mildenhall et al. 2021].

Figure 10: More editing results.