Free-Editor: Zero-shot Text-driven 3D Scene Editing

Nazmul Karim*,[1], Hasan Iqbal*,[2], Umar Khalid*,[1], Chen Chen[1], and Jing Hua[2]

[1] University of Central Florida, Orlando, FL, USA
[2] Department of Computer Science, Wayne State University, Detroit, MI, USA

Abstract. Text-to-Image (T2I) diffusion models have recently gained traction for their versatility and user-friendliness in 2D content generation and editing. However, training a diffusion model specifically for 3D scene editing is challenging due to the scarcity of large-scale datasets. Currently, editing 3D scenes necessitates either retraining the model to accommodate various 3D edits or developing specific methods tailored to each unique editing type. Moreover, state-of-the-art (SOTA) techniques require multiple synchronized edited images from the same scene to enable effective scene editing. Given the current limitations of T2I models, achieving consistent editing effects across multiple images remains difficult, leading to multi-view inconsistency in editing. This inconsistency undermines the performance of 3D scene editing [3] when these images are utilized. In this study, we introduce a novel, training-free 3D scene editing technique called Free-Editor, which enables users to edit 3D scenes without the need for model retraining during the testing phase. Our method effectively addresses the issue of multi-view style inconsistency found in state-of-the-art (SOTA) methods through the implementation of a single-view editing scheme. Specifically, we demonstrate that editing a particular 3D scene can be achieved by modifying only a single view. To facilitate this, we present an Edit Transformer that ensures intra-view consistency and inter-view style transfer using self-view and cross-view attention mechanisms, respectively. By eliminating the need for model retraining and multi-view editing, our approach significantly reduces editing time and memory resource requirements, achieving runtimes approximately 20 times faster than SOTA methods. We have performed extensive experiments on various benchmark datasets, showcasing the diverse editing capabilities of our proposed technique. Project Page: https://free-editor.github.io/

1 Introduction

Neural Radiance Fields (NeRF) [26], neural implicits [43] as well as subsequent work [23, 28, 41], collectively termed as *neural fields*, have emerged as powerful 3D neural representations. Recent advances in this field [6, 7] have focused

* * Equal Contribution

[3] Here, 3D scene editing indicates NeRF model editing. In this study, we mainly focus on NeRF-based 3D scene representation.

on both novel view synthesis, scene reconstruction as well as 3D scene manipulations such as color editing [17, 18, 21, 39, 49], scene composition [36, 38], and style transfer [10, 14]. Notably, it has been shown that text-guided 3D NeRF [12, 49] editing can be achieved through leveraging the diverse generation capability of 2D text-to-image (T2I) diffusion models [3, 11, 32]. Despite their demonstrated success existing methods need to **i)** re-train the editing model for each particular 3D scene which introduces computational and memory overhead, and **ii)** rely on the prior knowledge of specific editing types, which may not be feasible in most scenarios. For instance, InstructNerf2Nerf (IN2N) [12] iteratively edits the training images of a scene until it obtains the desired editing result of the scene. The iterative editing is inevitable since all training images may not have consistent style information during initial iterations. This is due to the current limitations of T2I diffusion models as achieving prompt-consistent edits in multiple images (even if they are from the same scene) is very challenging. Fig. 1 shows an example where the same target prompt produces different multi-view outcomes within the scene. Such inconsistent edits or styles lead to poor 3D editing performance even with extensive re-training. To tackle this issue of *multi-view style inconsistency*, IN2N proposes to iteratively update the edited training set based on direct feedback from the NeRF model. Achieving the same goal as IN2N without re-training limits us from updating the training set more than once. This leads to poor performance due to the aforementioned issues.

Fig. 1: Multi-View Inconsistency in Current Text-to-Image (T2I) Editing Models: The current T2I editing model [3] faces significant challenges with multi-view consistency. This issue adversely affects the quality of 3D scene editing, especially when these edited views are used to synthesize novel views. This specific limitation is also acknowledged in IN2N [12]. Note that this inconsistency is particularly problematic when editing is performed *without re-training*, which aligns with our objectives.

In our work, we tackle the problem of *text-driven 3D scene editing* from a fresh perspective. Given a 3D scene data with multiple source views with their pose information, we randomly choose a *starting view*. Our objective is to edit the entire 3D scene by editing only this starting view. By editing only one view per scene, we eliminate the possibility of multi-view style inconsistency (Fig. 1) while reducing the overall editing time significantly. In addition, we address the problem of re-training with the help of a generalized NeRF (Gen-NeRF) model [33, 42]. Specifically, we leverage the style information in the starting view and multi-view geometry information from several unedited source views to render a novel edited target view with the same style as the starting view.

Table 1: Comparison with SOTA. Unlike prior works, our proposed method does not re-train the model each time we have to edit a new scene.

Methods	Re-Training	Text-Driven	Style Transfer
Blend-NeRF [19]	✓	✗	✗
Blended-NeRF [9]	✓	✓	✓
DreamEditor [49]	✓	✓	✓
NeRF-Art [40]	✓	✓	✓
Instruct-N2N [12]	✓	✓	✓
Ours	✗	✓	✓

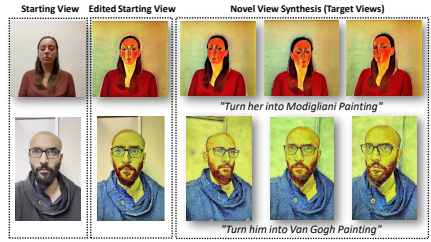

Fig. 2: 3D Scene Editing using our proposed method for different target poses.

We argue that zero-shot 3D scene editing can easily be achieved by introducing a few key architectural design changes to the Gen-NeRF.

These architectural changes are necessary as there are a few obvious challenges in achieving our goal: *First*, transferring style information from the starting view to the target view requires the view geometry correspondence information. In our framework, the view-geometry correspondence is solved by leveraging pixel-aligned features for each target pixel. To further enhance these features (obtained from unedited source views) with style information, we utilize an *Edit Transformer (ET)* that employs both self-view and cross-view attentions. While self-view attention helps us grasp long-range content information within the starting view, we can enrich the pixel-aligned features with content details from the starting view with cross-view attention. Subsequently, these style-informed features, obtained from ET, can easily be converted into RGB color using widely used Epipolar and Ray transformers [35,46]. *Second*, features obtained from ET lack necessary spatial awareness as closely situated neighboring views from the same scene should change continuously. This may lead to spatial non-smoothness in the pixel space which is highly undesirable for style transfer. To tackle this, we design a multi-view consistency loss that encourages the features corresponding to two spatially close points to be similar. In addition, we employ *self-view robust loss* to obtain consistent color in the final edited scene. Our main contributions can be summarized as follows:

- We propose Free-Editor, a zero-shot text-guided 3D scene editing technique that can synthesize edited novel views based on a text description while maintaining high 3D consistency. By introducing several novel modifications to a generalized NeRF model, our proposed method eliminates the requirement of scene-specific retraining across various editing styles.
- We propose an *Edit Transformer* to facilitate intra-view consistency and inter-view style transfer, enabling us to edit a particular 3D scene using a single edited view (see Fig. 2). This *single-view editing scheme* can effectively remove the bottleneck of *multi-view style inconsistency* in SOTA methods.
- Due to the unique design choices in Free-Editor, both training costs as well as editing time are reduced significantly. Extensive evaluation on multi-

ple datasets with various editing styles demonstrates the superiority of our proposed method.

2 Related Work

Novel View Synthesis With NeRF. was first introduced in [26], generates realistic novel views by fitting scenes as continuous 5D radiance fields using a Multilayer Perceptron (MLP). Since its inception, several advancements have enhanced NeRFs. For instance, Mip-NeRF [1, 2] efficiently handles object scale in unbounded scenes. Nex [45] models significant view-dependent effects whereas other works [29, 43] improve surface representation, extend to dynamic scenes [30], etc. Despite the tremendous success of NeRF, its time-consuming per-scene fitting poses a notable drawback. To address this, Generalizable NeRFs aim to bypass this optimization hurdle by framing new view creation as an image-based interpolation challenge across different views. Approaches like NeuRay [24], IBRNet [44], MVSNeRF [4], and PixelNeRF [47] construct a universal 3D representation using combined features from observed views. GPNR [35] and GNT [42] elevate the quality of generated new views through a Transformer-based aggregation method. In our work, we also consider Generalized NeRF as we do not aim to re-train the model.

Diffusion-based 3D Scene Editing. The emergence of text-to-image conversion models has notably impacted NeRF editing. Beginning with the Score Distillation Sampling method in DreamFusion [31], Vox-e [34] explored techniques to regulate alterations in pre-existing voxel fields. NeRF-Art [40] utilizes various regularization approaches during training to ensure that NeRF when edited using CLIP, preserves the original structure. InstructNerf2Nerf (IN2N) employed 2D image translation models [12] to adjust 2D image characteristics for NeRF training based on textual prompts. However, IN2N's reliance on IP2P [3] for updating NeRF training data tends to excessively modify scenes. Furthermore, encounter difficulties such as extended training durations and unstable loss functions. Addressing the challenge of undesired alterations, D-Editor [49] introduced a mesh-based neural field that efficiently converts 2D masks into 3D editing areas. This enables precise local modifications while avoiding unnecessary geometric changes when altering only the appearance. Similarly, Blended-NeRF [9] and Blend-NeRF [19] require additional cues like bounding boxes for localized editing. However, all of these methods require per-scene adaptation of the 3D model to induce any editing effects which increases computational overhead significantly. In our work, we propose a zero-shot editing technique that performs similarly or better than SOTA with more practical applicability.

3 Method

3.1 Background

Neural Radiance Fields. In neural radiance fields (NeRF) [26], the task is to find a neural network-based representation of 3D scenes. The neural network

here is a multi-layer perceptron (MLP) that maps a 3D location $\boldsymbol{x} \in \mathbb{R}^3$ and viewing direction $\boldsymbol{d} \in \mathbb{S}^2$ to an emitted color $\boldsymbol{c} \in [0,1]^3$ and a volume density $\sigma \in [0, \infty)$,

$$\mathcal{F}(\boldsymbol{x}, \boldsymbol{d}; \boldsymbol{\Theta}) \quad \mapsto \quad (\boldsymbol{c}, \sigma)\,, \tag{1}$$

where $\mathcal{F}$ and $\boldsymbol{\Theta}$ represent MLPs and the set of learnable parameters, respectively.

Volume Rendering. Let us define $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$ as a ray in a NeRF, where $\boldsymbol{o}$ is the camera center and $\boldsymbol{d}$ is the ray's unit direction vector. Along this ray, we can predict the color values $\boldsymbol{c}_i$ and volume densities σ_i of K sample points, $\{\boldsymbol{r}(t_i) \,|i = 1, ..., K\}$, by following this formal procedure:

$$\begin{aligned} \hat{C}(\boldsymbol{r}) &= \sum_{i=1}^{K} w_i \boldsymbol{c}_i, \quad \text{where} \\ w_i &= \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)\left(1 - \exp\left(-\sigma_i \delta_i\right)\right). \end{aligned} \tag{2}$$

Here, w_i indicates the weight or hitting probability of i-th sampling point [24] and δ_i is the distance between adjacent samples.

Text-to-3D Scene Editing Method, IN2N, operates by repeatedly updating the training dataset images using a diffusion model and then training the NeRF on these modified images to maintain a consistent 3D representation. This iterative approach allows the gradual integration of the diffusion priors into the 3D scene, enabling substantial edits. The image-conditioned diffusion model (IP2P [3]) helps preserve the original scene's structure and identity.

3.2 Free-Editor: Zero-shot Scene Editing

In this section, we describe our proposed method *Free-Editor*, a training-free approach for 3D scene editing without the requirements of *iterative updates of dataset* and *per-scene optimization*. Consider a dataset that contains L number of 3D scenes in terms of images and their camera intrinsic and extrinsic parameters. Let us define a 3D scene training data that contains N images with their corresponding camera parameters, $\{I_l \in \mathbb{R}^{H \times W \times 3}, P_l \in \mathbb{R}^{3 \times 4}\}$. First, we select a *starting view* I_0 and render a *target view* I_t. To apply specific 2D editing in I_t and I_0, we employ a text-guided diffusion model $\mathbf{D}$ with group attention,

$$\hat{I}_0 = \mathbf{D}(I_0, C^0_{in}, C^0_{tgt}, \Phi), \hat{I}_t = \mathbf{D}(I_t, C^t_{in}, C^0_{tgt}, \Phi), \tag{3}$$

where Φ is the diffusion model, C^0_{in} and C^t_{in} are input captions of I_0 and I_t, respectively. On the other hand, C^0_{tgt} is the target caption used for editing. As the next step, we perform *source-view selection* to select M source views, $S = \{I_m, P_m\}_{m=1}^{M}$, from the remaining source views $\{I_n, P_n\}_{n=1}^{N-1}$, where $M < N-1$. Details of selecting M views are in Sec. 3.3. The generalized NeRF model [16,24] with parameters θ, $\mathbf{G}(.;\theta)$, takes $\{I_m, P_m\}_{m=1}^{M}$ and $\hat{I}_0, P_0$ as inputs and predicts $\tilde{I}_t$ as output,

$$\tilde{I}_t = \mathbf{G}(\hat{I}_0, I_m, P_0, P_m, \theta | m = 1, \dots M) \tag{4}$$

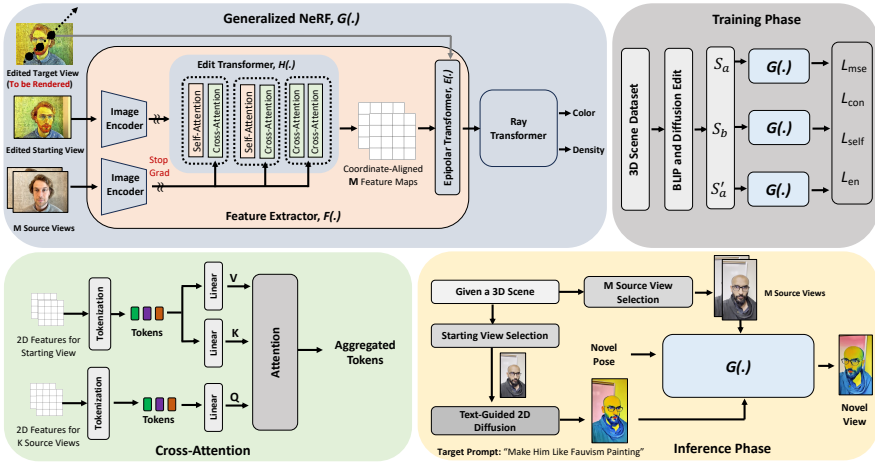

Fig. 3: Overview of our proposed method. ***Top Left.*** We train a generalized NeRF ($\mathbf{G}(.)$) model that takes an *edited starting view* and M source views to render a novel target view. Here, the *edited target view* is not the input to the model, rather will be rendered and works as the ground truth for the model output. In $\mathbf{G}(.)$, we employ a novel *Edit transformer* that utilizes: ***Bottom Left.*** cross-view attention to produce style-informed source feature maps that will be aggregated through an Epipolar transformer. ***Top Right.*** During training, we employ different sets of source views S_a, S_b, S'_b for 4 different loss functions. Note that S'_a is a variant of S_a with additional ray information for calculating $\mathcal{L}_{con}$. ***Bottom Right.*** During inference, only a single image needs to be edited to obtain a 3D-edited scene.

To train $\mathbf{G}(.;\theta)$, we minimize the following optimization objective,

$$\arg\min_{\theta} \mathcal{L}_{tot}(\theta; \hat{I}_t, \tilde{I}_t). \tag{5}$$

Style-aware Multi-view Feature Extraction. In Figure 3, we show the details of our proposed method. We can consider the generalized NeRF as a combination of feature extractor ($\boldsymbol{F}$) to extract coordinate-aligned feature maps before aggregating them and ray transformer ($\boldsymbol{R}$) [42] that will transform the features into color and density. Using $\boldsymbol{F}$), we adopt a feed-forward fashion to extract generalized features from multi-view images $\hat{I}_0, \{I_m, P_m\}_{m=1}^{M}$ and convert them later into 3D representation using $\boldsymbol{R}$. Eq. 1 shows how a 3D location can be mapped into color and density. Here, we do that in two stages: i) create a coordinate-aligned feature field using $\boldsymbol{F}$), and ii) aggregate point-wise features along different rays to form ray colors using an attention-based ray transformer. We use $\hat{I}_t$ as the ground truth for ray colors.

We first construct hierarchical image features using pre-trained 2D CNN Image Encoder [13] as follows, $\boldsymbol{f}_i = \boldsymbol{T}(I_i); m = 1, \dots M$, where $\boldsymbol{f}_i$ is the image feature for i^{th} source image. We then feed these features to the newly proposed edit transformer, $\boldsymbol{h}_i = \boldsymbol{H}(\hat{\boldsymbol{f}}_0, \boldsymbol{f}_i)$, which would give us the editing-informed multi-view feature maps. Here, $\hat{\boldsymbol{f}}_0$ is the 2D image feature corresponding to

the starting view, $\hat{I}_0$. We use both self-attention and cross-attention in our edit transformer (shown in Fig. 3) with different purposes. The functional mechanism of attention can be defined as $\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d}}) \cdot V$, where

$$Q = W^Q z, K = W^K z, V = W^V z. \tag{6}$$

Here, W^Q, W^K, and W^V denote trainable matrices that project the inputs to the query (Q), key (K), and value (V) components, respectively. z represents the latent features, and d represents the output dimension of the key and query features. The objective of self-attention is to learn long-range and relevant information within a given view. In our particular case, we aim to capture the exact editing effects that have taken place in $\hat{I}_0$ utilizing self-attention. However, capturing only single-view information using self-attention is not enough as it is required to have multi-view feature maps for a successful novel view synthesis. To understand why multi-view information is needed, we briefly analyze the recently proposed Epipolar Aggregated Transformer [35,46] which functions between the target pixels and pixels positioned on the epipolar line of multiple source views. Using the epipolar geometry constraint, the features can be aggregated to capture long-range content information within and across images. However, vanilla multi-view feature maps obtained from M source views do not have the editing information in $\hat{I}_0$. We find a simple fix to this issue by explicitly injecting editing information from $\hat{I}_0$ into the multi-view source feature maps with the help of cross-attention. As shown in Fig. 3, we first tokenize the 2D image features obtained from $\boldsymbol{T}$ which reduces the attention complexity significantly. To extract the key (K) and value (V), tokens from the starting view are used whereas we use source view tokens as the query (Q). Using cross-attention, we aggregate features from source views towards $\hat{I}_0$.

Now, for each target pixel in $\hat{I}_t$, we uniformly sample P coordinate-aligned 3D points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P\}$ from the set of points between far and near planes. Each of these points is projected into the feature maps obtained from $\boldsymbol{H}$ and then aggregated to form a coordinate-aligned feature field as follows,

$$\boldsymbol{F}(x_p, \phi) = \boldsymbol{E}(\boldsymbol{h}_1(\Pi_1(\boldsymbol{x}_p))), \ldots, \boldsymbol{h}_M(\Pi_M(\boldsymbol{x}_p))). \tag{7}$$

Here, $\boldsymbol{E}$ is the Epipolar Aggregated Transformer, $\Pi_i(\boldsymbol{x}_p)$ projects 3D point $\boldsymbol{x}_p$ onto the *i-th* source-image plane with the help of an extrinsic matrix. We use bilinear interpolation to compute the feature vector $\boldsymbol{h}_1(\boldsymbol{u})$ at projected 2D position $\boldsymbol{u} \in \mathbb{R}^2$. Finally, we utilize the Ray Transformer $\boldsymbol{R}$ along with an MLP to dynamically learn the blending weights along the ray for each point and produce the RGB color information. Given ray information, $\boldsymbol{r}$ and $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P\}$ as the uniformly sampled points along $\boldsymbol{r}$, an MLP ($\boldsymbol{V}$) can be used to map the pooled features vectors from $\boldsymbol{R}$ to RGB color $\tilde{\boldsymbol{C}}_t$,

$$\tilde{\boldsymbol{C}}_t(\boldsymbol{r}) = \boldsymbol{V}(\boldsymbol{R}(\boldsymbol{F}(\boldsymbol{x}_p, \phi))); \; p \in [1, P]. \tag{8}$$

For training the model end-to-end, we employ different loss functions that serve important roles in producing good performance.

3.3 Training Objectives and Data Generation

Photometric Loss, $\mathcal{L}_{mse}$. First, we adopt the photometric loss in NeRF [26] which is defined as the mean square error (MSE) between the predicted and ground truth pixel colors,

$$\mathcal{L}_{mse} = \sum_{r \in \mathcal{R}} ||\tilde{\boldsymbol{C}}_t(\boldsymbol{r}) - \hat{\boldsymbol{C}}_t(\boldsymbol{r})||^2, \tag{9}$$

where $\mathcal{R}$ represents the set of rays and $\hat{\boldsymbol{C}}_t(\boldsymbol{r})$ is the ground truth pixel values in $\hat{I}_t$ for ray $\boldsymbol{r} \in \mathcal{R}$.

Multi-view Consistency Loss, $\mathcal{L}_{con}$. As we are editing only a single image to edit an entire 3D scene, achieving spatial smoothness is challenging due to the constraints of view geometry in NeRF. To tackle this issue, we introduce a multi-view consistency loss to encourage a smooth transition between texture or color between neighboring views. Let us denote the feature distribution obtained at 3D point $\boldsymbol{x}_p^j$ as $\boldsymbol{e}_p^j = \{\boldsymbol{h}_1(\Pi_1(\boldsymbol{x}_p^j))), \ldots, \boldsymbol{h}_M(\Pi_M(\boldsymbol{x}_p^j))\}$, obtained from M source views. Here, the point $\boldsymbol{x}_p^j$ is sampled along the ray, $\boldsymbol{r}_j$. Let us select another ray $\boldsymbol{r}_{j'}$ which is very close to $\boldsymbol{r}_j$. For each point $\boldsymbol{x}_p^j$, we select its closest point $\boldsymbol{x}_p^{j'}$ along the ray $\boldsymbol{r}_{j'}$ based on their Euclidean distance, denoted as $d_{j,j'}^p = ||\boldsymbol{x}_p^j - \boldsymbol{x}_p^{j'}||$. To encourage consistency among the coordinate-aligned features of the closest points, we employ Jensen-Shannon Divergence (JSD) loss,

$$\mathcal{L}_J(\boldsymbol{x}_p) = JSD(\boldsymbol{e}_p^j || \boldsymbol{e}_p^{j'}), \tag{10}$$

where $\boldsymbol{e}_p^{j'}$ is the features corresponding to $\boldsymbol{r}_{j'}$. We use JSD loss for its symmetric nature which offers some notable advantages over Kullback–Leibler (KL) divergence [15] loss. Since closer points in the pixel space should have smaller distances in the feature space, we employ a weighted JSD loss for defining our final multi-view consistency loss,

$$\mathcal{L}_{con} = \sum_{p=1}^{P} \omega_p \mathcal{L}_J(\boldsymbol{x}_p). \tag{11}$$

Here, ω_p indicates the weight corresponding to the pair $(\boldsymbol{x}_p^j, \boldsymbol{x}_p^{j'})$ which can be expressed as $\omega_p = \frac{e^{-d_{j,j'}^p}}{\sum_{p=1}^{P} e^{-d_{j,j'}^p}}$. Our unique formulation of $\mathcal{L}_{con}$ imposes consistency on 3D points across various viewpoints, inherently promoting smoothness in the scene's geometry.

Self-View Robust Loss, $\mathcal{L}_{self}$. In general, when the training data for a 3D scene remains coherent, generating the same target view using different source views usually produces consistent outcomes. However, this may not hold true for our case as we are dealing with an edited target view. To address this, we choose two different sets of source views $S_a = \{I_m^a, P_m^a\}_{m=1}^M$ and $S_b = \{I_m^b, P_m^b\}_{m=1}^M$. The predicted target views utilizing S_a and S_b are $\hat{I}_t^a$ and $\hat{I}_t^b$, respectively which

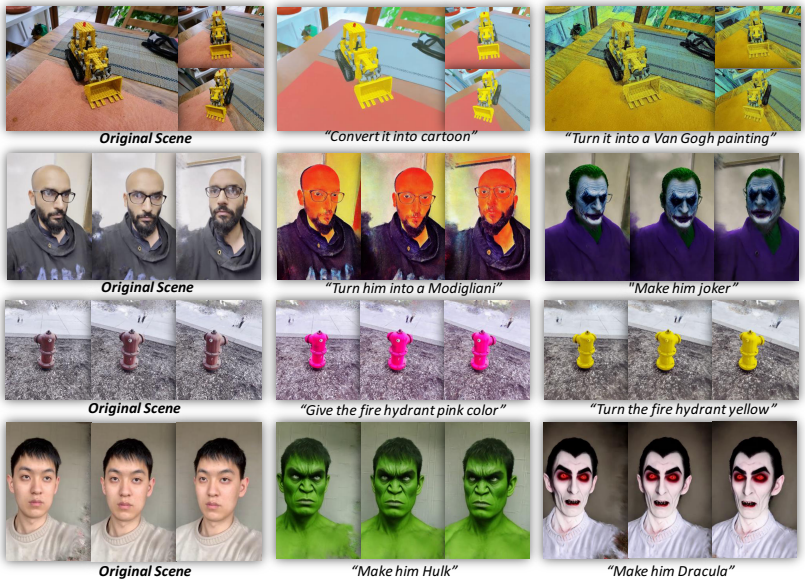

Fig. 4: Text-driven 3D scene editing. Illustration of text-driven 3D scene editing using our proposed method across various target poses. This figure showcases the view-consistent results generated by our method. A qualitative evaluation on multiple scenes reveals the efficacy of our approach: starting from a single view, our method successfully generates novel views that are conditioned on the editing prompt, demonstrating its robustness and versatility in 3D scene editing.

should have consistent content information. To ensure this consistency, we employ

$$\mathcal{L}_{self} = \sum_{r \in \mathcal{R}} ||\tilde{\boldsymbol{C}}_t^a(\boldsymbol{r}) - \tilde{\boldsymbol{C}}_t^b(\boldsymbol{r})||^2, \tag{12}$$

where $\tilde{\boldsymbol{C}}_t^a$ and $\tilde{\boldsymbol{C}}_t^b$ indicate RGB color values in $\hat{I}_t^a$ and $\hat{I}_t^b$, respectively.

Entropy Loss, $\mathcal{L}_{en}$. In addition, we consider an entropy loss for regularizing the hitting probabilities of the sampled points [20],

$$\mathcal{L}_{en} = -\sum w_i \log(1 - w_i). \tag{13}$$

Finally, the total loss function employed to train our framework can be expressed as follows,

$$\mathcal{L}_{tot} = \mathcal{L}_{mse} + \lambda_c \mathcal{L}_{con} + \lambda_s \mathcal{L}_{self} + \lambda_e \mathcal{L}_{en}, \tag{14}$$

where λ_c, λ_s, are λ_e the loss coefficients.

Training Data Generation. From each scene, we first select a target view and then identify a pool of $m.(M+1)$ nearby views (m is sampled uniformly at random from [1, 3]), among which a randomly sampled view is chosen as the

Table 2: Quantitative assessment of 3D scene editing focusing on text alignment and frame consistency is conducted. Our method exceeds all other state-of-the-art editing techniques in terms of Edit PSNR metric while achieving comparable performances in other scenarios. We use LLFF dataset here.

Metrics	C-NeRF	NeRF-Art	IN2N	DreamEditor	Ours
Edit PSNR	22.15	20.89	22.26	22.34	**22.47**
CTDS	0.2375	0.2503	**0.2804**	0.2788	0.2601
CDC	0.9672	0.9751	**0.9882**	0.9850	0.9781

Table 3: PSNR comparison with recent SOTA generalized NeRF methods. Here, *LLFF-E* indicates the performance on the edited LLFF dataset.

Method	LLFF	LLFF-E
PixelNeRF	18.66	11.03
MVSNeRF	21.18	16.74
IBRNet	25.17	19.05
Neuray	25.35	18.31
GeoNeRF	**25.44**	18.98
Free-Editor (Ours)	24.61	**22.47**

starting view while M other views are the source views. This sampling approach of m mimics diverse view densities during the training process, enhancing the network's ability to generalize across different view densities. We get the RGB images I_0 and I_t corresponding to the starting and target views, respectively. For editing I_0 and I_t, we utilize the open-source pre-trained models BLIP [22] and IP2P [3]. The BLIP model produces the input caption C^0_{in} of the *starting view* I_0. Later, we employ a GPT model to generate C^0_{tgt} by modifying C^0_{in}. In addition to GPT, we apply manually designed prompts as well. For example, C^0_{tgt} can be generated by simply following this format- *"**X** painting of C^0_{in} "*. ***X*** can be chosen from ["Leonardo da Vinci", "Sam Francis", "Max Ernst", "Henri Matisse", "Eva Hesse", "Carl Andre", "Cy Twombly"]. Finally, we feed C^0_{tgt} and I_0 to IP2P to produce $\hat{I}_0$. For $\hat{I}_t$, we generate multiple edited copies and select the one as the ground truth (for G) that has the highest CLIP consistency score with $\hat{I}_0$. In our work, we use e_n randomly chosen C^0_{tgt} for each scene where e_n is set to be 6. We do not choose a higher value for e_n as our objective is not to learn all types of editing available but rather how to transfer the edits from the *starting view* to other views. During training, we randomly select M from a uniform distribution of [8, 12]. More on this in the *supplementary*.

Inference Phase. During inference, we use different sets of scenes and target captions (C^0_{tgt}) than the training stage. This is to ensure that *Free-Editor* is generalizable in terms of both scenes and editing prompts. To edit a test scene, we first randomly select I_0 that will be edited and $M(= 12)$ source images. We then pass a novel target pose (close to the M source views, but not necessarily close to the I_0) to render an edited target view. As *Free-Editor* can consistently transfer the edits from one *starting view* to all other views, we can easily edit a 3D scene by editing a sufficient number (e.g. 70–80) of target views. Note that our proposed method edits any particular target view only once, in contrast to the iterative edits proposed in IN2N.

4 Experiments and Analysis

Datasets. Our model is trained on datasets including Google Scanned Objects [5], NerfStudio [37], Spaces [8], and IBRNet-collect [44], and

RealEstate10K [48]. For evaluation, we use IN2N [12], NeRF-Synthetic [27], LLFF [25], and our own dataset of four scenes.

Training Details. is in *supplementary*.

Baselines. We report qualitative and quantitative comparisons against four baseline NeRF editing methods including IN2N [12], NeRF-Art [40], C-NeRF [39], and DreamEditor [49]. The default 2D-image editing model is Instruct-Pix2Pix (IP2P) [3]

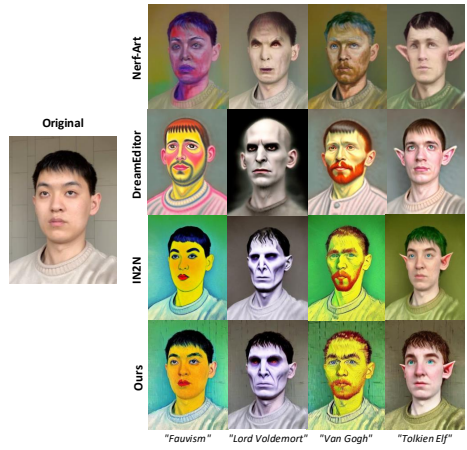

Fig. 5: Style Transfer Comparison. Exhibiting proficiency in conducting style edits within 3D NeRF Scenes, our method exemplifies its versatility and precision through intricate modifications and advanced prompt-guided editing in a three-dimensional environment. Visually, our outcomes resemble those of IN2N, since both methods utilize IP2P for 2D image editing. However, our method tends to preserve background details more effectively than IN2N.

4.1 Qualitative Results

Figures 4, 5 demonstrate the proficiency of our method in executing effective style edits, maintaining 3D coherence, and conforming to textual narratives.

Text-driven 3D Scene Editing. Figure 4 shows the text-driven editing performance of our proposed method. We use different text-scene pairs to show the diversity of our method. We start with the edited starting view and produce the novel target view from different poses. Our method shows notable alignment between the provided text description and the resulting views. Yet, there are instances where the alignment isn't perfect. For instance, when attempting to transform an image into Modigliani style, the details around the nose might not be accurately captured. This kind of intricate detail can be challenging to represent when working within significant limitations during the editing process.

Style Transfer. Figure 5 illustrates the visual comparisons between Free-Editor and other 3D scene style transfer methods. Owing to the use of the IP2P [3] editing backbone, our editing outcomes closely resemble those of IN2N [12]. However, the distinct advantage of our approach lies in its training-free nature, setting it apart from others. A notable distinction is that our method tends to preserve background details more effectively than IN2N [12], which often struggles to maintain this aspect. Comparisons with top-tier methods further corroborate the effectiveness of our technique and show the adeptness of our method in capturing both the color palette and stroke patterns of the desired style. Furthermore, our results are more realistic and preserve non-targeted areas, facilitating multiple sequential edits. Details are in *supplementary*.

Table 4: An ablation with the number of source views, M. A higher value of M produces slightly better performance before the performance saturates at a certain point. LLFF dataset has been used.

M	Edit PSNR	CTDS	CDC
3	20.94	0.2386	0.953
4	21.68	0.2471	0.962
6	22.09	0.2548	0.972
8	22.38	0.2563	0.975
10	22.44	0.2590	0.977
12	22.47	0.2601	**0.978**
18	**22.52**	**0.2604**	0.978

Table 5: Quantitative ablation study with different loss functions. Self-view robust loss impacts the color consistency while L_{con} impacts the smooth transfer of color information. Our own scenes have been used for this study.

Use Cases	w/o L_{con}	w/o L_{self}	All Loss
Case 1	20.98	19.63	**22.76**
Case 2	22.86	21.52	**23.11**
Case 3	23.14	22.29	**24.21**
Case 4	23.08	21.81	**23.96**
Case 5	21.37	20.13	**22.56**
Case 6	21.25	20.38	**22.03**

4.2 Quantitative Results

3D Scene Editing. The chosen metrics for the editing evaluation are CLIP Text-Image Directional Similarity (CTDS) and CLIP directional consistency (CDS), which serve as indicators of how effectively each method preserves 3D consistency across edited scenes. CTDS evaluates how well the executed 3D edits correspond to the text instructions. While CDS is akin to CTDS it assesses the similarity in direction between the original and edited images in successive frames along newly generated camera paths. Additionally, *Edit PSNR* compares the cosine similarity and PSNR between each rendered view from the edited and input NeRF. These metrics collectively provide insight into the integrity of the 3D scene post-editing. Table 2 showcases quantitative evaluations on the LLFF dataset across diverse scene editing tasks. Our findings indicate that both IN2N and our proposed approach produce outcomes consistent with the original viewpoints, demonstrated by their CLIP directional scores. Notably, the proposed method outperforms IN2N in terms of Edit PSNR, signifying better preservation of scene consistency with the original input. This suggests that our method maintains the details of the initial scene while implementing edits more effectively. Despite these advantages, our method slightly lags behind IN2N in overall performance, likely due to the zero-shot nature of Free-Editor. However, our findings underscore the strength of the proposed method in preserving scene integrity during editing.

Generalization Capabilities. Although the goal of our work is scene editing, we employed a generalized NeRF to achieve zero-shot capabilities. Therefore, it necessitates us to validate our method in generalization tasks too. For this, we consider recent generalized NeRF methods such as pixelNeRF [47], IBRNet [44], MVSNeRF [4], Neuray [24], GeoNeRF [16]. We modify the original forward-facing LLFF data dataset, LLFF-E based on diffusion-based editing, e.g. color or attribute changes. Table 3 shows our findings for this particular experiment. For the regular LLFF dataset, Free-Editor obtains slightly worse performance than previous methods which is somewhat understandable as it is developed mostly for scene editing. However, these methods severely underperform when

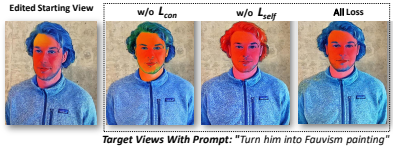

Fig. 6: Loss Sensitivity. An ablation study, to show the impact of different loss functions on the final performance.

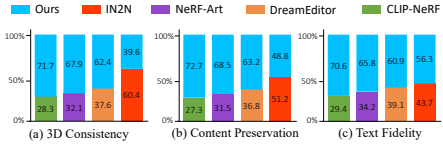

Fig. 7: In an extensive **user study** that assessed three evaluation metrics, our approach demonstrated comparable performance to IN2N [12]

we evaluate them on LLFF-E. This shows that one can not just use an off-the-shelf generalized NeRF for editing purposes. It also shows the necessity of developing a proper technique to obtain a generalizable method with editing capabilities. More on this is in *supplementary*

4.3 Ablation Studies

In this section, we perform an ablation study with different loss functions and the efficiency of Free-Editor.

Effect of Different Loss Functions. We study the impact of different loss functions on the overall performance of our proposed method. Table 5 shows 6 different use cases (scenes) where we apply 4 different text prompts for each scene. It can be observed that L_{self} has the most impact on the editing performance. The reason behind this is the compromised generalizability of the model without L_{self}. On the other hand, L_{con} helps us obtain better spatial smoothness, which can be also shown by the qualitative comparison we present in Fig. 6. Note that the impact of L_{con} wears off when we increase M.

Table 6: Runtime efficiency comparison of different methods. We take 2 of our scenes and apply 10 different editing before averaging.

Method	PSNR (dB)	Edit-time (mins.)	Time Complexity	Space Complexity
Clip-NeRF	21.15	1034.2	O(n)	O(n)
NeRF-Art	21.64	780.6	O(n)	O(n)
Instruct-N2N	22.98	62.1	O(n)	O(n)
DreamEditor	**23.18**	70.5	O(n)	O(n)
Ours	23.06	**3.2**	O(n)	**O(1)**

Model Efficiency. One of the main goals is to edit a particular 3D scene within a realistic timeframe. Table 6 shows that we are close to achieving that goal. Free-Editor obtains almost 20× better runtime efficiency compared to the previous SOTA. Our proposed approach reduces the total editing time while obtaining better space efficiency, leading to a constant space complexity of $O(1)$. On the contrary, previous methods necessitate the retraining of a model for each distinct scene or editing type, resulting in increased space complexity $O(n)$.

Effect of M. In Table 4, we study the impact of different numbers of source views. It can be observed that our proposed method can produce similar performance even with very few source views. The performance trade-off is not meaningful for $M > 12$.

4.4 User Study

We conducted a user study to observe the acceptability of our method in comparison to other leading-edge methods. This study involved a broad participant base, resulting in a total of 1000 responses across three critical evaluation criteria: the 3D spatial coherence, the retention of the original scene's elements, and the accuracy in reflecting the given textual descriptions. The outcomes of this user survey are visually represented in Figure 7. These results indicate a preference for results generated by our method and IN2N [12], highlighting Free-Editor's proficiency in these key areas. Detailed information regarding the methodology and execution of this user study is provided in the *supplementary*.

5 Discussion and Limitations

One potential solution to the issue of *multi-view inconsistency* within the same scene can be through *trial and error*. Specifically, we can generate a particular set of edited images and then observe the rendering performance. Since we are using a pre-trained 2D diffusion model, this process can be repeated until we get our desired editing effects in the target view. However, it may take hundreds of *trial and error* iterations before achieving a reasonable performance on the edited scene. Therefore, developing an efficient method for 3D scene editing is necessary and makes practical sense.

Limitations. Since we depend on the 2D image pre-editing process [3] for such edits, *multi-view inconsistency* could still be an issue here. To tackle this, we can use the CLIP consistency score to see whether we have a good match between the starting and target views. However, the probability of inconsistent edits between 2 views is much lower as compared to the scenario where we need to edit all training images. Another limitations of our work is to heavily focusing on style transfer as there are other area of interests such as object addition or removal. We leave this to the future studies of our work. Another limitation could be that for complex and large scenes, the model may need fine-tuning, not full training, for the desired performance.

6 Conclusion

We proposed a zero-shot text-driven 3D scene editing technique that does not require any re-training. Although the issue of re-training can be addressed by training a generalized NeRF model, the produced features do not contain the necessary editing information. To overcome this, our proposed edit transformer can effectively transfer the style information to the rendered target views through cross-attention. In addition, multi-view consistency loss and self-view robust loss are employed to further enhance spatial smoothness and color consistency. Our method offers not only diverse editing capabilities but also considerable benefits in processing speed and storage efficiency when compared with prior methods with requirements of retraining for individual scenes or modifications.

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
3. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: Learning to follow image editing instructions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18392–18402 (2023)
4. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14124–14133 (2021)
5. Downs, L., Francis, A., Koenig, N., Kinman, B., Hickman, R., Reymann, K., McHugh, T.B., Vanhoucke, V.: Google scanned objects: A high-quality dataset of 3d scanned household items. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 2553–2560. IEEE (2022)
6. Fang, S., Wang, Y., Yang, Y., Xu, W., Wang, H., Ding, W., Zhou, S.: Pvd-al: Progressive volume distillation with active learning for efficient conversion between different nerf architectures. arXiv preprint arXiv:2304.04012 (2023)
7. Fang, S., Xu, W., Wang, H., Yang, Y., Wang, Y., Zhou, S.: One is all: Bridging the gap between neural radiance fields architectures with progressive volume distillation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 597–605 (2023)
8. Flynn, J., Broxton, M., Debevec, P., DuVall, M., Fyffe, G., Overbeck, R., Snavely, N., Tucker, R.: Deepview: View synthesis with learned gradient descent. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2367–2376 (2019)
9. Gordon, O., Avrahami, O., Lischinski, D.: Blended-nerf: Zero-shot object generation and blending in existing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2941–2951 (2023)
10. Gu, J., Liu, L., Wang, P., Theobalt, C.: Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. arXiv preprint arXiv:2110.08985 (2021)
11. Han, L., Li, Y., Zhang, H., Milanfar, P., Metaxas, D., Yang, F.: Svdiff: Compact parameter space for diffusion fine-tuning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7323–7334 (2023)
12. Haque, A., Tancik, M., Efros, A.A., Holynski, A., Kanazawa, A.: Instruct-nerf2nerf: Editing 3d scenes with instructions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19740–19750 (2023)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. Höllein, L., Johnson, J., Nießner, M.: Stylemesh: Style transfer for indoor 3d scene reconstructions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6198–6208 (2022)

15. Huszár, F.: How (not) to train your generative model: Scheduled sampling, likelihood, adversary? arXiv preprint arXiv:1511.05101 (2015)
16. Johari, M.M., Lepoittevin, Y., Fleuret, F.: Geonerf: Generalizing nerf with geometry priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18365–18375 (2022)
17. Karim, N., Khalid, U., Joneidi, M., Chen, C., Rahnavard, N.: Save: Spectral-shift-aware adaptation of image diffusion models for text-driven video editing. arXiv preprint arXiv:2305.18670 (2023)
18. Khalid, U., Iqbal, H., Karim, N., Hua, J., Chen, C.: Latenteditor: Text driven local editing of 3d scenes. arXiv preprint arXiv:2312.09313 (2023)
19. Kim, H., Lee, G., Choi, Y., Kim, J.H., Zhu, J.Y.: 3d-aware blending with generative nerfs. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 22906–22918 (2023)
20. Kim, M., Seo, S., Han, B.: Infonerf: Ray entropy minimization for few-shot neural volume rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12912–12921 (2022)
21. Kobayashi, S., Matsumoto, E., Sitzmann, V.: Decomposing nerf for editing via feature field distillation. arXiv preprint arXiv:2205.15585 (2022)
22. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: International conference on machine learning. pp. 19730–19742. PMLR (2023)
23. Liu, L., Gu, J., Zaw Lin, K., et al.: Neural sparse voxel fields. NeurIPS 2020 **33**, 15651–15663 (2020)
24. Liu, Y., Peng, S., Liu, L., Wang, Q., Wang, P., Theobalt, C., Zhou, X., Wang, W.: Neural rays for occlusion-aware image-based rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7824–7833 (2022)
25. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019)
26. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
27. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
28. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
29. Oechsle, M., Peng, S., Geiger, A.: Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5589–5599 (2021)
30. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021)
31. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=FjNys5c7VyY

32. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22500–22510 (2023)
33. Sajjadi, M.S., Meyer, H., Pot, E., Bergmann, U., Greff, K., Radwan, N., Vora, S., Lučić, M., Duckworth, D., Dosovitskiy, A., et al.: Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6229–6238 (2022)
34. Sella, E., Fiebelman, G., Hedman, P., Averbuch-Elor, H.: Vox-e: Text-guided voxel editing of 3d objects. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 430–440 (2023)
35. Suhail, M., Esteves, C., Sigal, L., Makadia, A.: Generalizable patch-based neural rendering. In: European Conference on Computer Vision. pp. 156–174. Springer (2022)
36. Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8248–8258 (2022)
37. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., et al.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. pp. 1–12 (2023)
38. Tang, J., Chen, X., Wang, J., Zeng, G.: Compressible-composable nerf via rank-residual decomposition. Advances in Neural Information Processing Systems **35**, 14798–14809 (2022)
39. Wang, C., Chai, M., He, M., et al.: Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In: CVPR 2022. pp. 3835–3844 (2022)
40. Wang, C., Jiang, R., Chai, M., He, M., Chen, D., Liao, J.: Nerf-art: Text-driven neural radiance fields stylization. IEEE Transactions on Visualization and Computer Graphics (2023)
41. Wang, C., Wu, X., Guo, Y.C., et al.: Nerf-sr: High quality neural radiance fields using supersampling. In: ACM MM 2022. pp. 6445–6454 (2022)
42. Wang, P., Chen, X., Chen, T., Venugopalan, S., Wang, Z., et al.: Is attention all nerf needs? arXiv preprint arXiv:2207.13298 (2022)
43. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
44. Wang, Q., Wang, Z., Genova, K., Srinivasan, P.P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4690–4699 (2021)
45. Wizadwongsa, S., Phongthawee, P., Yenphraphai, J., Suwajanakorn, S.: Nex: Real-time view synthesis with neural basis expansion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8534–8543 (2021)
46. Yang, Z., Ren, Z., Shan, Q., Huang, Q.: Mvs2d: Efficient multi-view stereo via attention-driven 2d convolutions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8574–8584 (2022)
47. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4578–4587 (2021)

48. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. arXiv preprint arXiv:1805.09817 (2018)
49. Zhuang, J., Wang, C., Lin, L., Liu, L., Li, G.: Dreameditor: Text-driven 3d scene editing with neural fields. In: SIGGRAPH Asia 2023 Conference Papers. pp. 1–10 (2023)