NeRFEditor: Differentiable Style Decomposition for 3D Scene Editing

Chunyi Sun, Yanbin Liu, Junlin Han, Stephen Gould
Australian National University
{chunyi.sun, yanbin.liu, junlin.han, stephen.gould}@anu.edu.au

Abstract

We present NeRFEditor, an efficient learning framework for 3D scene editing, which takes a video as input and outputs a high-quality, identity-preserving stylized 3D scene. Our goal is to bridge the gap between 2D and 3D editing, catering to a wide array of creative modifications such as reference-guided alterations, text-based prompts, and user interactions. We achieve this by encouraging a pre-trained StyleGAN model and a NeRF model to learn mutually consistent renderings. Specifically, we use NeRF to generate numerous (image, camera pose)-pairs to train an adjustor module, which adapts the StyleGAN latent code for generating high-fidelity stylized images from any given viewing angle. To extrapolate edits to novel views, i.e., those not seen by StyleGAN pre-training, while maintaining 360° consistency, we propose a second self-supervised module that maps these views into the hidden space of StyleGAN. Together these two modules produce sufficient guidance for NeRF to learn consistent stylization effects across the full range of views. Experiments show that NeRFEditor outperforms prior work on benchmark and real-world scenes with better editability, fidelity, and identity preservation.

1. Introduction

Imagine if we can take a short video and generate an editable 3D scene. This ability will facilitate interesting applications in game and movie products, e.g., freely editing an identity-preserving character in real scene according to various user demands. Current techniques require 3D modeling expertise and long development times per scene, which is infeasible for real-time and customized editing.

Motivated by this, existing attempts push current techniques towards 3D consistent and real-time editing, as summarized in Tab. 1. Using existing latent space manipulation techniques, 2D GANs [8, 13, 25] can produce stylized images from multiple camera poses. However, these 2D methods are confined to the training pose distribution (*in-domain poses*) and cannot ensure 3D consistency. 3D-aware synthesis methods [1, 2, 7, 20] can generate multiview-consistent images using unstructured 2D images for training. Since their primary goal is not free-view editing, they experience noticeable quality degradation when extrapolating to unseen camera views (*out-of-domain poses*). Directly editing the neural radiance field (NeRF) is a promising direction for full scene editing. However, existing NeRF editing methods [15, 34] only support basic shape and color editing of simple objects. These approaches demand diverse 3D real-scene data for training, which is expensive to obtain.

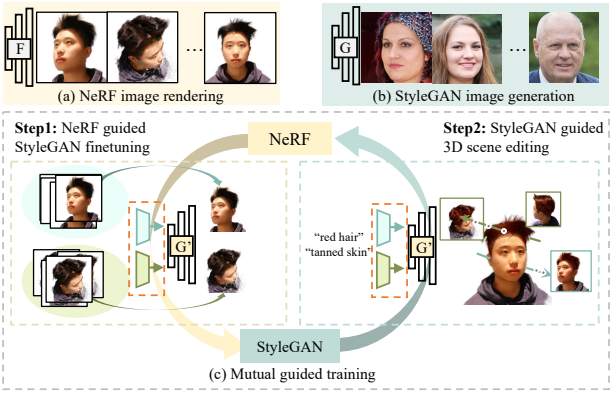


Figure 1. **Method overview.** (a) NeRF can render images from different views but focuses on a single scene. (b) StyleGAN can generate high-quality images but lacks generation control and is restricted to a small range of viewing angles. (c) We propose two modules (highlighted in the orange box) for a pre-trained StyleGAN to generate highly controllable stylized images from in-domain views and out-of-domain views, which can provide guided images to fine-tune the NeRF to achieve 3D scene editing.

In this paper, we present *NeRFEditor*, an efficient learning framework for 3D editing of real scenes, which supports diverse editing types to produce high-fidelity, identity-preserving scenes. Our framework leverages the novel-view synthesis capability of NeRF and the well-behaved latent space of a pre-trained StyleGAN model (Fig. 1). The former ensures 3D-consistent scene editing, while the latter facilitates flexible manipulation and high-quality generation. However, it is not straightforward to incorporate a 3D rendering model (NeRF) and a 2D generative model (StyleGAN) into a unified learning framework.

To generate view-consistency guided stylized images from any camera view, we design two additional learnable modules for StyleGAN. First, a *latent code adjustor* is devised that takes camera pose as an input and re-renders the image from the target view by manipulating the latent code of StyleGAN. In the latent code adjustor, we introduce a differentiable decompositor to decompose the pre-trained StyleGAN latent space $\mathcal{W}$ into an orthogonal basis. Then, the adjustor disentangles the pose from other styles to generate multiview stylized images to guide NeRF rendering. Second, to maintain 3D style consistency across different views (i.e., editing human clothing color in the frontal view while preserving a coherent 3D style across all angles), it is essential to address the limitation of StyleGAN in generating images for poses outside its training domain. We aim to enable the adaptation of in-view edited styles to out-of-domain poses. To achieve this, we introduce an innovative component termed the *hidden mapper*. This module maps novel-view images to the hidden feature space of StyleGAN. We develop a self-supervised training algorithm for the hidden mapper so that StyleGAN can adapt its generative ability to out-of-domain views (i.e., camera poses not seen during StyleGAN training).

For 3D-consistent scene editing, we use the above-generated in-domain and out-of-domain stylized images to finetune an adapted conditional NeRF model. The resulting model can extrapolate a wide variety of 2D edits to 3D scenes, such as text-prompt editing, image-guided editing, interactive editing, and style transfer. It outperforms prior work on standard evaluation metrics and image fidelity.

To sum up, we make the following contributions:

- We propose an efficient framework for 3D scene editing, which only takes a few minutes on a single GPU. A summary of the features of our method over existing approaches is given in Tab. 1.
- We design a differentiable latent code adjustor to disentangle camera pose from other styles, combining NeRF and pre-trained StyleGAN into a unified learning framework.
- We devise a self-supervised hidden mapper to facilitate the out-of-domain style adaptation.

Our method achieves state-of-the-art results on a public benchmark dataset and a newly-collected real-scene human dataset. Moreover, it shows promising application potentials for diverse high-quality editing types.[1]

2. Related Work

2D Latent Space Manipulation. Previous 2D GAN manipulation works [8, 33, 35] show that the latent space of pre-trained GANs can be decomposed to control the image generation process for attribute editing. The viewing direction can also be disentangled from other attributes, which allows a GAN to produce images from different viewing directions. However, since the training dataset only covers a limited range of viewing angles, both the supervised [14, 25, 27, 32] and unsupervised [8, 26, 33, 35] manipulation methods struggle to make accurate and out-of-domain control of viewing directions. Moreover, these 2D methods have difficulties in generating a 3D-consistent scene. In contrast, we design a mutual NeRF-StyleGAN learning framework, which inherits the 3D-consistent ability of NeRF and maintains the latent space manipulation property of StyleGAN. Our framework can also extrapolate the editing capability beyond the training viewing distributions with a novel self-supervised hidden mapper.

Table 1. Compare the main features with previous works.

Method	360° Editing	w/o 3D Auxiliary data	Real image Editing	3D Consistent	Real-time Editing
2D GANs [8, 13, 25]	✗	✓	✓	✗	✓
3D-aware [1, 2, 7, 20]	✗	✓	✗	✓	✗
NeRF editing [15, 34]	✓	✗	✗	✓	✗
Ours	✓	✓	✓	✓	✓

3D-aware Synthesis. Recent 3D-aware synthesis methods rely on NeRF [16] to generate 3D-consistent images using unstructured 2D training images. As a pioneering work, pi-GAN [1] represents the implicit neural radiance field using a SIREN network [28] and applies the FiLM [23] conditioning on SIREN, leading to view-consistent synthesis. FENeRF [29], EG3D [2] and StyleNeRF [7] take a two-step procedure: 1) condition a NeRF on viewpoint and style to render a low-resolution feature map; 2) convert the map to a high-resolution image with a CNN-based generator. These methods encounter two problems that prevent their direct application to real 3D scene editing: poor out-of-domain view extrapolation and inaccurate GAN inversion. In our method, the first problem is addressed by a well-devised hidden mapper that can generate out-of-domain stylized images. For the second problem, our proposed differentiable adjustor can improve the GAN inversion.

Editable NeRF. NeRF [16] encodes the radiance field of a scene in the weights of a MLP to conduct novel-view rendering. As an implicit function, NeRF is challenging to edit. Existing works [15, 34, 37, 41] only support editing on local parts of objects or with simple types. EditNeRF [15] was the first work to edit the shape and color of NeRF on local parts of simple objects. CLIP-NeRF [34] improves EditNeRF by leveraging a CLIP model to support text prompt or exemplar images, but still on simple objects. Compared with them, our method can edit complex scenes and objects (e.g., human face) to generate high-fidelity 3D-consistent images and support more diverse editing types.

3D style transfer aims at editing the 3D NeRF scene to

[1]Coda and dataset available from https://github.com/Chuny1/NeRFEditor

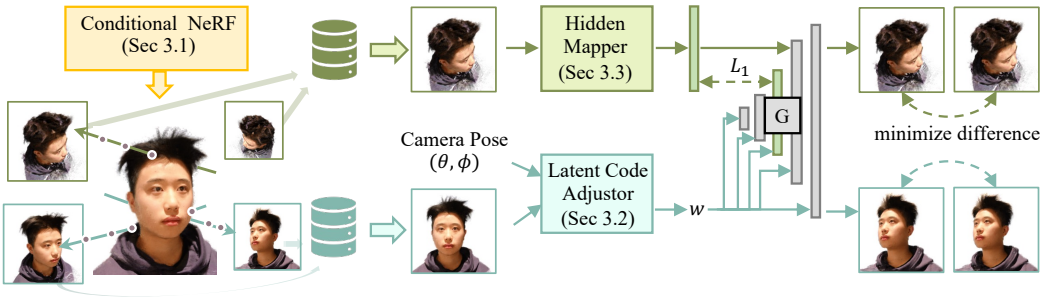


Figure 2. The framework of NeRFEditor. (1) The ***conditional NeRF*** is pre-trained on the original scene to produce sufficient (image, pose) pairs. (2) The pairs are used to train a ***Latent Code Adjustor***, which disentangles camera pose from other styles in the learned latent space of StyleGAN. (3) To enable out-of-domain extrapolation, a ***Hidden Mapper*** maps novel-view images to the hidden feature space of StyleGAN. Finally, conditional NeRF will be finetuned for 3D-consistent editing (Sec. 3.4).

match the style of a reference image. Previous methods [3, 11, 18, 39] usually design a style transfer loss to guide the rendered images from different views, which might result in unstable optimization. For our method, although the main focus is 3D scene editing, it can perform style transfer for any object category as a by-product. Meanwhile, our method achieves more stable 3D style transfer than the previous method tailored to this task.

3. Mutual NeRF-StyleGAN Training

As shown in Fig. 2, we design an efficient learning framework for 3D scene editing, which leverages the 3D consistency strength of NeRF and latent space manipulation ability of StyleGAN. To facilitate mutual training, we adapt a conditional NeRF (Sec. 3.1) and devise two novel modules for StyleGAN: a latent code adjustor (Sec. 3.2) and a hidden mapper (Sec. 3.3). Finally, the conditional NeRF is finetuned for 3D-consistent style editing (Sec. 3.4).

3.1. Conditional NeRF

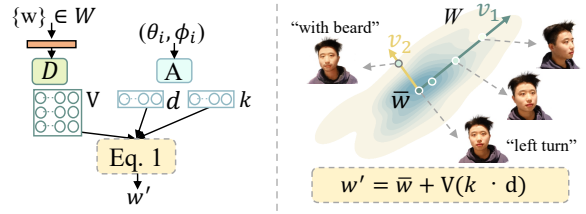
We start by adapting a conditional NeRF model to (1) produce (image, pose) pairs for style manipulation in StyleGAN and (2) use manipulated images to effectively finetune the NeRF model for 3D consistent scene editing (Sec. 3.4).

The NeRF model is a continuous function $\mathbf{F}$ that maps 3D coordinates $\mathbf{x} = (x, y, z)$, and viewing direction $\mathbf{v} = (\theta, \phi)$ to colors $\mathbf{c} = (r, g, b)$ and density σ. To enable the interaction with StyleGAN for scene editing, we condition NeRF on a style code $\boldsymbol{\alpha}$ and an appearance code $\boldsymbol{\beta}$. We assign different style codes for the original scene ($\boldsymbol{\alpha} = 0$) and stylized scene ($\boldsymbol{\alpha} = 1$) and assign unique appearance codes $\boldsymbol{\beta}$ for each different image in the stylized guided set[2]. The style code allows us to train the NeRF conditioned on different styles and the appearance code could help further handle the appearance inconsistency in the stylized guided set. Here, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are encoded into latent space by trainable MLPs $\mathcal{E}_{\text{style}}(\cdot)$ and $\mathcal{E}_{\text{app}}(\cdot)$; $\mathbf{x}$ is encoded by a hash encoder $\mathcal{H}(\cdot)$ [17]; and $\mathbf{v}$ is encoded by position encoding $\mathcal{R}(\cdot)$. The colors $\mathbf{c}$ and density σ are predicted as follows:

$$\mathbf{F}_{\text{dens}} : \left(\mathcal{E}_{\text{style}}(\boldsymbol{\alpha}), \mathcal{E}_{\text{app}}(\boldsymbol{\beta}), \mathcal{H}(\mathbf{x})\right) \mapsto (\mathbf{f}, \sigma),$$
$$\mathbf{F}_{\text{color}} : \left(\mathcal{E}_{\text{style}}(\boldsymbol{\alpha}), \mathcal{E}_{\text{app}}(\boldsymbol{\beta}), \mathcal{R}(\mathbf{v}), \mathbf{f}\right) \mapsto \mathbf{c},$$

where $\mathbf{F}_{\text{dens}}$, $\mathbf{F}_{\text{color}}$ denote the density and color sub-functions of $\mathbf{F}$, $\mathbf{f}$ is the output hidden feature from $\mathbf{F}_{\text{dens}}$.

3.2. Latent Code Adjustor



(a) Latent Code Adjustor (b) Style Disentangling

Figure 3. **Latent Code Adjustor.** (a) All sampled latent code in $\mathcal{W}$ are passed to an MLP (orange box) and decomposed by $\mathbf{D}$ to get an orthogonal basis V. An adjustor $\mathbf{A}$ predicts the view-related coordinates k of V and their strengths d. Then, an original code $\bar{w}$ is adjusted to w'. (b) An example of how the adjustor works. Suppose the learned latent space $\mathcal{W}$ only contains two styles: *"left turn"* and *"with beard"*. Hence, $\mathcal{W}$ is decomposed into two directions v_1 and v_2. Traversing over v_1 only modifies pose.

After obtaining sufficient (image, pose) pairs, we employ a pre-trained StyleGAN to generate high-fidelity stylized images from multiple camera views. Recent works [8, 25] found that pre-trained StyleGAN has a well-behaved latent space, which involves interpretable styles such as pose, color, expression, etc. Motivated by this, we devise a differentiable decomposition module, named *latent code adjustor*, to disentangle the camera pose from other styles, thereby enabling view-consistent image stylization. To ensure high-fidelity, we restrict the camera pose range to lie

[2]We use frame indexes generated by COLMAP and normalize to [0,1].

in StyleGAN's training pose distribution [1, 20, 29]. This range defines the in-domain camera views.

In StyleGAN, a mapping network $\mathbf{M}$ converts a vector $z \in \mathcal{Z}$ (sampled from a normal distribution) into a latent code $w \in \mathcal{W}$. The latent code w is then passed to a generator $\mathbf{G}$ to generate images. To manipulate a real-world image, GAN inversion methods are used to map the image to the latent code, and most methods [4, 10, 31] train an encoder $\mathbf{E}$ to map an image back to the latent code. However, the inverted latent code may not generate a nearly identical image, hindering high-fidelity manipulation. We add an MLP layer on the top of a pre-trained encoder $\mathbf{E}$ to refine the predicted latent code for better GAN inversion.

Differentiable Style Decomposition. The architecture of the latent code adjustor is shown in Fig. 3(a). A decomposition node $\mathbf{D}$ is applied on the intermediate latent space $\mathcal{W}$ to get an orthogonal basis V representing different styles. Specifically, we first sample a large batch of $z \sim \mathcal{Z}$ from the normal distribution and use the mapping network $\mathbf{M}$ to get a large set $\{w\} \subseteq \mathcal{W}$. We then compute the eigen-decomposition [5] of the estimated covariance matrix$\text{COV}(\{w\})$ to obtain the orthogonal basis V.

As discussed in [24], it is challenging for the inversion method to get a good latent vector that both captures the target image appearance and maintains edibility. Also, the pose attributes may not be fully disentangled in the pre-trained StyleGAN latent space $\mathcal{W}$. To tackle this issue, our additional MLP applied to latent codes w, allows the pre-trained StyleGAN $\mathcal{W}$ space to be gently fine-tuned. This is realized by differentiating the eigen-decomposition to make all modules trained end-to-end. We explicitly derive gradients using implicit differentiation [6] to efficiently back-propagate through eigen-decomposition.[3]

Latent Code Adjustment. Given the orthogonal basis V and a finetuned latent code w, image editing can be done by constructing a new latent code $w' = w + Vx$, where components of x are separate control parameters.

To perform target view editing, we need to determine the view-related coordinates and the corresponding adjusting strengths that need to traverse that coordinates to the target view. Given the target pose (θ, ϕ), we first use a lightweight classifier $\mathbf{C}$ to classify the view-related coordinates of x as $k = \text{sigmoid}(\mathbf{C}(\theta, \phi))$ and then use a lightweight regressor $\mathbf{R}$ to predict the corresponding view-adjusting strengths as $d = \mathbf{R}(\theta, \phi)$. Now, given a latent code w, we can generate the edited image $\widehat{I}^{(\theta,\phi)}$ of the target view by adjusting the latent code: $w' = w + V(k \cdot d)$ and $\widehat{I}^{(\theta,\phi)} = \mathbf{G}(w')$.

To train the latent code adjustor, we use the unedited frontal image $\bar{I}$ to obtain the latent code $\bar{w} = \mathbf{E}(\bar{I})$. With the generated target view image $\widehat{I}^{(\theta,\phi)}$ and the groundtruth $I^{(\theta,\phi)}$ generated by NeRF, we employ four loss terms for training $\mathbf{R}$, $\mathbf{C}$ and the MLP as follows:

$$L_{total} = L_2 + L_{vgg} + L_{id} + L_{reg}\,. \tag{1}$$

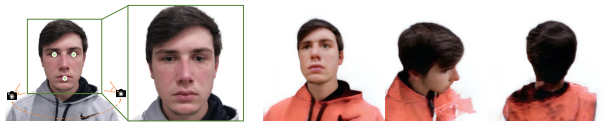
Here, the first three terms are the ℓ_2 distance, VGG perceptual distance [12], and identity distance computed between $\widehat{I}^{(\theta,\phi)}$ and $I^{(\theta,\phi)}$, respectively. The last term L_{reg} is an entropy regularization term to encourage k to be sparse, as only few coordinates are view-related.

In-domain Stylized Set $\mathcal{I}_{in}$. We can apply any existing latent code editing methods [8, 25] on the frontal latent code $\bar{w}$ to get w_{style}. We use the latent code adjustor to generate multi-view stylized images from a single frontal image. For the full set of in-domain images, we generate a stylized image for every camera pose in the NeRF-guided image set.

3.3. Hidden Mapper

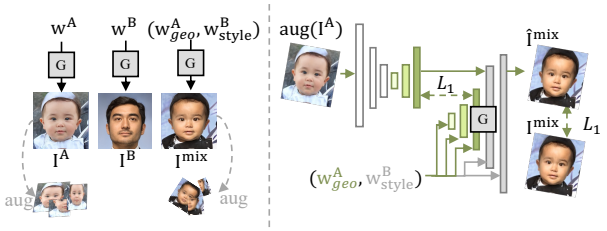
Despite the high-fidelity generation in the training pose distribution, StyleGAN struggles to extrapolate to extreme camera views beyond the training distribution, i.e., out-of-domain views. Fig. 4(a) demonstrates how data is processed when training the generative model. The image is aligned and cropped based on detected key points, and the camera manifold is restricted around the frontal face. These two factors decide the StyleGAN cannot generate out-of-domain view images and performs worse when approaching the boundary of the in-domain camera manifold. Fig. 4(b) shows that the hidden mapper generates guided

[3] Problem formulation and derivation are in the supplementary.



(a) StyleGAN in-domain camera manifold (b) Out of domain stylization generation

Figure 4. **Full scene editing.** (a) 2D alignment is applied to GAN's training data, and the training data is limited to a small range of camera views. (b) Our hidden mapper can make guided stylized images for out-of-domain views.



(a) Augmentation (b) Hidden Mapper Training

Figure 5. **Hidden Mapper.** (a) The codes are split as a geometric code w_{geo} and a style code w_{style}. Same augmentation is applied on hidden features controlled by w_{geo} and style-mixed images controlled by mixed codes (w^A_{geo}, w^B_{style}). (b) After augmentation, Hidden Mapper is trained with the L_1 reconstruction loss on both hidden space and image space.

Algorithm 1 Self-supervised Hidden Mapper Training.

Input: $\mathbf{G}_{geo}, \mathbf{G}_{style}$, image I^A, styles w^A, w^B
F^A = aug($\mathbf{G}_{geo}(w^A_{geo})$) # augment the hidden map
$\widehat{F}^A$ = $\mathbf{H}$(aug(I^A)) # predict the hidden map
I^{mix} = $\mathbf{G}_{style}(F^A, w^B_{style})$ # generate mixed image
$\widehat{I}^{mix}$ = $\mathbf{G}_{style}(\widehat{F}^A, w^B_{style})$ # predict mixed image
Return: loss = $L_1(I^{mix}, \widehat{I}^{mix}) + L_1(F^A, \widehat{F}^A)$

images without limitations on camera pose and helps to achieve 3D style consistency. We design a *hidden mapper* $\mathbf{H}$ to tackle this limitation of StyleGAN.

Self-supervised Training. Latent codes of shallow layers generally control the geometric aspects (e.g., pose and shape), while latent codes of deep layers control the style attributes (e.g., skin color). Style-mixing [13] can mix the style of two images by combining two latent codes. We sequentially split latent codes as (w_{geo}, w_{style}).

Each block of StyleGAN takes as input a hidden feature map produced by the previous block and an additional latent code. Therefore, we can perform style-editing by mixing the hidden maps with w_{style}. Most importantly, we observe that when we perform heavy augmentation to the hidden map, the style can still transfer to the correct spatial location in the image. If we can learn a function that maps images to the hidden space of StyleGAN, we will be able to transfer styles to images captured on any camera location. We introduce a self-supervised algorithm to achieve this goal.

We represent the early layers of the generator that takes w_{geo} as $\mathbf{G}_{geo}$ and the deeper layers of the generator that takes w_{style} as $\mathbf{G}_{style}$. The generation process is denoted as $F = \mathbf{G}_{geo}(w_{geo})$ and $I = \mathbf{G}_{style}(F, w_{style})$. The hidden mapper is trained to map real images to the output space of $\mathbf{G}_{geo}$, which is achieved by reconstructing both the hidden feature F and the generated image I. The training algorithm is shown in Alg. 1 and Fig. 5.

Obtaining Out-of-domain Images. Once the hidden mapper is trained we can easily generate out-of-domain stylized images by first using our conditional NeRF to generate an unedited image, which is then mapped into hidden features F. The hidden features are combined with target editing styles w_{style} to obtain the final stylized images.

3.4. 3D-Consistent Style Editing

With trained latent code adjustor and hidden mapper in hand, our model is capable of producing stylized images over complete 360° viewing angles. We use the images $\mathcal{I}_{ori}$ generated by the original scene, in-domain set $\mathcal{I}_{in}$ and out-of-domain stylized set $\mathcal{I}_{out}$ to finetune the conditional NeRF for 3D-consistent rendering with style edits.

To finetune conditional NeRF, we assign $\boldsymbol{\alpha} = 1$ for the stylized sets and a different $\boldsymbol{\beta} \in [0, 1]$ for each stylized image in $\mathcal{I}_{in}$ and $\mathcal{I}_{out}$. To focus on editing the objects, we calculate a foreground mask M. Then we define the following masked-guided losses w.r.t. colors $\mathbf{c}$ and density σ:

$$L_{style} = \sum_{i,j} \left(\widehat{\mathbf{c}}^{style}_{i,j} \cdot M_{i,j} - \mathbf{c}^{style}_{i,j} \cdot M_{i,j}\right)^2, \quad (2)$$

$$L_{br} = \sum_{i,j} \left(\widehat{\sigma}^{style}_{i,j} \cdot (1 - M_{i,j}) - \sigma^{ori}_{i,j} \cdot (1 - M_{i,j})\right)^2, \quad (3)$$

$$L_{ori} = \sum_{i,j} (\widehat{\sigma}^{ori}_{i,j} - \sigma^{ori}_{i,j})^2, \quad (4)$$

where i, j index the image pixels, $\widehat{\mathbf{c}}$ and $\widehat{\sigma}$ denote the rendered colors and density, ^{style} and ^{ori} denote the stylized and original sets. L_{style} optimizes the foreground style editing process. L_{br} is background regularization to constrain the density change in the background region. L_{ori} is used to avoid forgetting the original scene, which can stabilize the training and also help background regularization.

To further handle inconsistencies in the guided training set, we follow [19] to render the depth map $\mathbf{d}$ and adopt a depth regularisation term L_{depth} to smooth the stylized scene. The final loss used for our 3D-consistent NeRF editing is:

$$L = (L_{style} + L_{br} + L_{ori}) + \lambda \cdot L_{depth}\,, \quad (5)$$

where λ controls the strength of the depth estimation.

4. Experiments

Datasets. We evaluate *NeRFEditor* on two datasets: FaceScape [36, 41]and TIFace (Tiny-scale Indoor Face collection). FaceScape contains 359 different subjects, each with 20 expressions and 120 multiview images for each expression. Images are captured in a lab environment with the subject wearing a turban. To evaluate the editing capability for real 3D scenes, we collect TIFace, a real-world dataset including ten different persons from a realistic environment, without any restriction on hairstyle. Details of the dataset collection are in the supplementary.

Evaluation Metrics. Following previous works [1, 2, 7], we use various metrics to evaluate the image quality: PSNR, SSIM, and LPIPS [40]. To quantify the 3D consistency, we employ Pose (pose adjusting error) [2]. To evaluate identity preservation, we use ID Score [21], which measures the confidence of classifying one image to have the same identity as the ground truth image. Meanwhile, we use APS (attribute-preservation score) to measure the preservation of non-edited attributes after editing; the score is calculated by an attribute classifier [9] pre-trained on CelebA. To compare with the state-of-the-art 3D editing method (i.e., CLIP-NeRF [34]), we also report FID and FR (face recognition score) before and after editing. FR is the score of an image being recognized as a human face by a pre-trained model.

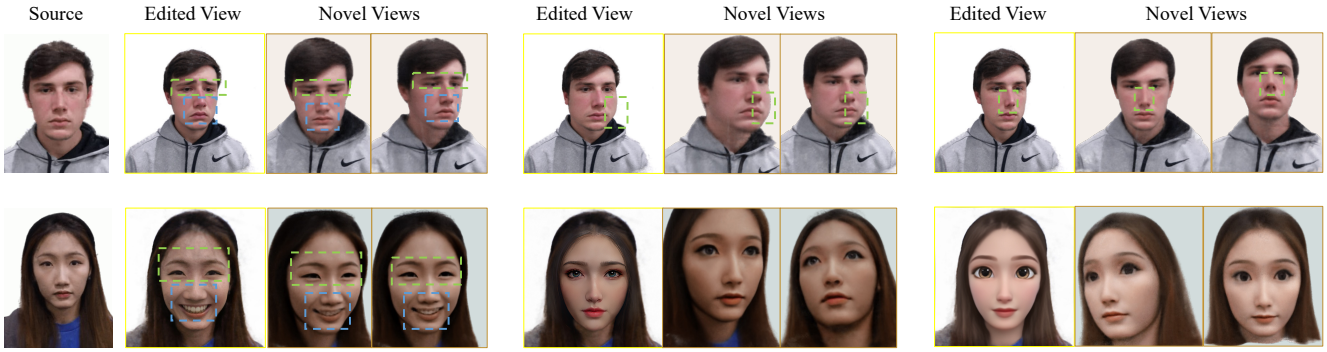


Figure 6. **Interactive editing qualitative results.** Our approach efficiently converts 2D edits into 3D while ensuring exceptional 3D consistency. The first row depicts how our method manages major geometric changes (marked in green and blue boxes) to a human face from a single view (altering facial expression, making face chubbier, shrinking nose), guaranteeing smooth and coherent geometry across various views. The second row highlights our capacity to handle significant stylistic and geometric modifications adeptly.

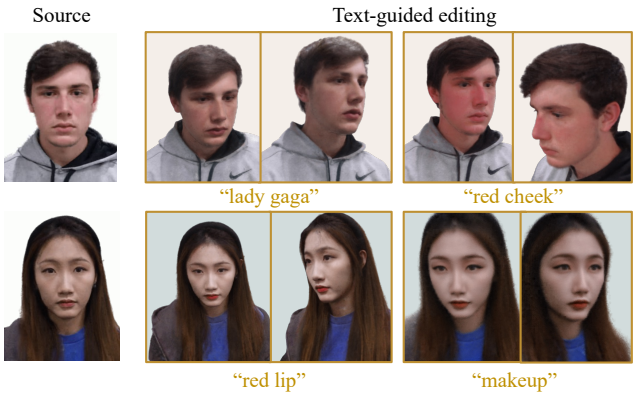


Figure 7. **Text editing qualitative results.** The results demonstrate that our method excels in text-guided editing, ensuring 3D consistency while preserving facial identity effectively.

Figure 8. **3D Style Transfer.** The style-guided image is on the top-left corner of the stylized scene.

Optimization Time. We measure the speed of our method on a single RTX3070 GPU. Training required 10k ($<$ 2 minutes) and 20k ($<$ 4 minutes) iterations for FaceSpace and TIFace, respectively. Finetuning needed a quarter of the initial iterations. We achieved real-time rendering of 1024×1024 images at 50 fps. Additionally, the training of the latent code adjustor and the finetuning of the StyleGAN took approximately 4 minutes across all datasets.

4.1. 3D Editing Results

Face Editing. As our inversion techniques work well and our method can produce an identity-preserved image, we can edit the frontal image with any existing image editing tools (even non-learning-based tools such as Photoshop) and get the projected latent code $\bar{w}_{style}$. Then, we can use our latent adjustor and hidden mapper to get stylized guided samples and perform 3D editing.

We show some interactive face editing results on the TIFace dataset in Fig. 6. In this case, any 2D image editing tool can be used. In Fig. 7, we show text-guided results, which use the latent code $\bar{w}_{style}$ generated by StyleCLIP [22]. Our method can successfully transfer diverse 2D editings to 3D even with a large domain gap, showing the potential for realistic applications.

Class-agnostic 3D Style Transfer. 3D style transfer generates stylized images from novel views using a style image. Utilizing a StyleGAN model trained on wikiArt [30], we facilitate class-agnostic 3D style transfer with a hidden mapper, as shown in Fig. 8, our method can successfully transfer the style from a given image without losing view consistency. Supplementary materials detail the implementation and demonstrate the stability and effectiveness of our method compared to the state-of-the-art method ARF [38] with almost no failure cases.

4.2. Comparing with Generative Baselines

We first compare with the 2D manipulation and 3D-aware baselines to demonstrate that our method can obtain high-fidelity and identity-preserving image generation, while also ensures good 3D consistency after editing.

For 2D manipulation baselines, we choose GANSpace

Table 2. **Comparison with generative baselines on FaceScape.**

	w/o Editing				w Editing	
	ID↑	PSNR↑	SSIM↑	LPIPS↓	APS↑	Pose ↓
GANSpace [8]	44.47	27.75	0.3993	0.3848	0.8142	9.38
InterFaceGAN [25]	62.16	29.61	0.7623	0.2028	0.8592	8.48
StyleNeRF [7]	15.39	27.44	0.3482	0.4845	0.3421	30.4
EG3D [2]	43.14	29.86	0.7738	0.2369	0.7694	7.95
Ours (w/o 3D)	80.45	32.22	0.8414	0.1316	0.8840	8.42
Ours	**94.59**	**33.71**	**0.8464**	**0.1277**	**0.9321**	**4.48**

[8] and InterFaceGAN [25], both of which are able to control the pose direction. To endow them with accurate angle adjustment ability, we augment these two methods with an extra MLP to predict the angle distance to be adjusted. For 3D-aware baselines, we adopt two state-of-the-art methods: StyleNeRF [7] and EG3D [2]. For our method, we report on two variants: the model without NeRF finetuning, i.e., "Ours (w/o 3D)" and the whole model, i.e., "Ours.".

Tab. 2 shows a quantitative comparison. **(1)** We investigate how well each method can control the camera pose. To realize this, we restrict the pose range to StyleGAN's training pose domain and align the images on FaceScape. After that, we apply each method to generate images on the same views as the testing set. This is done without editing the latent code, denoted as w/o Editing. We measure the ID score, PSNR, SSIM and LPIPS between the generated images and the groundtruth images. Tab. 2 demonstrates that our method outperforms all 2D and 3D-aware baselines, even without 3D-consistent module (Ours (w/o 3D)). **(2)** Then, we measure the disentanglement (APS) and 3D-consistency (Pose) capability.

According to Tab. 2, our method achieves the best APS score and lowest Pose error. The former shows that our method can better decompose edited attributes from non-edited. The latter indicates that our method can perform pose-dependent editing to ensure 3D-consistency. The qualitative comparisons are shown in Fig. 9. All methods, except StyleNeRF, can create good frontal images but fail in identity preservation compared to ours, as reflected by the lower ID scores in Tab. 2. However, pose variations lead to rapid degradation in baselines: GANSpace shows obvious background, InterFaceGAN has a large shift, and EG3D results are blurry, revealing poor disentanglement between pose and other styles. Notably, *Ours (w/o 3D)* has less accurate pose control than *Ours*, underlining the importance of the fine-tuning.

4.3. Compare with State-of-the-art 3D Editing

Here we compare against the state-of-the-art 3D editing methods to demonstrate that our approach supports high-fidelity editing on complex scenes.

For 3D editing, CLIP-NeRF [34] improves the previous EditNeRF [15] and is considered state-of-the-art. On complex scenes, CLIP-NeRF supports color editing with text prompts but fails to perform satisfying shape editing [34]. So we pre-define a set of color-editing texts w.r.t. the attributes of the FaceScape dataset, such as tanned skin, red lips, and blue eyes. In our method, we also define a set of shape-editing texts to verify the broader editing capability. For a fair comparison, we adapt CLIP-NeRF to use the same Hash encoding and the same configurations of the density net and color net as our method.

Figure 9. **Qualitative comparison with generative baselines.** The same human face is inverted by different methods to the latent code for pose manipulation.

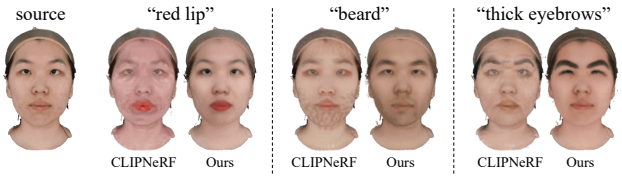


Figure 10. **Compared to CLIP-NeRF.** Our editing correctly edits the scene with given texts and maintains the visual quality.

The comparison results in Tab.3 reveal that after *color* editings, the FID score of CLIP-NeRF increases significantly, whereas ours shows a modest shift, indicating better image quality maintenance with our method. This advantage is more pronounced with the real-world TIFace dataset, as evidenced by a smaller FR difference (19.12 vs. 2.10). When extended to *shape* editing and *shape+color* editing, our method remains consistent performances, demonstrating the wider and more stable 3D editing capability. We also outperforms CLIP-NeRF in APS and ID scores, indicating superior attribute and identity preservation. Fig.10 visually contrasts our method with CLIP-NeRF. Our method performs accurate editings that fit the text descriptions and also maintains the image quality, 3D consistency and identity. For the CLIP-NeRF, although some editings can change the image to reflect the text prompts, obvious artifacts and floating noisy pixels appear around the face region.

4.4. User Study

We conducted a user study comparing human evaluations of our method with baselines and 3D editing methods using 60 scenes from the FaceScape test set and Ama-

Table 3. **Comparing with state-of-the-art 3D editing method.**

	FaceScape						TIFace			
	FID↓		FR↑		APS	ID	FR↑		APS	ID
	Before	After	Before	After			Before	After		
CLIP-NeRF [34] *(color)*	6.24	65.38 (+59.14)	85.72	69.23 (-16.49)	77.42	76.38	88.44	69.32 (-19.12)	80.96	75.79
Ours (color)	6.23	38.31 (+32.08)	85.43	81.08 (-4.35)	86.42	87.42	87.83	85.73(-2.10)	87.96	89.03
Ours *(shape)*	6.23	27.31 (+21.08)	85.43	81.97 (-3.46)	85.43	88.53	87.83	86.01(-1.82)	89.54	87.03
Ours *(shape+color)*	6.23	39.42 (+33.19)	85.43	81.34 (-4.09)	84.37	86.98	87.83	84.95(-2.88)	87.04	85.96

$*$ According to [34], CLIP-NeRF fails to get satisfying shape editing results on the complex scene (also see Fig. 10).

Table 4. **Statics from our user study.** Reported is percentage of workers voting for our method over the competing method.

	Realistic↑	Editing Accuracy↑
Ours vs. GANSpace	99.2	80.0
Ours vs. InterFaceGAN	97.6	75.2
Ours vs. StyleNeRF	99.6	100.0
Ours vs. EG3D	97.2	96.8
Ours vs. CLIP-NeRF	98.0	94.4

Table 5. **Ablation study.**

	removed module	FID↓	FR↑	APS	ID
Ours		39.42	81.34	84.37	86.98
latent code adjustor	*w/o* finetuning	43.94	79.40	80.46	79.03
	w/o differentiable	41.53	80.05	82.05	85.49
3D-consistent style editing	*w/o* L_{ori}	42.96	80.10	79.95	85.69
	w/o L_{br}	42.35	79.94	79.06	84.93
	w/o L_{ori}&L_{br}	45.96	76.46	77.03	81.94
	w/o L_{depth}	40.00	80.97	84.38	86.05

(a) Ours (b) w/o finetune (c) w/o L_{ori} (d) w/o L_{br} (e) w/o L_{org} & L_{br}

Figure 11. **Visual comparison of ablation study.**

(a) Ours (b) w/o Hidden Mapper

Figure 12. **Ablation study of hidden mapper.** (a) With hidden mapper, we can consistently transfer style to the entire scene. (b) W/o hidden mapper, the views outside the GAN training domain remain the original style and result in a scene with inconsistent styles across views.

zon Mechanical Turk. Participants assessed 360° and (-90°, 90°) rotated videos for match with text description and visual quality. Tab. 4 summarizes the results, indicating most users rated our method highest in quality. Detailed methodology, including video generation and evaluation criteria, is provided in the supplementary material.

4.5. Ablation Study

In this section, we provide ablation studies on FaceScape to verify the effectiveness of the proposed modules, training techniques, and loss terms.

Finetuning and differentiable decomposition play important roles in the latent code adjustor. As shown in Tab. 5, without finetuning applied, all metrics drop significantly, especially the ID score. Fig. 11(b) also shows how finetuning could affect identity preservation. Differentiable decomposition could help the latent code adjustor gain more disentangling results, which reduces the visual degradation from the 3D-inconsistent guidance. Without decomposing the latent vector, all metrics get worse.

In 3D-consistent style editing, all loss terms in Eqn. 5 contribute significantly to the result (Tab. 5). Without L_{br}, we will lose the constraint to the unedited region. As shown in Fig. 11(d), there are obvious floating artifacts around the face boundary. From Fig. 11(c), if we remove the L_{ori}, the constrain ability for L_{br} will be decreased, since the model tends to forget the original scene. Removing both L_{br} and L_{ori}, the finetuning will fail and the scene will become diverged, thus resulting in poor visual results (Fig. 11(e)).

We also show the importance of hidden mapper in real-world usage. In Fig. 12(b), as the color predicted by NeRF is conditioned on the viewing direction, without guided images on the GAN out-of-domain views, we will result in a scene with inconsistent styles from different views.

5. Conclusion

We present an efficient learning framework to bridge the gap between 2D editing and 3D editing, which is realized by two novel modules and a stable finetuning strategy. Our method can successfully transfer various editing patterns to 3D scenes, including text prompts, style-mixing, interactive warping, and style transfer. Our method outperforms all previous 3D editing methods with more flexible control and can support consistent editing by overcoming the domain shift problem in 2D GAN and 3D-aware GAN using self-supervised training techniques.

References

[1] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5795–5805, 2021. 1, 2, 4, 5

[2] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. 1, 2, 5, 7

[3] Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei-Sheng Lai, and Wei-Chen Chiu. Stylizing 3d scene via implicit representation and hypernetwork. *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 215–224, 2021. 3

[4] Tan M. Dinh, A. Tran, Rang Ho Man Nguyen, and Binh-Son Hua. Hyperinverter: Improving stylegan inversion via hypernetwork. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11379–11388, 2022. 4

[5] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Eigenvalue and generalized eigenvalue problems: Tutorial. *arXiv preprint arXiv:1903.11240*, 2019. 4

[6] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):3988–4004, 2021. 4

[7] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations*, 2021. 1, 2, 5, 7

[8] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems*, 33:9841–9850, 2020. 1, 2, 3, 4, 7

[9] Keke He, Yanwei Fu, Wuhao Zhang, Chengjie Wang, Yu-Gang Jiang, Feiyue Huang, and X. Xue. Harnessing synthesized abstraction images to improve facial attribute recognition. In *International Joint Conference on Artificial Intelligence*, 2018. 5

[10] Xueqi Hu, Qiusheng Huang, Zhengyi Shi, Siyuan Li, Changxin Gao, Li Sun, and Qingli Li. Style transformer for image inversion and editing. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11327–11336, 2022. 4

[11] Yihua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. Stylizednerf: Consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18321–18331, 2022. 3

[12] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 4

[13] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116, 2020. 1, 2, 5

[14] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. *Advances in neural information processing systems*, 28, 2015. 2

[15] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Junyan Zhu, and Bryan C. Russell. Editing conditional radiance fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5753–5763, 2021. 1, 2, 7

[16] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[17] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 41:1 – 15, 2022. 3

[18] Thu Nguyen-Phuoc, Feng Liu, and Lei Xiao. Snerf: Stylized neural implicit representations for 3d scenes. *ArXiv*, abs/2207.02363, 2022. 3

[19] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5480, 2022. 5

[20] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021. 1, 2, 4

[21] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *BMVC*, 2015. 5

[22] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2065–2074, 2021. 6

[23] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2

[24] Daniel Roich, Ron Mokady, Amit H. Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Transactions on Graphics (TOG)*, 2021. 4

[25] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9240–9249, 2020. 1, 2, 3, 4, 7

[26] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1532–1540, 2021. 2

[27] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6483–6492, 2019. 2

[28] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020. 2

[29] Jingxiang Sun, Xuan Wang, Yong Zhang, Xiaoyu Li, Qi Zhang, Yebin Liu, and Jue Wang. Fenerf: Face editing in neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7672–7682, 2022. 2, 4

[30] Wei Ren Tan, Chee Seng Chan, Hernan E Aguirre, and Kiyoshi Tanaka. Improved artgan for conditional synthesis of natural image and artwork. *IEEE Transactions on Image Processing*, 28(1):394–409, 2018. 6

[31] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021. 4

[32] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1283–1292, 2017. 2

[33] Andrey Voynov and Artem Babenko. Unsupervised discovery of interpretable directions in the gan latent space. In *International conference on machine learning*, pages 9786–9796. PMLR, 2020. 2

[34] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 1, 2, 5, 7, 8

[35] Xianglei Xing, Tian Han, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Unsupervised disentangling of appearance and geometry by deformable generator network. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10346–10355, 2019. 2

[36] Haotian Yang, Hao Zhu, Yanru Wang, Mingkai Huang, Qiu Shen, Ruigang Yang, and Xun Cao. Facescape: a large-scale high quality 3d face dataset and detailed riggable 3d face prediction. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 601–610, 2020. 5

[37] Yu-Jie Yuan, Yang tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: Geometry editing of neural radiance fields. *ArXiv*, abs/2205.04978, 2022. 2

[38] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. 6

[39] Kai Zhang, Nicholas I. Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, 2022. 3

[40] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 5

[41] Yiyu Zhuang, Hao Zhu, Xusen Sun, and Xun Cao. Mofanerf: Morphable facial neural radiance field. In *European Conference on Computer Vision*, 2022. 2, 5