NeRFEditor: Differentiable Style Decomposition for Full 3D Scene Editing

Chunyi Sun, Yanbin Liu, Junlin Han, Stephen Gould

Australian National University

Project page: https://chuny1.github.io/NeRFEditor/nerfeditor.html

Abstract

We present NeRFEditor, an efficient learning framework for 3D scene editing, which takes a video captured over 360° as input and outputs a high-quality, identity-preserving stylized 3D scene. Our method supports diverse types of editing such as guided by reference images, text prompts, and user interactions. We achieve this by encouraging a pre-trained StyleGAN model and a NeRF model to learn from each other mutually. Specifically, we use a NeRF model to generate numerous image-angle pairs to train an adjustor, which can adjust the StyleGAN latent code to generate high-fidelity stylized images for any given angle. To extrapolate editing to GAN out-of-domain views, we devise another module that is trained in a self-supervised learning manner. This module maps novel-view images to the hidden space of StyleGAN that allows StyleGAN to generate stylized images on novel views. These two modules together produce guided images in 360° views to finetune a NeRF to make stylization effects, where a stable finetuning strategy is proposed to achieve this. Experiments show that NeRFEditor outperforms prior work on benchmark and real-world scenes with better editability, fidelity, and identity preservation.

1. Introduction

Imaging if we can take a short video and generate an editable 3D scene. This ability will facilitate interesting applications in game and movie product, 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.

There are a few previous works that take steps towards enabling similar abilities, as summarized in Tab. 1. Using existing latent space manipulation techniques, 2D GANs [7,10,21] can produce stylized images from multiple camera poses. However, these 2D methods are confined to the training pose distribution and cannot ensure 3D consistency. 3D-aware synthesis methods [1,2,6,16] can generate
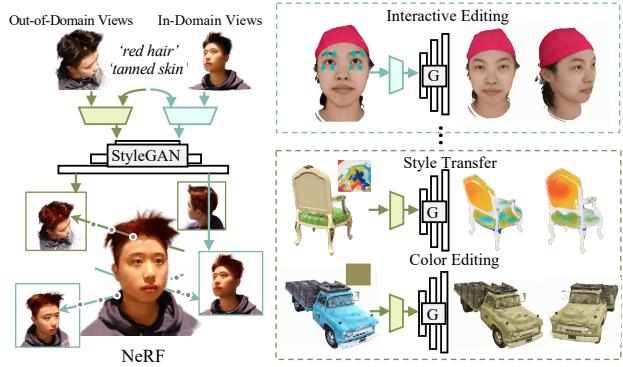

Figure 1. **Method overview.** (Left) We propose two modules for a pre-trained StyleGAN to generate stylized images from in-domain views and out-of-domain views. Then, we use these styled images to finetune a conditional NeRF for 3D-consistent 360° scene editing. (Right) Our method supports diverse types of editing.

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. Directly editing the neural radiance field (NeRF) is a promising direction for full scene editing. However, existing NeRF editing methods [12,31] 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.

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 (Fig. 1). Our framework leverages the novel-view synthesis merit of a NeRF and the well-behaved latent space of a pre-trained StyleGAN. The former ensures 3D-consistent scene editing, while the later 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 guided stylized images on any camera pose, we design two additional modules for StyleGAN. First, a *latent code adjustor* is devised to take the camera param-

Method	360° Editing	w/o 3D Auxiliary data	Real image Editing	3D Consistent	Real-time Editing
2D GANs [7, 10, 21]	✗	✓	✓	✗	✓
3D-aware [1, 2, 6, 16]	✗	✓	✗	✓	✗
NeRF editing [12, 31]	✓	✗	✗	✓	✗
Ours	✓	✓	✓	✓	✓

Table 1. **Compare the main features with previous works.**

eter as input and change the image to the target view by manipulating the latent code of StyleGAN. In the latent code adjustor, we introduce a differentiable decompositor to decomposite the pre-trained StyleGAN latent space $\mathcal{W}$ into orthogonal basis. Then, the adjustor disentangles the pose from other styles to generate multiview stylized images to guide NeRF stylization. Second, to overcome the limitation that StyleGAN cannot produce images on the out-of-domain poses, we design a novel *hidden mapper*. This maps novel-view images to the hidden feature space of StyleGAN, where style mixing is applied to generate stylized images to guide the NeRF stylization training. We develop a self-supervised training algorithm for hidden mapper so that StyleGAN can adapt generative ability to out-of-domain views.

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 supports a wide range of 3D editing applications, 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 full 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, which brings NeRF and pre-trained StyleGAN into a unified learning framework.

- We devise a self-supervised hidden mapper to facilitate the out-of-domain style editing.

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

2. Related Work

2D Latent Space Manipulation. Previous 2D GAN manipulation works [7, 30, 32] show that the latent space of pre-trained GANs can be decomposed to control the image

---

[1]Code and dataset will be released.

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 cannot cover a diverse and continuous range of viewing directions, both the supervised [11, 21, 23, 29] and unsupervised [7, 22, 30, 32] 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 feature outside the training viewing distribution with a novel self-supervised hidden mapper.

3D-aware Synthesis. Most recent 3D-aware synthesis methods rely on the NeRF [13] technique to generate 3D-consistent images using unstructured 2D training images. As a pioneer work, pi-GAN [1] represents the implicit neural radiance field by a SIREN network [24] and applies the FiLM [18] conditioning on SIREN, leading to the view-consistent synthesis. FENeRF [26], EG3D [2] and StyleNeRF [6] 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. Despite the view-consistent synthesis, the 3D-aware 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, 2D GAN inversion has been well-explored, and our proposed differentiable adjustor further improves the GAN inversion.

Editable NeRF. NeRF [13] encodes the radiance field of a scene in the MLP weights to conduct novel-view rendering. As an implicit function, NeRF is challenging to edit. Existing works [12, 31, 34, 37] only support editing on local parts of object or with simple types. EditNeRF [12] was the first work to edit the shape and color of NeRF on local parts of simple objects. CLIP-NeRF [31] 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 the complex scenes and objects (e.g., human face) to generate high-fidelity 3D-consistent images. Meantime, we support more diverse editing types such as reference image, text, and user interactions.

3. Mutual NeRF-StyleGAN Training

As shown in Fig. 2, we design an efficient learning framework for 360° scene editing, which leverages the 3D consistency strength of NeRF and latent space manipulation ability of StyleGAN. To facilitate mutual training, we adapt
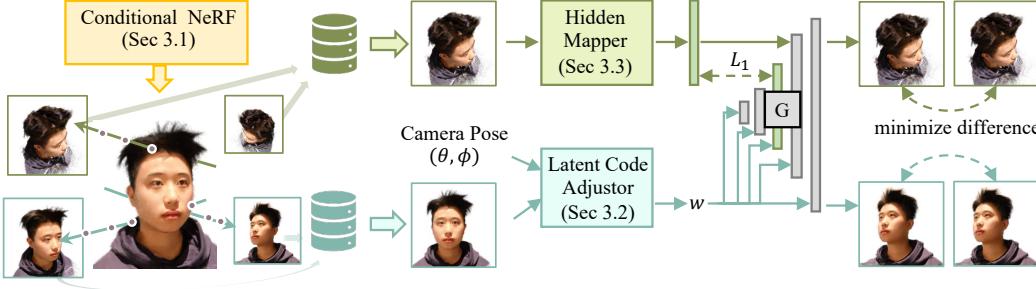
Figure 2. **The framework of NeRFEditor.** (1) The *conditional NeRF* is pre-trained on the original scene to produce sufficient (image, pose) pairs. (2) The pairs are used to train a *Latent Code Adjustor*, which disentangles camera pose from other styles in the learned latent space of StyleGAN. (3) 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).

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 adapt a conditional NeRF model to (1) produce (image, pose) pairs for style manipulation in Style-GAN 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 and stylized scene, and assign unique appearance codes for each different image in the stylized guided set. 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 by trainable MLPs $\mathcal{E}_{\text{style}}(\cdot)$ and $\mathcal{E}_{\text{app}}(\cdot)$; $\mathbf{x}$ is encoded by a hash encoder $\mathcal{H}(\cdot)$ [14]; and $\mathbf{v}$ is encoded by position encoding $\mathcal{R}(\cdot)$. The colors $\mathbf{c}$ and density σ are predicted as follows:

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

where $\mathbf{F}_{\text{dens}}$ and $\mathbf{F}_{\text{color}}$ denote the density and color subfunctions of $\mathbf{F}$.

3.2. Latent Code Adjustor

After obtaining sufficient (image, pose) pairs, we employ a pre-trained StyleGAN to generate high-fidelity stylized images from multiple camera views. Recent works [7, 21] found that the pre-trained StyleGAN has a well-behaved latent space, which involves interpretable styles such as pose,
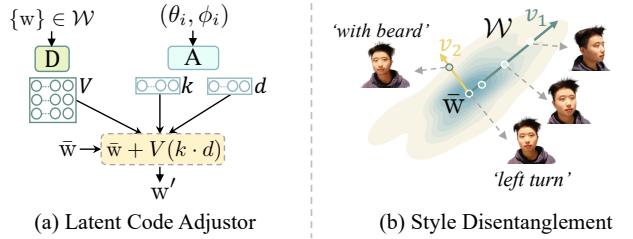

(a) Latent Code Adjustor   (b) Style Disentanglement

Figure 3. **Latent Code Adjustor.** (a) The latent space $\mathcal{W}$ is decomposed by a differentiable node $\mathbf{D}$ 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 different styles: *'left turn'* and *'with beard'*. Hence, $\mathcal{W}$ is decomposed into two directions v_1 and v_2. Traversing over v_1 will only enforce pose variations.

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 the high-fidelity, we restrict the camera pose range to lie in StyleGAN's training pose distribution [1, 16, 26]. This range is defined as the in-domain camera views $\mathbf{V}_{\text{in}}$.

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 [3, 8, 28] 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 $\mathcal{W}$

3

to get an orthogonal basis V representing different styles. Specifically, we first sample a large batch of z from the normal distribution and use the mapping network $\mathbf{M}$ to get a large set of $w \subseteq \mathcal{W}$. Then, we compute the covariance matrix $\text{COV}(\{w\})$ and solve an eigen-decomposition problem [4] on $\text{COV}(\{w\})$ to obtain the orthogonal basis V.

The eigen-decomposition is differentiable and trained end-to-end with other modules. We explicitly derive gradients using DDNs technique [5] to efficiently back-propagate through eigen-decomposition.[2]

Latent code adjustment. Given the orthogonal basis V and a latent code w, image editing can be done as $w' = w + Vx$, where entry x_k of x is a separate control parameter.

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 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: $w' = w + V(k \cdot d)$, $\widehat{I}^{(\theta,\phi)} = \mathbf{G}(w')$.

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 as following:

$$L_{total} = L_2 + L_{vgg} + L_{id} + L_{reg}. \qquad (1)$$

Here, the first three terms represent the ℓ_2 distance, VGG perceptual distance [9], 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 [7, 21] on the frontal latent code $\bar{w}$ to get w_{style}. We use the latent code adjustor to achieve generating multi-view stylized images. To generate the in-domain stylzied set, we generate the 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 domain, StyleGAN struggle to extrapolate to extreme camera views outside the pose distribution, denoted as $\mathbf{V}_{out}$. We design a *hidden mapper* $\mathbf{H}$ to tackle a limitation of StyleGAN (extrapolation inability of unseen poses), thus enabling 360° style editing.

Self-supervised Training. Latent codes of shallow layers generally control the geometric aspects (e.g. pose and

---
[2]Problem formulation and derivation are in the Appendix.
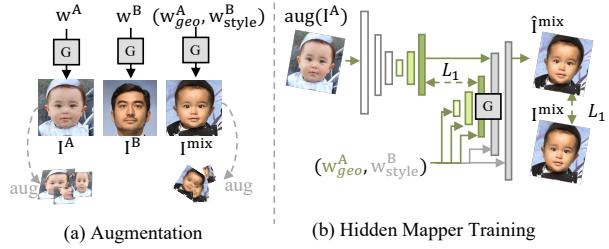

(a) Augmentation   (b) Hidden Mapper Training

Figure 4. **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_{geo}^A, w_{style}^B). (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
$F^A = \text{aug}(\mathbf{G}_{geo}(w_{geo}^A))$ # augment the hidden map
$\widehat{F}^A = \mathbf{H}(\text{aug}(I^A))$ # predict the hidden map
$I^{mix} = \mathbf{G}_{style}(F^A, w_{style}^B)$ # generate mixed image
$\widehat{I}^{mix} = \mathbf{G}_{style}(\widehat{F}^A, w_{style}^B)$ # predict mixed image
Return: loss $= L_1(I^{mix}, \widehat{I}^{mix}) + L_1(F^A, \widehat{F}^A)$

---

shape), while latent codes of deep layers control the style attributes (e.g. skin color). Style-mixing [10] 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 also perform style-mixing by mixing the hidden maps with w_{style}. Most importantly, we observe that, when we perform augmentation operations (crop, rotate, rescale) to the hidden map, the style can still transfer to the correct location. If we can learn a mapper 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 $F = \mathbf{G}_{geo}(w_{geo}), I = \mathbf{G}_{style}(F, w_{style})$. 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. 4.

Out-of-domain stylized set $\mathcal{I}_{out}$. We first get the unedited images from out-of-domain views $\mathbf{V}_{out}$ with conditional NeRF. Then hidden mapper converts these images to the hidden map F, which are combined with target editing styles w_{style} to obtain the stylized images.

3.4. 3D-Consistent Style Editing

Once we have trained the latent code adjustor and the hidden mapper, the model is capable of producing stylized images of $360°$. We use the images $\mathcal{I}_{ori}$ generated by 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 style editing.

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

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

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

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

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

To further handle the inconsistency in the guided training set, we follow [15] to render the depth map $\mathbf{d}$ and adopt a depth regularisation term to smooth the stylized scene:

$$L_{depth} = \sum_{i,j} (\widehat{\mathbf{d}}_{i,j} - \widehat{\mathbf{d}}_{i,j+1})^2 + (\widehat{\mathbf{d}}_{i,j} - \widehat{\mathbf{d}}_{i+1,j})^2. \quad (5)$$

The final loss used for our 3D-consistent NeRF editing is:

$$L = (L_{style} + L_{br} + L_{ori}) + \lambda \cdot L_{depth}. \quad (6)$$

4. Experiments

Datasets. We evaluate *NeRFEditor* on two datasets: FaceScape [33, 37] as a high-quality 3D face benchmark and TIFace (Tiny-scale Indoor Face collection) as a newly-collected real-world dataset. FaceScape contains 359 different subjects each with 20 expressions, and 120 multiview images for each expression. It is captured in a lab environment and the subject wears a turban. To evaluate the editing capability for real 3D scenes, we collect TIFace, a real-world dataset including 10 different persons from a realistic environment, without any restriction on dressing. Details of the dataset collection can be found in the Appendix.

Evaluation Metrics. Following previous works [1, 2, 6], we use various metrics to evaluate the image quality: PSNR, SSIM, and LPIPS [36]. To quantify the 3D-consistency, we employ Pose (pose adjusting error) [2]. To evaluate the identity-preservation, we use ID Score [17], which measures the confidence of classifying one image to have the same identity as the groundtruth image. Meanwhile, we use APS (attribute-preservation score) to measure the preservation of non-edited attributes after editing.

To compare with the state-of-the-art 3D editing method (i.e. CLIP-NeRF [31]), we also report FID and FR (face recognition score) before and after editing. FR is the score of an image being recognized as human face.

Optimization Time Measure. We measure the training speed of our NeRF model in a single RTX3070 GPU, and the training time of each step is $\sim$8ms. During training in the original scene, we set the iterations to 10k for FaceSpace and 20k for TIFace. During finetuning, we set the iterations to 1/4 of the training iterations of the original scene. Thus, the total NeRF training time is less than 4 minutes for the complex scene, and 2 minutes for the simple scene. To render a 1024×1024 image, the speed is 50 fps, which can be rendered in real time. The time to train the latent code adjustor and finetune the StyleGAN is around 4 minutes consistently for all datasets in 1024×1024 resolution.

4.1. Compare 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 [7] and InterFaceGAN [21], 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 [6] and EG3D [2]. For our method, we report on two variants: the model without 3D-consistent style editing, i.e. Ours (w/o 3D) and the whole model, i.e. Ours. For a fair comparison, we apply PTI [19] to all methods to achieve better inversion.

The quantitative comparison results are shown in Tab. 2. **(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. For GANSpace and InterFaceGAN, we use their own stylization

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

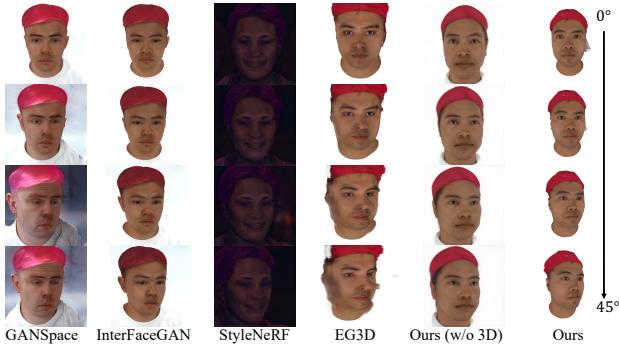|	w/o Editing				w Editing	
	ID↑	PSNR↑	SSIM↑	LPIPS↓	APS↑	Pose↓
GANSpace [7]	44.47	27.75	0.3993	0.3848	0.8142	9.38
InterFaceGAN [21]	62.16	29.61	0.7623	0.2028	0.8592	8.48
StyleNeRF [6]	15.39	27.44	0.3482	0.4845	0.3421	30.4
EG3D [2]	43.14	29.86	0.7738	0.2369	0.7694	7.95
Ours (w/o 3D)	80.45	32.22	0.8414	0.1316	0.8840	8.42
Ours	**94.59**	**33.71**	**0.8464**	**0.1277**	**0.9321**	**4.48**

Figure 5. **Qualitative comparison with generative baselines.** The same human face is inverted by different methods to the latent code for pose manipulation.

method for editing. For StyleNeRF and EG3D, we apply 2D editing method [7] on the frontal image and get an inverted latent code. According to Tab. 2, our method achieves the best APS score and least 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. 5. It illustrates that all methods except for StyleNeRF can produce relatively good-quality frontal image, with the corresponding inversion strategies. However, they exhibit worse identity-preserving effect compared with our method, which is consistent with their lower ID scores in Tab. 2. Furthermore, when we vary the pose, the baselines degrade quickly: GANSpace incurs obvious background; InterFaceGAN has a large shift; EG3D obtains blurry results. This reveals bad disentanglement between pose and other styles. *Ours (w/o 3D)* shows less accurate pose control than *Ours*, verifying the necessity of the latent code adjustor.

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

We then compare with the state-of-the-art 3D editing method to demonstrate that our method supports high-fidelity editing on complex scenes. For 3D editing, CLIP-NeRF [31] improves the previous EditNeRF [12] and becomes the state-of-the-art method. On complex scenes, CLIP-NeRF supports color editing with text prompts but fails to perform satisfying shape editing [31]. So we pre-
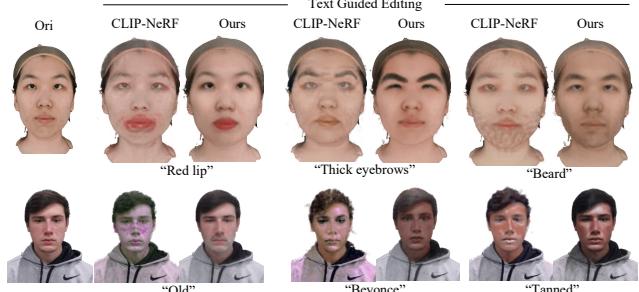

Figure 6. **Compared to CLIP-NeRF.** Our editing correctly edits the scene with given texts and maintains the quality of the original scenes. We invite readers to check the project page for better comparison.
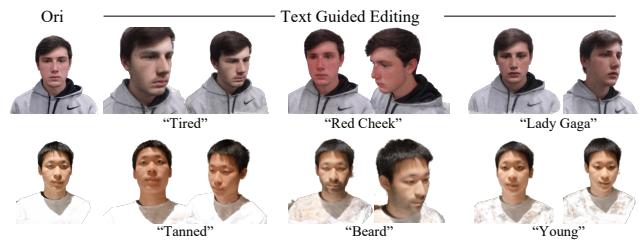

Figure 7. **Results of our method on TIFace dataset.** Our method can make accurate editing with given guided text descriptions. Please refer to our project page to see high-quality editing result with a moving camera.

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.

The comparison results are shown in Tab. 3. After applying color editings to FaceScape, the FID score of CLIP-NeRF increases significantly, while our method shows a reasonable shift. Similarly, after editing, the FR of CLIP-NeRF drops by 16.49, but our method only reduces by 4.35. For the real-world TIFace dataset, the FR differences are more severe (19.12 versus 2.10). The change of two metrics indicates that color editing by CLIP-NeRF significantly harms the image quality, while our method can better maintain the image quality. 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 measure the APS and ID score to evaluate the non-edited attribute-preserving and identity-preserving features of the 3D editing methods. Our method outperforms CLIP-NeRF on both metrics.

When applying our method to the real-world dataset, we model each scene as an unbounded scene. We fol-

Table 3. **Compare 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 [31] (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 [31], CLIP-NeRF fails to get satisfying shape editing results on the complex scene (also see Fig. 6).

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

low [35] to use an additional background network to encode the background, which can successfully surpass the floating artefacts and improve the image quality. Fig. 6 shows the qualitative comparison between our method and 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. In Fig. 7, we demonstrate more editing results guided by various text prompts on the real-world TIFace dataset. Our method achieves high-quality editing results and scales well for real-world scenes.

4.3. User Study

We also conduct a user study to compare the human evaluations between our method and all baselines and 3D editing methods. We randomly sample 60 scenes from the FaceScape test set and sample 5 different text descriptions for each scene. To compare with the CLIP-NeRF, we generate 360° rotated videos before and after editing for both methods. Other methods can not extrapolate well to the extreme camera views. Thus, we generate rotated videos only (-90°, 90°) around the frontal face before and after editing for each method. Afterward, we use Amazon Mechanical Turk to perform a user study. We require workers to choose the video that (1) better matches the text description and (2) has a higher visual quality (i.e. fewer artifacts, clear boundaries, and more natural). For each pair, we shuffle the positions randomly and ask five workers to make comparison. Tab. 4 summarizes the comparison, where most users consistently rate our method as the high-quality one.

4.4. 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. 8(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 differentiable approach, all metrics get worse.

In 3D-consistent style editing, all loss terms in Eqn. 6 contribute significantly to the result (Tab. 5). Without L_{br}, we will lose the constraint to the unedited region. As shown in Fig. 8(d), there are obvious floating artifacts around the face boundary. From Fig. 8(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. 8(e)).

We also show the importance of hidden mapper in real-world usage. In Fig. 9(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.

4.5. Applications

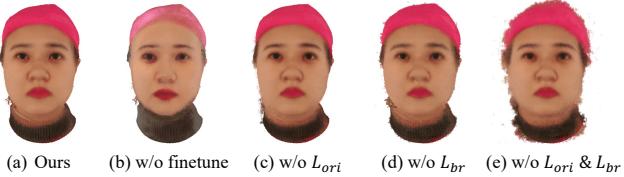**Style Mixing.** We can create the style-mixed image using StyleGAN. We apply style mixing [10] to the frontal

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

Figure 8. **Visual comparison of ablation study.**

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

Figure 9. **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.

Figure 10. **Style-Mixing Result.** Our method can adapts Style-Mixing to 3D. Refer to the project page for $360°$ results.

Figure 11. **Interactive Editing Result.** $360°$ results are in the project page.

Figure 12. **3D Style Transfer.** The style-guided image is on the top-left corner of the stylized scene.

image latent code $\bar{w}$ and the w_{style}. Fig. 10 provides the result on various views, showing that our method can successfully transfer the mixed style from the source image, even with large appearance and environmental changes.

Interactive 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 $\bar{w}_{style}$. We can use our latent adjustor and hidden mapper to get stylized guided samples and perform 3D editing. In Fig. 11, the left example is edited by makeup transfer tool and the right one is edited by image warping tool. Our method can successfully transfer both 2D editings to 3D. This indicates that our method is not limited by StyleGAN-based editing, but instead it can leverage any 2D image editing tool to edit one view of the 3D scene and generalize the effect to all views to produce an edited 3D scene.

Class-agnostic 3D Style Transfer. 3D style transfer aims to generate photorealistic images from arbitrary novel views given a style image. The major difficulty is how to achieve cross-view consistent style transfer. The StyleGAN model trained on wikiArt [27] can generate class-agnostic paintings. With this pre-trained model, we can perform class-agnostic 3D style transfer using the hidden mapper. For each image on the NeRF training set, we use the hidden mapper to map them into a common pre-trained Style-GAN hidden space and generate the latent code w_{style} for the guided style image to perform style transfer.

As depicted on Fig 12, our method can successfully transfer the style from a given image without loss of view-consistency and works for any class. Our simple approach can achieve comparable results with the current state-of-the-art 3D style transfer methods. By mapping the image into different StyleGAN blocks and changing the style-mixing factor, we can also enable the control of the abstract level and strength of stylization. The style transfer is also temporal consistent, combined with deformable NeRF will allow our method generalize to dynamic scenes. Details and more results can be found in the Appendix.

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 $360°$ 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, 3, 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, 6

[3] 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. 3

[4] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Eigenvalue and generalized eigenvalue problems: Tutorial. *arXiv preprint arXiv:1903.11240*, 2019. 4, 11

[5] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):3988–4004, 2021. 4, 11

[6] 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, 6

[7] 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, 5, 6

[8] 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. 3

[9] 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

[10] 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, 4, 7

[11] 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

[12] 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, 6, 11

[13] 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

[14] 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, 12

[15] 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

[16] 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, 3

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

[18] 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

[19] 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)*, 2022. 5

[20] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016. 12

[21] 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, 5, 6

[22] 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

[23] 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

[24] 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

[25] Jiayu Sun, Zhanghan Ke, Lihe Zhang, Huchuan Lu, and Rynson W. H. Lau. Modnet-v: Improving portrait video matting via background restoration. *ArXiv*, abs/2109.11818, 2021. 12

[26] 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 Con-*

ference on Computer Vision and Pattern Recognition, pages 7672–7682, 2022. 2, 3

[27] 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. 8

[28] 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. 3

[29] 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

[30] 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

[31] 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, 6, 7, 11

[32] 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

[33] 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

[34] 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

[35] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *ArXiv*, abs/2010.07492, 2020. 7

[36] 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

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

A. Conditional NeRF Architecture

Fig. 13 provides a detailed network architecture of the conditional NeRF. Compared to previous approaches [12, 31] that train the network on massive 3D data, our network can be much lighter. This also helps us achieve faster speed in the style converting step.
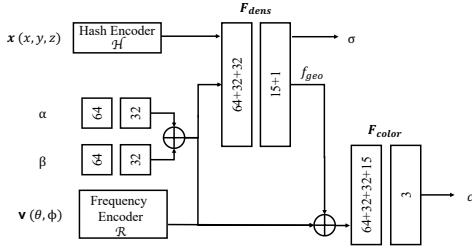

Figure 13. **Conditional NeRF Architecture.**

B. Differentiable Style Decomposition

In the main paper (Sec. 3.2), we propose the differentiable style decomposition to decomposite the latent space $\mathcal{W}$ into an orthogonal basis V. Here we summarize the problem formulation and explicitly derive the gradients for back-propagation.

We first sample a large batch of $z \in \mathcal{N}(0, \mathbf{I})$ (normal distribution), then use the mapping network to get a large set of styles $w \in \mathcal{W}$. Let $\mathbf{W} = \{w\} \in \mathbb{R}^{N \times D}$, we can calculate the covariance matrix as:

$$\mathbf{\Sigma} = \mathrm{COV}(\mathbf{W}) = \frac{1}{N-1}(\mathbf{W} - \bar{\mathbf{W}})^\top (\mathbf{W} - \bar{\mathbf{W}}), \quad (7)$$

where $\bar{\mathbf{W}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{W}_i$.

Eigen-decomposition. To explore disentangled styles in the latent space $\mathcal{W}$, we adopt the eigen-decomposition to find an orthogonal basis $V \in \mathbb{R}^{D \times k}$, consisting of eigenvectors w.r.t the top-k eigenvalues. The formulation is

$$V \in \min_{U \in \mathbb{R}^{D \times k}} -\mathbf{tr}(U^\top \mathbf{\Sigma} U)$$
$$\text{subject to } U^\top U = I. \quad (8)$$

To solve Eq. (8), we use a widely-used power iteration (PI) method [4], which is a numerical method to iteratively update until converge to eigenvectors.

Backward Gradient Derivation. To integrate Eq. (8) in deep neural networks for end-to-end training, a straightforward way is to perform auto-differentiation and backpropagate through the power iterations. However, this method is in-efficient in both memory and computation. Thereby,

we adopt the DDNs technique [5] to explicitly derive the gradients for efficient backward propagation. Formally, we define $f(\mathbf{\Sigma}, V) = -\mathbf{tr}(V^\top \mathbf{\Sigma} V)$ and $h(V) = V^\top V - I$. According to [5], the gradient of V w.r.t $\mathbf{\Sigma}$ can be calculated as:

$$DV(\mathbf{\Sigma}) = H^{-1} A^\top (A^\top H^{-1} A)^{-1} A H^{-1} B - H^{-1} B, \quad (9)$$

where

$$A = D_V h(V), \quad (10)$$
$$B = D^2_{\mathbf{\Sigma} V} f(\mathbf{\Sigma}, V), \quad (11)$$
$$H = D^2_{VV} f(\mathbf{\Sigma}, V) - \lambda D^2_{VV} h(V), \quad (12)$$

and λ satisfies $\lambda^\top A = D_V f(\mathbf{\Sigma}, V)$.

At neural network training time, the gradients flow from loss function to V, then flow through $\mathbf{\Sigma}$ according Eq. 9.

C. Hidden Mapper Architecture

Fig. 14 provides the hidden mapper architecture, which contains a pertained VGG16 as the backbone and a few convolutional layers to produce different sizes of StyleGAN hidden features maps ($F_{32 \times 32}$, $F_{64 \times 64}$, and $F_{128 \times 128}$). During self-supervised training, we randomly sample one of the three hidden maps to perform style-mixing. The training takes 20,000 iterations with a batch size of 2, which only performs once and can be used for all scenes.
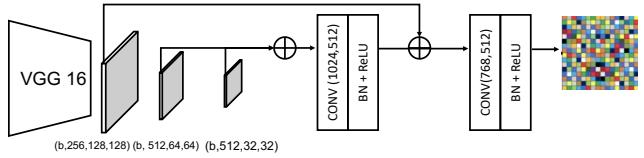

Figure 14. **Architecture of Hidden Mapper.**

D. 3D Style Transfer

Fig. 15 further shows how our method can control the abstraction level of style transfer. Mapping the image to a shallow style block of StyleGAN will allow the transferred image more similar to the abstraction level of the style image. As we map the image to the hidden space, the shallow level will produce more abstract transferred results. If we map the image to the deeper layer, as shown in Fig. 15 bottom row, the output images almost only share color information of the given style image. To control the strength of the style transfer, we can choose one image from the NeRF dataset, and then calculate its corresponding latent code. When performing style transfer, we can mix it with the w_{style} with different mixing factors to control the style transfer strength.

Fig. 16 further shows that our method can produce temporally consistent style transfer. The top example shows in

t = 300, the style is still consistent even the camera changes its position. The bottom examples show that for a fast-moving target, the style can also transfer consistently. This support that our method can even be applied to video style transfer and extend to deformable NeRF for 4D scene style transfer.
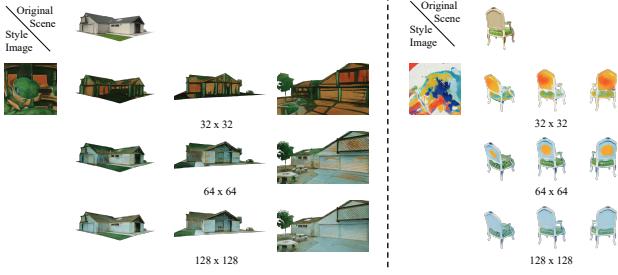

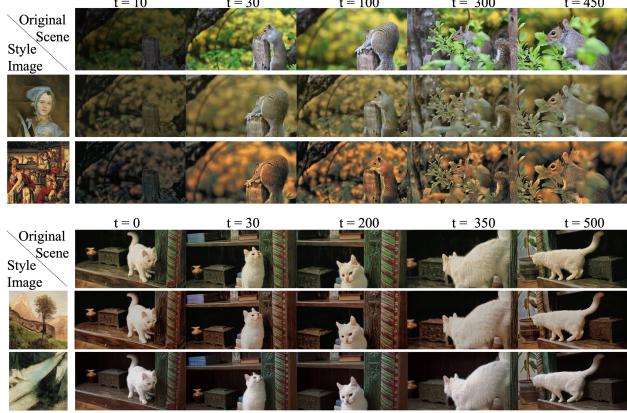Figure 15. **Style Transfer Result on Different Level.**

Figure 16. **Style Transfer Result on Video.**

E. Dataset Collection

We capture the 360 ° video using a single camera around each participant and require the participant to sit and stay for one minute. To make the background static but still under natural lightering conditions, we choose empty rooms for data collection. For each participant, we capture three videos. For each video, we first filter blurry frames by calculating the sharpness of each frame, and then we keep 100 frames for each video. After filtering, we use COLMAP [20] to compute the pose for each image and the camera intrinsics. Segmenting the background can enable faster training and reduce floating noise. We use the previous state-of-the-art video matting model [25] to segment the foreground and save the foreground mask for each frame. After that, we use the instantNeRF [14] to reconstruct each scene. For each participant, we kept the data sample with the best visual quality.