SketchFaceNeRF: Sketch-based Facial Generation and Editing in Neural Radiance Fields

LIN GAO*, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences , China
FENG-LIN LIU, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences, China
SHU-YU CHEN, Institute of Computing Technology, Chinese Academy of Sciences, China
KAIWEN JIANG, Institute of Computing Technology, CAS and Beijing Jiaotong University, China
CHUNPENG LI, Institute of Computing Technology, Chinese Academy of Sciences, China
YU-KUN LAI, School of Computer Science and Informatics, Cardiff University, UK
HONGBO FU, School of Creative Media, City University of Hong Kong, China

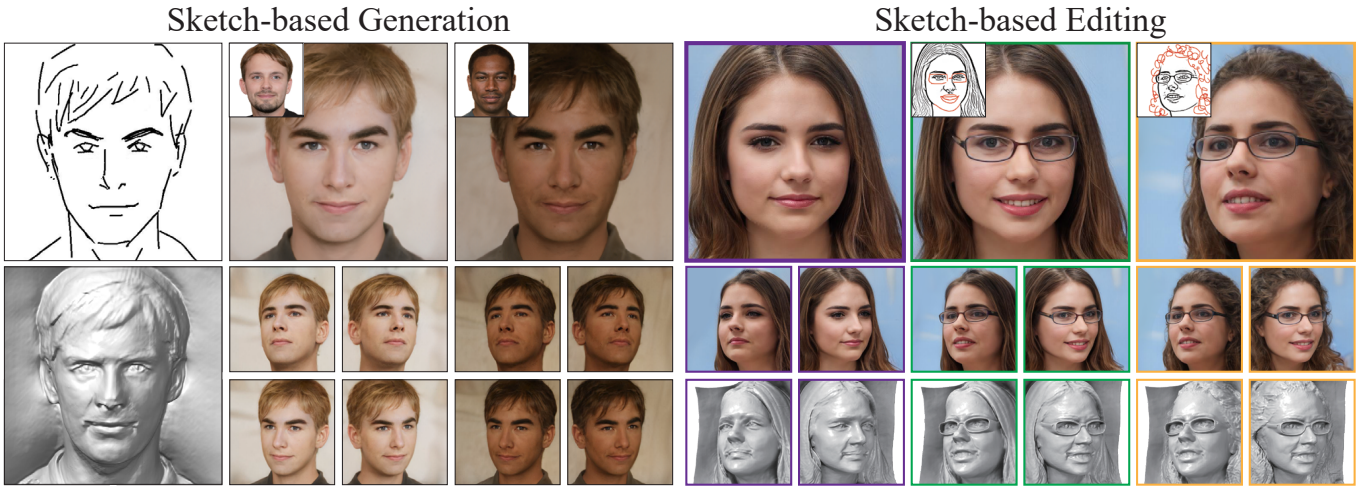


Fig. 1. Our SketchFaceNeRF system supports both generation and editing of high-quality facial NeRFs from 2D sketches. As shown in the left half part, given a hand-drawn sketch (top-left corner), photo-realistic rendering results with different appearance are synthesized from scratch. The detailed geometry model and free-view rendering results are shown at the bottom. On the right half part, we show sketch-based editing of facial NeRFs and the corresponding geometry, where original faces and geometry are shown in purple boxes, and the results of two consecutive editing steps are shown in green and orange boxes, respectively. During editing, local regions are modified according to the edited sketches highlighted in red, while the geometry and appearance features in unedited regions are well preserved.

Realistic 3D facial generation based on Neural Radiance Fields (NeRFs) from 2D sketches benefits various applications. Despite the high realism of free-view rendering results of NeRFs, it is tedious and difficult for artists to achieve detailed 3D control and manipulation. Meanwhile, due to its conciseness and expressiveness, sketching has been widely used for 2D facial image generation and editing. Applying sketching to NeRFs is challenging due to the inherent uncertainty for 3D generation with 2D constraints, a significant gap in content richness when generating faces from sparse sketches, and potential inconsistencies for sequential multi-view editing given only 2D sketch inputs. To address these challenges, we present SketchFaceNeRF, a novel sketch-based 3D facial NeRF generation and editing method, to produce free-view photo-realistic images. To solve the challenge of sketch sparsity, we introduce a Sketch Tri-plane Prediction net to first inject the appearance into sketches, thus generating features given reference images to allow color and texture control. Such features are then lifted into compact 3D tri-planes to supplement the absent 3D information, which is important for improving robustness and faithfulness. However, during editing, consistency for unseen or unedited 3D regions is difficult to maintain due to limited spatial hints in sketches. We thus adopt a Mask Fusion module to transform free-view 2D masks (inferred from sketch editing operations) into the tri-plane space as 3D masks, which guide the fusion of the original and sketch-based generated faces to synthesize edited faces. We further design an optimization approach with a novel space loss to improve identity retention and editing faithfulness. Our pipeline enables users to flexibly manipulate faces from different viewpoints in 3D space, easily designing desirable facial

*Corresponding author is Lin Gao (gaolin@ict.ac.cn).

Authors' addresses: Lin Gao, Feng-Lin Liu, Shu-Yu Chen, Kaiwen Jiang and Chunpeng Li are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences. Yu-Kun Lai is with the School of Computer Science and Informatics, Cardiff University. Hongbo Fu is with the School of Creative Media, City University of Hong Kong. Authors' e-mails: gaolin@ict.ac.cn, liufenglin21s@ict.ac.cn, chenshuyu@ict.ac.cn, kevinjiangedu@gmail.com, cpli@ict.ac.cn, LaiY4@cardiff.ac.uk, hongbofu@cityu.edu.hk.

0730-0301/2023/1-ART1
https://doi.org/10.1145/3592100

models. Extensive experiments validate that our approach is superior to the state-of-the-art 2D sketch-based image generation and editing approaches in realism and faithfulness.

CCS Concepts: • **Human-centered computing** → **Graphical user interfaces**; • **Computer systems organization** → Neural networks; • **Computing methodologies** → *Rendering*; Volumetric models.

Additional Key Words and Phrases: Sketch-based Interaction, Neural Radiance Fields, Face Modeling, Face Editing

ACM Reference Format:
Lin Gao, Feng-Lin Liu, Shu-Yu Chen, Kaiwen Jiang, Chunpeng Li, Yu-Kun Lai, and Hongbo Fu. 2023. SketchFaceNeRF: Sketch-based Facial Generation and Editing in Neural Radiance Fields. *ACM Trans. Graph.* 1, 1, Article 1 (January 2023), **??** pages. https://doi.org/10.1145/3592100

1 INTRODUCTION

Highly realistic and stereoscopic face modeling is a popular topic in computer graphics and has a wide range of applications, including digital character design, avatar-based virtual meetings, *etc.* Nevertheless, creating high-quality facial models in terms of both geometry and appearance from scratch requires laborious authoring with professional software (e.g., MAYA [**?**], ZBrush [**?**] and NVIDIA Omniverse [**?**]). More importantly, it is very difficult for existing mesh-based approaches to render photo-realistic facial images without professional skills or high costs, and realism is probably the most critical aspect of practical applications. Thus, how to conduct facial modeling in an easy-to-use yet realistic way is a worth-studying research problem. Thanks to the development of deep learning approaches, Neural Radiance Fields (NeRFs) [**?**], a powerful implicit 3D representation, can easily reconstruct face models from multi-view images and render photo-realistic free-view results. However, directly using a vanilla NeRF to perform face manipulation is very challenging since information describing only a specific object or scene is encoded by a NeRF network. Although various 3D GAN (Generative Adversarial Network) approaches [**????**] have been proposed to generate facial NeRFs by random sampling instead of reconstructing real scenes, such methods still lack detailed control and interpretable manipulations over synthesized faces.

Several methods [**????**] have attempted to address these issues by using sliders to interactively edit predefined attributes based on 3DMM (3D Morphable Models) such as FLAME [**?**], but they provide limited manipulation freedoms. On the other hand, it is natural for humans to describe images with a long-lasting tool, namely pens, which can also be utilized for efficient and realistic manipulation in the context of 3D facial GAN. Previous methods [**???**] resort to semantic masks as an editing interface, which, however, does not offer fine control of facial details like hairstyles, beards, etc. Another promising pen-based editing interface is sketching. 2D sketching has been widely used to condition facial image generation [**???**] and editing [**???**]. Sketch-based interfaces have also been explored for 3D face modeling [**???**]. However, these solutions are specifically designed for mesh-based models, which lack high-quality texture to render realistic images. NeRF naturally synthesizes realistic faces, but applying sketch to NeRF is challenging. First, due to the domain gap between sparse, monochromatic sketches and real 2D facial images, 2D sketch-based facial modeling is already challenging, not to mention the inference of 3D information from single-view sketch inputs. Furthermore, users may perform local editing operations from different views. Supporting multi-step local manipulations from different views and preserving unedited 3D regions via 2D sketches is not easy to achieve.

In order to generate and edit facial NeRFs from 2D sketches, a possible approach is to first leverage 2D sketch-based image generation methods [**????**] or editing methods [**????**] to generate photo-realistic facial images, and then project the generated images into latent space of 3D GAN such as EG3D [**?**]. However, as shown in Figs. **??** and **??**, these approaches are not robust enough against hand-drawn sketches, so the 2D intermediate faces may have artifacts, which would be inherited by final projection results. Inspired by pSp [**?**], another possible solution is to directly project 2D input sketches into 3D latent space by a CNN encoder. However, this approach tends to overfit synthesized style sketches because of the domain gap, as we will later show in Fig. **??**. To solve the above problems, we translate 2D sketches into 3D tri-plane features [**?**], which supplement sketches with color and stereoscopic information (i.e., volumetric distribution of 3D faces) to reduce the domain gap. The tri-plane feature prediction strategy not only improves the robustness for hand-drawn sketches and adds appearance control but also supports multi-view detailed manipulation because of the representation consistency with the existing tri-plane-based EG3D generator.

We present SketchFaceNeRF, a novel sketch-based facial NeRF generation and editing method. It synthesizes high-quality 3D facial NeRFs from scratch with single-view hand-drawn sketches (see Fig. **??**). As discussed before, instead of directly projecting 2D sketches into 3D latent space, we propose a *Sketch Tri-plane Prediction* net to translate 2D sketches into a compact 3D tri-plane representation. Specifically, with an appearance reference image, we first transform sketches into 2D feature maps, which are then lifted into 3D feature volumes in the 3D Euclidean volume rendering space. Inspired by [**?**], the feature of each 3D position is computed from the 2D feature maps by perspective projection transformation and bilinear interpolation. The tri-plane features are then generated by volume reshaping and convolutions. It is noteworthy that such a module is less sensitive to the style of input sketches due to the involved transformation and projection processes (Fig. **??**). The tri-plane features are concatenated and encoded into the latent space of EG3D to synthesize realistic facial NeRFs.

Given synthesized facial NeRFs (*e.g.*, EG3D samples, sketch-based generations, or real-image inversions), the 3D representation allows users to edit facial details in different views. To solve the challenge of preserving unedited 3D regions during local editing via 2D sketches, we first estimate 2D masks, which indicate edited regions based on the user-performed sketch creation and erasing operations. With the rendered depth maps of NeRF models, a *Mask Fusion* module further lifts the estimated 2D masks into 3D masks, which are used to pick features from the original tri-plane features on the unedited regions and predicted tri-plane features from the edited sketches on the edited regions. The fused tri-plane features are encoded back into the latent space of EG3D to predict an initial edited face. We include an optimization process with a novel space loss term to further ensure faithfulness and consistency for challenging cases (see examples in Fig. **??**). Note that our pipeline can be performed

repeatedly on a single face, supporting multi-step 3D-aware human face editing from different views via hand-drawn sketches.

The main contributions of this work can be summarized as follows:

- We propose the first novel sketch-based 3D facial NeRF generation and editing method, which enables a user-friendly interface for authoring 3D-aware faces and produces photo-realistic results.
- We develop a novel network for translating 2D sketches to 3D facial NeRFs. Single-view sketches are augmented in color and volumetric space to improve the robustness against hand-drawn sketches and allow appearance control.
- We introduce a mask fusion module and a sketch-based optimization approach to achieve detailed editing while preserving the original facial features in unedited regions.

2 RELATED WORK

Our work is related to existing works, including facial NeRF generation and editing, neural sketch-based face generation, and sketch rendering of 3D shapes.

2.1 Facial NeRF Generation and Editing

Generative NeRFs. Existing 2D image generation methods [**??**] randomly sample from a Gaussian distribution to generate high-quality facial images. Utilizing only 2D image datasets, many works further apply this idea to 3D generation with NeRFs [**?**]. For example, GRAF [**?**] first conditions a coordinate-based MLP (Multi-layer Perception) representation in NeRF on the additional shape and appearance codes and utilizes a multi-scale patch-based discriminator instead of a reconstruction loss to train the models. Pi-GAN [**?**] further uses a SIREN-based network [**?**] with FiLM [**?**] conditioning to improve the image quality. However, during GAN training, complete images are rendered instead of individual rays, so the resolution of results is limited due to memory restriction. To address this issue, GIRAFFE [**?**] generates low-resolution feature maps based on volume rendering, followed by a 2D CNN-based network to achieve fast inference and super-resolution. This approach has been extensively used in subsequent works, but the adopted 2D network seriously affects the view consistency. To address this, StyleNeRF [**?**] proposes a specific upsampler combined with a NeRF path regularization loss to reduce the 3D-inconsistent artifacts. Many works propose novel representations of feature fields to improve the quality and efficiency further. For example, StyleSDF [**?**] builds an architecture with an SDF (Signed Distance Field)-based 3D volume renderer to achieve view-consistent facial results with more detailed 3D shapes. GRAM [**?**] represents radiance fields as a set of implicit surfaces, replacing dense Monte Carlo sampling with a few intersection points to render high-resolution images directly. EG3D [**?**] concurrently introduces a lightweight tri-plane 3D representation, combined with a super-resolution network and dual discrimination, to ensure view consistency and image quality. Instead of random sampling, we translate a 2D sketch into a tri-plane representation to support detailed control in facial NeRF generation. To maintain 3D consistency during editing, local swapping and fusion operations are further proposed in the tri-plane space.

Facial NeRF Editing. Besides multi-view image reconstruction or random generation, many works generate radiance fields based on single-view inputs, including RGB images [**?**], and even semantic masks [**?**] or sketches [**?**]. Instead of conditional generation, semantic masks have been further utilized to achieve structure and appearance disentanglement, supporting detailed radiance field editing. FENeRF [**?**] uses two decoupled latent codes to generate spatially-aligned semantics and texture volumes, which share the same geometry but with different discriminators for supervision. Based on the tri-plane representation, IDE-3D [**?**] and NeRFFaceEditing [**?**] generate geometry planes for semantic volume rendering and geometry control and appearance planes for texture control. Unlike contiguous 3D semantic masks, a sketch is view-dependent, especially at the surface contour, making it unsuitable for 3D geometry and appearance decomposition. Besides, the above methods edit facial radiance fields based on time-consuming optimization or single-view encoders. In contrast, our approach can efficiently edit facial NeRFs in free views with a carefully designed prediction and refinement architecture.

2.2 Neural Sketch-based Face Generation

Sketch-based 2D Facial Image Generation. Sketching has been widely used in facial image generation and editing. Since there is a domain gap between freehand sketches and synthetic edge maps, existing approaches [**?????**] introduce various strategies to improve the robustness against different styles of freehand sketches. Instead of synthesizing new faces, sketch-based editing approaches [**???????**] aim to manipulate real facial images while retaining the original identity features. Human-drawn or predicted 2D masks are usually utilized to achieve local editing results. The above approaches only synthesize 2D results, while our method applies sketching to realistic 3D face generation and editing in NeRF. Instead of generating RGB images, we first generate 3D tri-plane features from single-view sketches and then project them into the latent space of EG3D. Similar mask guidance is also adapted into 3D space to support local manipulations.

Sketch-based 3D Facial Model Generation. Many efforts have been made to utilize sketching to design 3D face geometry. One category of methods [**??**] predict the coefficients of a bilinear face representation based on sketches, combined with displace maps [**??**] to manipulate surface details. Another category of approaches [**???**] utilize sketches to guide the template deformation and generate diverse types of 3D models such as animal faces. These previous methods succeed in generating 3D models, but they could not produce photo-realistic face images directly since it is difficult to estimate high-quality texture maps and materials only from sketches. Moreover, most of the above approaches only focus on face regions separately without hair and facial details like pupils. In our method, the 3D-aware hair and pupils are generated together. Thanks to the NeRF representation, our method not only generates photo-realistic 3D faces from 2D sketches, but also constructs high-fidelity facial geometry, as shown in Fig. **??**.

2.3 Sketch Rendering of 3D Shapes

Rendering high-quality sketches of 3D shapes benefits sketch-based modeling since paired training data can be synthesized and analyzed. A differentiable sketch rendering approach can further utilize sketches as constraints to optimize models. To render sketches, a straightforward approach is to first render the depth, normal or RGB maps, and then utilize image-space algorithms [??????] to generate line drawings. However, these image-based methods tend to generate inconsistent results across views because of the insufficient utilization of 3D information. Another branch of methods analyzes the mathematical features of surface geometry and defines different shape-depicted lines [????], which are combined together [?] to generate high-quality sketches. However, these methods require extremely high-quality explicit mesh models as inputs, which are not cheap to obtain from a 3D implicit representation like NeRF with commonly used techniques, such as marching cubes [?] and DMTet [?].

In order to synthesize high-quality sketches in NeRF, neural style transfer [?] is a promising solution. However, there are two main challenges in NeRF style transfer: extensive memory usage caused by rendering whole images during transfer loss calculation and view inconsistency when adapting image transfer algorithms. Chiang et al. [?] first propose a memory-efficient patch sub-sampling algorithm and train a color branch with a fixed geometry branch. StylizedNeRF [?] further designs a mutual learning framework and learnable conditional latent codes to improve the view consistency. Other methods maintain the original NeRF networks but design novel training strategies. For example, SNeRF [?] alternates the NeRF training and stylization optimization steps, consuming less memory in each stage while retaining the original consistency of NeRF. ARF [?] introduces a new view-consistent style loss and a deferred back-propagation method to enable memory-intensive optimization. However, existing methods are specifically designed for single-scene style transfer and cannot be used for generative NeRFs, such as EG3D [?]. Besides, since the above methods stylize a scene based on a single example image, it is unclear how to apply these methods to paired image datasets for stylization.

3 METHOD

3.1 Overview and Preliminaries

Fig. ?? illustrates our sketch-based facial NeRF generation and editing framework. In Sec. ??, we describe the sketch-based facial NeRF generation approach. Taking as inputs a 2D sketch and an appearance reference image, a *Sketch Tri-plane Prediction* net augments the sketch with color and stereoscopic information to synthesize tri-plane features, which are projected into EG3D's latent space to generate high-quality facial NeRFs. In Sec. ??, we describe our sketch-based facial NeRF editing approach. To support free view editing and maintain the identity characteristics in unseen and unedited 3D regions, the tri-plane features synthesized by the edited sketches and the original tri-plane features are fused by a *Mask Fusion* module, and encoded back into the latent space of EG3D. Then, we further refine the rendered result by a latent code optimization with sketch constraints. It should be noted that the sketch-based generation and editing share the same 3D tri-plane prediction and projection networks. The sketch-based generated faces can be further edited to achieve detailed local manipulations from flexible viewpoints, as shown in Fig. ??. This architecture helps users easily design 3D facial NeRFs with detailed control.

We build our approach based on EG3D [?], a pretrained 3D face NeRF generator. Given the latent code w, three orthogonal plane features (referred to as tri-plane features) p_{xy}, p_{xz}, and p_{yz} are generated from the StyleGAN2 [?] backbone. Each 3D queried position $x \in \mathbb{R}^3$ is projected onto the three feature planes to get the corresponding features F_{xy}, F_{xz}, and F_{yz} via bilinear interpolation, and such features are then aggregated into 3D features F via summation. An image decoder interprets F to color features and densities, with the subsequent volume rendering [?] to synthesize low-resolution feature maps whose first three channels are RGB images. A super-resolution module further translates the feature maps into high-resolution images. More details of the generator can be found in [?]. Our method generates 3D faces in EG3D's $\mathcal{W}^+$ space, composed of 14 different latent codes. The efficient tri-plane representation is also utilized in our 3D feature representation.

3.2 Sketch-based Facial NeRF Generation

Faithfully translating hand-drawn 2D sketches into realistic 3D faces is an attractive but challenging task. A naïve approach is to directly encode the input 2D sketches into the latent space of EG3D with a 2D encoder (e.g., [?]) and utilize style-mixing [??] to control the appearance. However, since the encoder is originally designed for 2D GANs and the style-mixing cannot exactly control the appearance in EG3D [?], such a solution is not robust to the input sketches at details such as hairstyles and does not guarantee accurate appearance styles, as shown in Fig. ??. In contrast, we are motivated to first lift 2D inputs into 3D inputs and then encode them into 3D outputs. The tri-plane representation is utilized because of its high expressive capacity and adaptability for 2D convolution in encoders.

In particular, to lift 2D inputs into 3D, we draw inspiration from PixelNeRF [?], which is devised to infer 3D information from sparse 2D inputs. Given a single-view hand-drawn sketch S, we design a *Sketch Tri-plane Prediction* net to lift S into a 3D tri-plane representation p^s, including feature maps p^s_{xy}, p^s_{xz}, and p^s_{yz}, which is expected to lie in the same distribution as the EG3D-generated tri-plane representation. The input sketch only contains geometry information, but the tri-plane features and rendered images have diverse appearance details. So we first translate the input sketch S to a colorized feature map f_c to inject colors, lighting, and texture information. We train an appearance encoder E_c to extract the appearance information from the reference image I. To transfer the appearance to S, we leverage a translation network G_c with adaptive instance normalization [?] (AdaIN) to generate the colorized feature map:

$$f_c = G_c(S, E_c(I)). \tag{1}$$

To predict a 3D tri-plane representation from a 2D feature map, we build a 3D feature volume in the Euclidean space where the volume rendering is performed. Using the estimated camera intrinsics and extrinsics, each point x in the 3D volume is projected onto the input image space to get 2D coordinates $\pi(x)$. Then, the corresponding

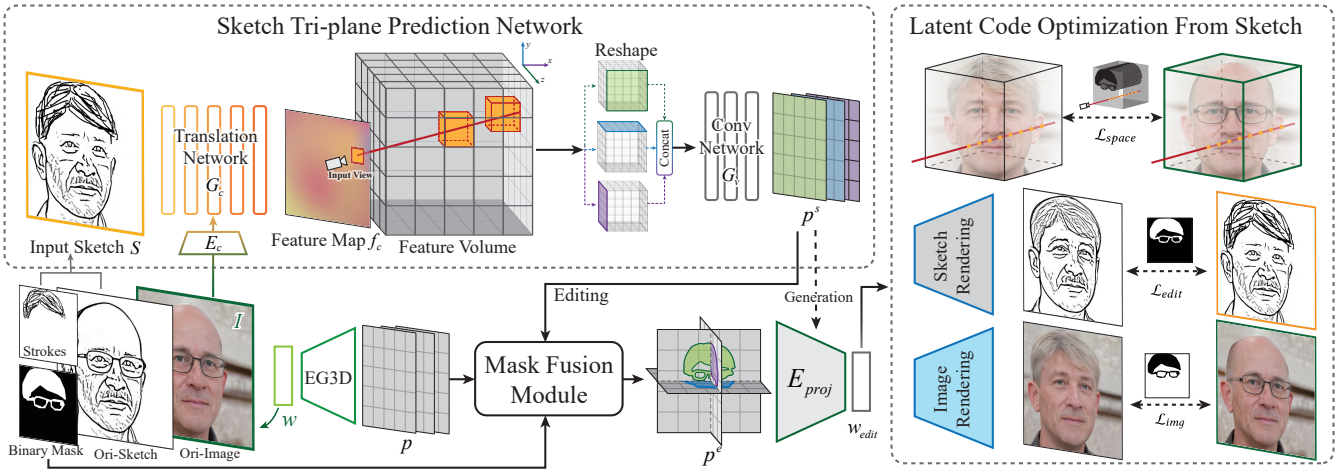


Fig. 2. Overview of our unified sketch-based facial NeRF generation and editing framework. Given an input sketch S, the *Sketch Tri-plane Prediction* Network generates a tri-plane feature representation p^s for 3D information. The color information is supplemented by an appearance encoder E_c and a translation network G_c to generate a colorized feature map f_c, and the stereoscopic information is supplemented by G_v. To generate a facial NeRF from scratch (indicated by the dotted line below p^s), the tri-plane feature p^s is directly encoded by E_{proj} to generate 3D faces. For facial NeRF editing (indicated by the solid line below p^s), the *Mask Fusion* module fuses p^s and the original tri-plane feature p to generate p^e, which is also encoded by the shared E_{proj}. Moreover, we propose a sketch-based optimization approach to improve editing faithfulness and original feature retention further.

feature vector $f_c(\pi(x))$ is retrieved for each point x via bilinear interpolation from colorized feature maps f_c. To balance performance and efficiency, we build the feature volume with a resolution of 128 in each axis, resulting in the final shape of $128 \times 128 \times 128 \times 3$, where 3 is the channel number of feature maps f_c. The 3D feature volume is further reshaped along each axis to generate three $128 \times 128 \times 384$ feature maps, denoted as V_{xy}, V_{xz}, and V_{yz}. These feature maps are concatenated in the feature channel, using a 2D convolution network G_v to upsample and translate them into a $256 \times 256 \times 96$ sketch feature map, which is split channel-wise and reshaped to form three 32-channel feature planes p^s_{xy}, p^s_{xz}, p^s_{yz}, abbreviated as the tri-plane feature p^s:

$$p^s = G_v(V_{xy}, V_{xz}, V_{yz}). \tag{2}$$

Although our *Sketch Tri-plane Prediction* net thoroughly analyzes the 3D information, only 2D convolution is utilized here to improve memory and time efficiency. After lifting the 2D inputs into 3D as the tri-plane representation, we utilize a 2D encoder E_{proj} [?] to project the tri-plane features into the $\mathcal{W}^+$ space of EG3D to improve the quality, which renders final photo-realistic free-view facial images.

Training Objective. The *Sketch Tri-plane Prediction* net is trained using a synthesized multi-view dataset. Given a set of latent codes, ground-truth tri-plane features p are synthesized based on EG3D. For each example, we randomly sample multiple camera poses to synthesize paired sketches and images, using the sketch generation approach discussed in Sec. **??**.

Given an input single-view sketch, the *Sketch Tri-plane Prediction* net synthesizes a tri-plane feature p^s, which generates images I^s through volume rendering with the original decoder of EG3D [?]. These rendered images have different views t from the input sketch. The above strategy enhances the 3D information by enforcing the network to imagine faces from other views, as in [?]. The loss function $\mathcal{L}(E_c, G_c, G_v)$ to train the *Sketch Tri-plane Prediction* net is defined as:

$$\mathcal{L}(E_c, G_c, G_v) = \beta_1 \mathcal{L}_1(p^s, p) + \beta_2 \mathcal{L}_1(I^s, I^t) + \beta_3 \mathcal{L}_{VGG}(I^s, I^t), \tag{3}$$

where $\mathcal{L}_1$ denotes the L1 distance, $\mathcal{L}_{\text{VGG}}$ denotes the perception distance [?], and I^t is the ground-truth image in a target view. For high-quality results, we both constrain the synthesized tri-plane representation p^s to be in the same distribution as the ground-truth tri-plane features p, and measure the similarity between the rendered images and ground-truth images. In our experiments, we set $\beta_1 = 0.01$, $\beta_2 = 1.0$, and $\beta_3 = 0.1$.

To train the encoder E_{proj}, we use the same strategy of pSp [?], but instead of RGB images, it takes concatenated tri-plane features p^s as input. Both of the tri-plane features generated by the original EG3D backbone and the *Sketch Tri-plane Prediction* net are fed into the network to improve the generalization of E_{proj}. Pixel-wise L2 loss, LPIPS loss [?], identity loss [?], and regularization loss are used for training such an encoder. Please refer to [?] for more details. We first train the *Sketch Tri-plane Prediction* net to convergence and then train the encoder.

3.3 Sketch-based Facial NeRF Editing

Although the generation of 3D faces from 2D sketches is qualified for applications like character design, users may desire to further adjust 3D faces by interactively editing the corresponding 2D sketches from different views, as in [?]. This motivates us to design a sketch-based interface for facial NeRF editing, which allows 2D local editing from different views and achieves consistent 3D editing effects while preserving the unedited regions.

3.3.1 Free-view Sketch Generation. First of all, we synthesize free-view rendering sketches which users can modify to edit the corresponding 3D faces. This process is differentiable and further used for latent code optimization as discussed in Sec. **??**. An additional sketch generation path, which has a similar architecture to the image generation path of EG3D [**?**], is therefore designed to generate the corresponding sketch at any view of a latent code. The sketch and image rendering branches share the same StyleGAN backbone and projection features but have separate decoders and super-resolution modules. Specifically, a new sketch decoder interprets the sample features to sketch features, combined with the density of the image branch to generate low-resolution sketch feature maps based on volume rendering. The 1st channel of the sketch feature maps corresponds to low-resolution sketches (128×128), denoted as $S^{'}_{raw}$. A sketch super-resolution module similar to the original super-resolution module in EG3D is also used to synthesize final sketches $S^{'}$.

To train such sketch generation path, a pretrained pix2pixHD [**?**] is used to convert rendered facial images into ground-truth high-resolution sketch S_{GT}, which is, however not 3D-consistent. We carefully design training losses to ensure consistency from different views and synthesize high-quality sketches. A reconstruction loss is used to match the original sketch distribution:

$$\begin{aligned}\mathcal{L}_{recon} &= \alpha_1 \mathcal{L}_1(S^{'}, S_{GT}) + \alpha_2 \mathcal{L}_1(S^{'}_{raw}, S_{raw}) \\ &+\alpha_3 \mathcal{L}_{VGG}(S^{'}, S_{GT}) + \alpha_4 \mathcal{L}_{VGG}(S^{'}_{raw}, S_{raw}),\end{aligned} \tag{4}$$

where S_{raw} represents the downsampled sketch of a high-resolution sketch S_{GT}. In our experiments, we empirically set $\alpha_1 = \alpha_2 = 3.0$ and $\alpha_3 = \alpha_4 = 2.0$.

Inspired by [**?**], we utilize a regularization term to enforce the 3D consistency of sketches. Although the predicted ground-truth sketch S_{GT} is not view-consistent, the inherent multi-view consistency of volume rendering constrains the final results of the super-resolution module. This loss term compares the sub-sampled pixels on the final sketch results and those generated by NeRF:

$$\mathcal{L}_{view} = \alpha_5 \frac{1}{|C|} \sum_{(i,j)\in C} \left\| S^{'}[i,j] - Render(r_{i,j}) \right\|_1, \tag{5}$$

where C is the set of randomly sampled pixels in final sketches and i, j are the coordinates in the high-resolution 2D sketch space, $Render(r_{i,j})$ is the pixel result of direct volume rendering with the corresponding ray $r_{i,j}$. In our experiments, $\alpha_5 = 3.0$ and $|C| = 8,192$.

The final training objective is:

$$\mathcal{L}_{sketch} = \mathcal{L}_{recon} + \mathcal{L}_{view}. \tag{6}$$

During the training of the free-view sketch generation, the weights of the StyleGAN backbone and the image generation path are fixed. We only update the weights of the sketch decoder and sketch super-resolution module.

Finally, users are able to directly edit the rendered sketches (adding or removing strokes) to obtain edited sketches S instead of drawing sketches for every view from scratch.

3.3.2 Mask Fusion Module. Given the modified sketch S, the introduced *Sketch Tri-plane Prediction* net together with E_{proj} in Sec. **??** is already able to generate high-quality 3D faces. However, unedited regions might suffer from undesirable changes despite local editing operations on the 2D sketch since the 2D sketch at a certain view cannot describe the entire 3D object, as demonstrated in Fig. **??**. Targeting this issue, we propose a *Mask Fusion* module to spatially fuse the original 3D face generated from the input latent code w, and the newly generated 3D face based on the edited sketch with a 3D mask, preserving the unedited regions and retaining the edited regions. Notice that since both faces are represented as tri-plane features, the fusion can be conducted on tri-plane features directly. Subsequently, the spatially fused 3D face is encoded back into the latent space of EG3D as w_{edit}. It should be noticed that the input latent code w can be an EG3D sample, real image projection, or the previously edited result to apply multi-step manipulations from different views.

To obtain the guiding 3D fusion mask, after users perform the editing operations via our interface, we first estimate a 2D binary mask M indicating the edited regions (introduced in Sec. **??**). For the input latent code w, the original tri-plane representation (p_{xy}, p_{xz}, p_{yz}) is synthesized based on the EG3D backbone along with a generated depth map D. Based on the depth map D, each pixel in M is converted to its 3D location in the Euclidean space, forming a set of 3D points. Since the fusion is essentially conducted on the tri-plane features, such a set of 3D points are projected into the tri-plane space to synthesize three sets of 2D points. For the modified sketch S_{edit}, a new tri-plane representation (p^s_{xy}, p^s_{xz}, p^s_{yz}) is synthesized by the *Sketch Tri-plane Prediction* net along with its generated depth map, and another three sets of 2D points are obtained similarly. By uniting two sets of 2D points on each plane, dilating them for a smoother border, and connecting sufficiently close regions, we obtain the guiding 3D fusion mask, which is composed of three 2D fusion masks on each plane, denoted as M_{xy}, M_{xz}, M_{yz}.

Then, the original and predicted tri-plane features are fused as:

$$P^e_{xy} = M_{xy} \cdot p^s_{xy} + (1 - M_{xy}) \cdot p_{xy}. \tag{7}$$

Here we only take xy as an example, and operations on the other two planes are the same. We denote the generated new fused tri-plane features as $p^e_{xy}, p^e_{xz}, p^e_{yz}$. It should be noticed that swapping a region in the tri-plane features affects the whole orthogonal 3D columnar space, so the fused tri-plane features cannot be directly used for volume rendering. To solve this problem, we utilize the same encoder E_{proj} in Sec. **??** to further project the fused tri-plane feature into the $\mathcal{W}^+$ space, as $w_{edit} = E_{proj}(p^e_{xy}, p^e_{xz}, p^e_{yz})$.

3.3.3 Latent Code Optimization from Sketch. We further refine w_{edit} to ensure better sketch faithfulness and original feature retention in challenging cases. For example, the predicted hair structure or eyeglass shapes may have small biases with drawn sketches, as shown in Fig. **??**. The background of the original faces is sometimes too sophisticated to reconstruct. Our optimization strategy is designed to solve these problems in challenging editing situations.

Editing Optimization. Inspired by [**?**], we propose an optimization approach to refine the predicted latent code w_{edit}. With the image rendering branch $\mathcal{R}_x$ and the sketch rendering branch $\mathcal{R}_s$, the following loss terms are optimized:

$$\mathcal{L}_{edit} = \mathcal{L}_{VGG}(\mathcal{R}_s(w_{edit}) \odot M, S \odot M), \tag{8}$$

$$\mathcal{L}_{img} = \mathcal{L}_{VGG}(\mathcal{R}_x(w_{edit}) \odot \widetilde{M}, \mathcal{R}_x(w) \odot \widetilde{M}), \tag{9}$$

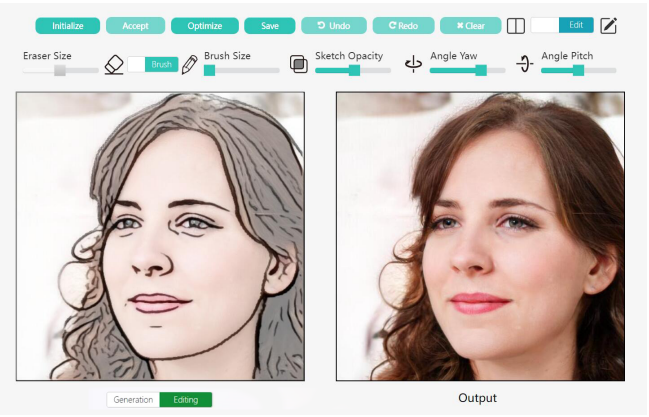


Fig. 3. The user interface of SketchFaceNeRF. The interface has two modes. In the generation mode, the left window is used for creating freehand sketches. In the editing mode, the left window is used for editing line drawings from a 3D face. In both modes, the right window shows the generated face. A control panel at the top of the interface supports many essential operations, including the selection of pencil or eraser, brush size control, and rotation of viewpoints.

where $\widetilde{M}$ refers to the unedited regions, and $\odot$ denotes pixel-wise multiplication. These loss terms encourage the edited regions to have consistent sketches as S and remaining regions to be unaltered as much as possible.

Utilizing the above loss terms is adequate to solve the 2D image editing problem but not enough for 3D face editing in NeRF, since the stereoscopic features should be maintained to ensure that unedited regions are retained in arbitrary views. One possible approach is to add multi-view losses, but viewpoint selection and view-specific 2D mask transformation are difficult. So, we propose a novel space loss term to measure the similarity of sample points' features utilized in volume rendering:

$$\mathcal{L}_{space} = \frac{1}{N|\widetilde{M}|}\sum_{r}^{\widetilde{M}}\sum_{i}^{N} \|\Phi(w_{edit}, r(i)) - \Phi(w, r(i))\|_1, \qquad (10)$$

where $r(i)$ is the ith sample point along the rendering ray r in unedited regions, N is the number of sample points, and Φ denotes the point-wise feature calculation process, including tri-plane projection, and feature decoding. Hierarchical volume sampling [**?**] is used in NeRF rendering, while we only calculate the space loss on coarse samples which have the same position for different identities. The overall loss function for optimization is:

$$\mathcal{L}(w_{edit}) = \gamma_1\mathcal{L}_{edit} + \gamma_2\mathcal{L}_{img} + \gamma_3\mathcal{L}_{space}, \qquad (11)$$

where γ_1, γ_2, and γ_3 are hyper-parameters tuned by users. In our experiments, they are set as $\gamma_1 = 40$, $\gamma_2 = 20$, and $\gamma_3 = 0.2$ by default, and the iteration steps are set as 10 with the balance of efficiency and quality.

4 USER INTERFACE

As shown in Fig. **??**, we design a user interface on top of the proposed pipeline to support sketch-based facial generation and multi-step editing in different views. To generate facial NeRFs from scratch, users draw sketches on the left drawing canvas, and our system then synthesizes and displays photo-realistic 3D faces in the right window. Our interface further supports detailed facial editing via editing sketches synthesized from previously generated 3D faces (Fig. **??**), EG3D random samples, and real face images (Fig. **??**). Users can change the views with the sliders on the control panel, during which process the generated 3D faces and corresponding sketches are rotated simultaneously. Thanks to our free-view sketch generation approach, the synthesized sketches are consistent during view changes, thus improving the user interaction experience. During editing, users may erase undesired lines and draw new lines depicting desired structures. These operations provide adequate information to infer the mask M representing the edited regions. Specifically, we dilate the newly drawn lines and unite them with the erasing regions to generate an initial mask. Then, the small holes within the edited regions are filled by connection detection, and the border is smoothed by polygonal curves. With the input face, modified sketch, and inferred mask, our algorithm generates new edited NeRF faces, which are rendered and shown in our UI system. After editing in a single viewpoint, users can rotate the face and continuously edit it in other views, supporting detailed and expressive facial manipulation.

5 EVALUATION

In this section, a series of qualitative and quantitative experiments are conducted to demonstrate the superiority of our framework. In Sec. **??**, we show the sketch-based facial NeRF editing and generation results of our method. In Sec. **??**, the qualitative and quantitative comparisons with state-of-the-art methods are conducted to demonstrate the better performance of our approach. In Sec. **??**, we conduct an ablation study to validate the effectiveness of each module and network design in this framework. A user study is presented in Sec. **??** to further prove the superiority of our approach.

Implementation Details. To train the facial NeRF manipulation framework, we synthesize a multi-view dataset based on EG3D with 110k training samples. For each example, 25 rendered images from different views are generated while the tri-plane features are synthesized on the fly during training. Our networks are trained and tested on an NVIDIA RTX 3090 GPU. During optimization, we use the ADAM [**?**] optimizer with a learning rate of $5e-3$ and one iteration costs 0.18s on our device. More details of the dataset and training settings can be found in supplementary materials.

5.1 Results

Our method supports high-quality facial NeRF generation based on single-view sketches and appearance reference images. We treat the facial NeRF generation from scratch as a special editing situation where the predicted tri-plane features are directly projected into the latent space without mask fusion and optimization. The viewing angles of hand-drawn sketches are estimated by [**??**]. As shown in Fig. **??**, given the hand-drawn sketches that represent the facial geometry details, including the facial component shapes, hair structures, and beard, our approach generates high-quality 3D geometry models with good faithfulness for sketches. Although the hand-drawn sketches have various drawing styles that are different

Fig. 4. Facial NeRF generation results given hand-drawn sketches. The input sketches and generated geometry are shown in the first two columns. Photo-realistic rendering results with free viewpoints are shown in the following columns, with different appearance images in the top-left corner of each example. Our method generates detailed geometry by sketches and controls the appearance with reference images. Since we focus on facial region generation, the background of reference images is masked. The generated images have semi-random backgrounds entangled with faces in the EG3D space.

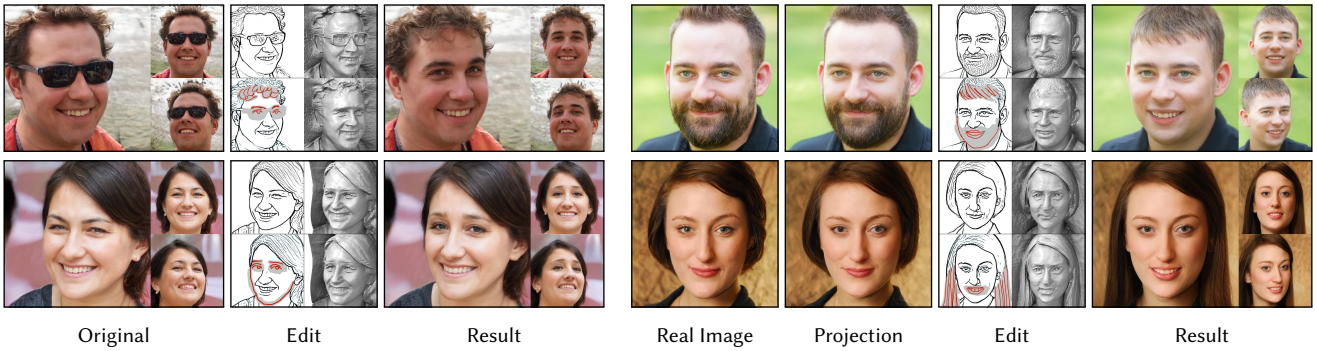


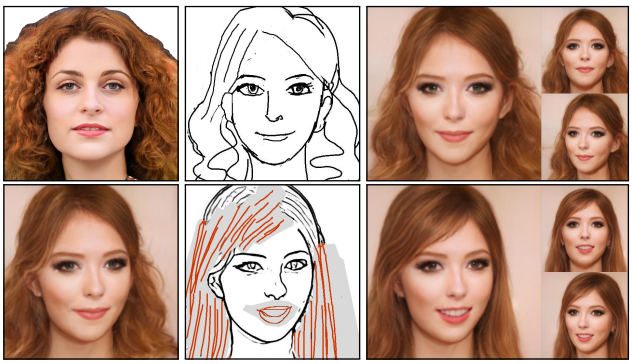
Fig. 5. Interactive Facial NeRF editing results. Users can interactively select an arbitrary view and then edit 3D faces by modifying the rendered sketches. Hand-drawn sketches are labeled in red, and the gray regions are the inferred masks of the user interface. Some editing examples are shown in this figure, e.g., hairstyle, glasses, beard, eyes, and expression. Our method edits the local regions while maintaining the global features of the original faces. Real images can be projected and edited to synthesize free-view results.

from those in the training dataset, our method is robust and can still generate high-quality facial models. Sketches only provide geometry information but lack color information, so our face regression network controls the appearance with example images. Facial NeRF models with different colors, materials, and lighting are synthesized, and the viewpoints can be freely controlled by users, as shown in Fig. **??**. More generation results can be found in the supplementary materials.

With our carefully designed framework and user interface, users can interactively edit facial details via sketches from free viewpoints. After the sketch modification using brush and eraser tools, the user interface automatically infers the edited masks (colored as gray in Fig. **??**). As illustrated, the masks accurately label locally edited regions. High-quality results are generated by our method with various types of editing operations, including adding/removing eyeglasses, changing hair structures, and modifying facial shapes or expressions. The edited regions show good editing effects, while the global features are well maintained. As shown in Fig. **??**, our system generates good multi-step editing results for a single example from multiple views. The editing operations performed from different views are all effective. The results show no deterioration with the accumulation of editing operations thanks to our tri-plane projection and optimization approaches. More editing results with different drawing styles can be found in our supplementary materials.

5.2 Comparison

Sketch-based Facial NeRF Generation. Since our method can generate high-quality facial NeRFs based on single-view sketches, we compare it with possible existing sketch-based facial NeRF generation approaches, with some adaptations. PixelNeRF [**?**] synthesizes NeRFs with the input of single-view images, which are replaced with single-view sketches in our experiments to support our task. As shown in Fig. **??**, this approach cannot control the face appearance and is not robust for hand-drawn sketches with fuzzy details. DeepFaceEditing [**?**] synthesizes face *images* for hand-drawn sketches and controls the appearance accurately, but there are still artifacts



(a) Image (b) Sketch (c) Results

Fig. 6. Sketch-based generation (1st row) and editing (2nd row) for facial NeRF design. The 1st row shows a sketch-based generation example based on an appearance image (a) and a hand-drawn sketch (b). Users can further apply detailed editing via a synthesized sketch (b) while maintaining the original identity characteristics.

around the neck and hair regions. Since the data-driven manifold projection is utilized, some less common appearances, like the big curly hair (3rd row) and bangs (4th row), cannot be synthesized well. We further project the results of DeepFaceEditing into EG3D's latent space to generate NeRF results, as shown in the 5th column in Fig. **??**. Based on the pretrained generator, the projection results are more realistic but still have low faithfulness with input sketches, e.g., the mistaken hair structures. pSp [**?**] is a new image translation approach based on 2D StyleGAN [**?**]. For fair comparison, we replace the StyleGAN with the EG3D generator and utilize the style-mixing to swap the last 7 layers of latent codes to support appearance control. Although this approach generates good results on synthesized sketches (see supplementary material), it is not robust for hand-drawn sketches and has poor geometry faithfulness. The style-mixing also cannot control the appearance accurately in 3D GAN because of the complicated rendering process. Our method is the first approach to synthesizing facial NeRFs from sketches and has better results than possible baselines.

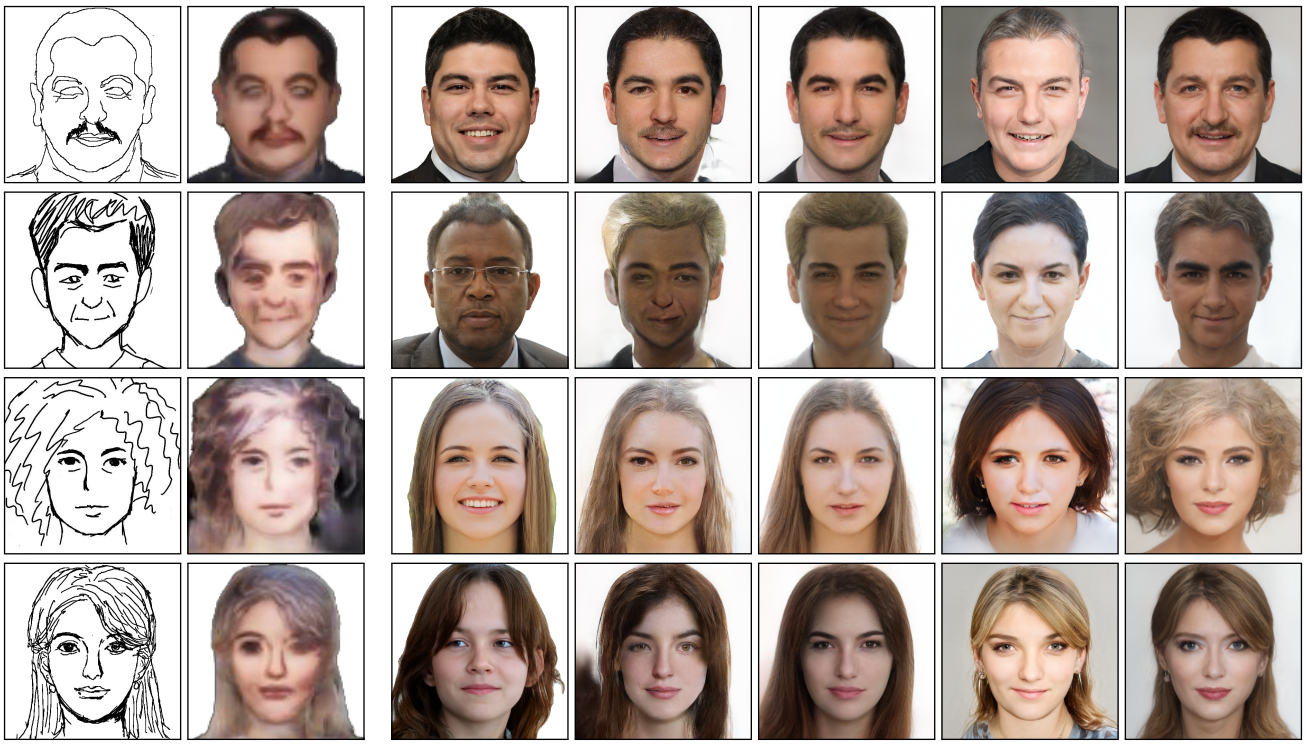
Sketch-based Facial NeRF Editing. Our method supports detailed sketch-based editing of 3D human faces and generates high-quality editing results in given views. Thus, we compare it with existing sketch-based face editing methods. However, due to the 2D nature of the existing methods, they are not 3D-aware as our method since we are the first to edit 3D human faces by sketch. As shown in Fig. **??**, given original facial images (a) and edited sketches (b), DeepPS [**?**] dilates the sketches to achieve higher-quality results than those without dilation but compromises the faithfulness, e.g., failing to turn the straight hair into curly hair in the second row. Besides, it has obvious artifacts near the boundary of the masks due to its inpainting fusion. DeepFaceEditing [**?**] produces reasonably edited results, but the image quality degrades with complex editing manipulations. Its results also exhibit artifacts such as darkened areas on the eyes (1st row) and forehead (2nd row) because of the local appearance disturbance of the original images. SketchEdit [**?**]

(a) Original (b) Sketch (c) Result (d) Result Front

Fig. 7. Results of multi-step editing (from top to bottom) in different views. In (a), the topmost image is the original face, and the rest are previously edited results for further editing. Modified sketches and generated results are shown in (b) and (c), respectively. The editing manipulations are added in different views. Our method well maintains the original features in unedited regions (d) and avoids deterioration even though the results are recursively used.

generates unrealistic edited results for glasses removal and curly hair, despite its efforts to estimate the masks for the edited regions. In contrast, our method produces high-quality and 3D-aware facial images rendered from different views. Additionally, our method is faithful to the edited sketches and preserves the untouched regions well. The reason is that we carefully predict the edited tri-plane features directly from the edited sketches for faithfulness, fuse them with the original tri-plane features for consistency, and encode the fused tri-plane features back to the 3D-aware generative prior to improve the quality. We also include the optimization process to ensure faithfulness and consistency better. Notice that our method also does not require laborious manual mask drawing, similar to SketchEdit.

Quantitative Comparison. To measure the facial NeRF generation and editing quality of the compared approaches, we report the Fréchet Inception Distance (FID) [**?**] and Kernel Inception Distance (KID) [**?**] in Tables **??** and **??**. For sketch-based facial generation, we collected 100 hand-drawn sketches, which were shared by the



(a) Sketch (b) PixelNeRF (c) Appearance (d) DFE (e) DFE-Projection (f) pSp-Ref (g) Ours

Fig. 8. Comparisons with state-of-the-art methods for hand-drawn sketches to facial NeRF translation. In each row, (a) is a user-drawn sketch, and (c) is an appearance reference image. PixelNeRF (b) cannot control the appearance and generates blurry results. Other existing methods (d)~(f) generate results with the input of reference appearance but have poor faithfulness with sketches (d,e) or appearance (f), while our method (g) generates the best results. DFE and DFE-proj are the abbreviations of DeepFaceEditing [?] and its NeRF projection version.

(a) Image (b) Sketch (c) DeepPS (d) DeepFaceEditing (e) SketchEdit (f) Ours

Fig. 9. Comparisons with state-of-the-art methods for sketch-based facial editing. Given original images (a) and modified sketches (b), DeepPS generates plausible results (c) in edited regions but has obvious artifacts on the editing region boundaries. DeepFaceEditing generates results (d) with the appearance of input images (a), thus causing color bias in local regions when removing the hair and glasses. SketchEdit is not robust against hand-drawn sketches and synthesizes blurry results (e). Our method not only generates better results (f) in the original views than the other approaches but can also render realistic free-view results.

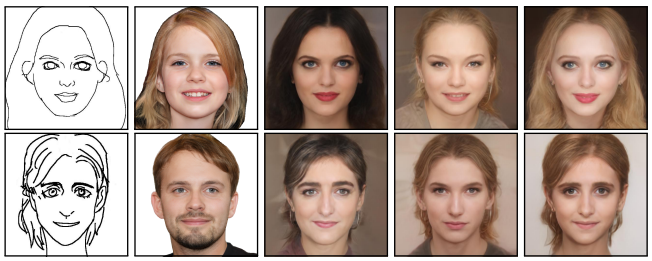
authors of DeepFaceDrawing [?] and collected with their online demo system. As shown in Table ??, our method outperforms the

	PixelNeRF	DFE	DFE-proj	pSp	pSp-Ref	Ours
FID↓	189.30	77.63	97.94	94.15	80.69	**72.63**
KID↓	16.25 ± 0.2	**1.98± 0.2**	2.99 ± 0.2	4.52± 0.2	2.56± 0.2	2.06± 0.2

Table 1. Quantitative results compared with sketch-based facial generation approaches. We report the FID and KID mean×100 ± std.×100. Our results have lower (i.e., better) values compared with other approaches, except comparable values with DeepFaceEditing (DFE), which, however, is designed only for 2D images instead of NeRF.

	DeepPS	DFE	SketchEdit	Ours
FID↓	108.8	98.65	112.51	**87.68**
KID ↓	5.94 ± 0.5	4.23 ± 0.5	6.44 ± 0.5	**4.22** ± 0.4

Table 2. Quantitative results compared with sketch-based face editing approaches. We report the FID and KID mean×100 ± std.×100. Our results have the lowest values among all the compared approaches, indicating the best image quality.



(a) Sketch (b) Appear (c) w/o Color (d) w/o Volume (e) Ours

Fig. 10. Ablation study of the *Sketch Tri-plane Prediction* net given hand-drawn sketches (a) and reference appearance images (b). Without colorization, the appearance cannot be controlled (c). Without the feature volume and directly predicting tri-planes features based on input sketches, the results have low faithfulness (d) with the input sketches. Our method generates the best results (e) based on the sketches and appearance images.

other approaches except for the comparable KID compared with DeepFaceEditing [**?**], whose results are only 2D images and have low sketch faithfulness, as shown in Fig. **??**. After projection, the artifacts are accumulated and have worse value results. For sketch-based facial editing, we collect 50 editing examples based on the user interface (Fig. **??**). As shown in Table **??**, even though our method is designed explicitly for NeRF editing, it outperforms all the state-of-the-art image editing methods at the manipulation viewpoints. During the quantitative calculation, the background is masked out because we only focus on the quality of facial regions.

5.3 Ablation Study

We conduct ablation studies to prove the effectiveness of each component in our framework. The key components of the *Sketch Tri-plane Prediction* net and *Mask Fusion* module are disabled to show their impacts. Then, the loss terms in sketch-based optimization are evaluated respectively to prove their effectiveness. We also replace the sketch generation approach with other approaches to evaluate the view consistency of our 3D sketches.

In the *Sketch Tri-plane Prediction* net, the sketches are translated into feature maps to supplement color information. As shown in Fig. **??**, the appearance is unable to control without such an appearance transfer process, inconsistent with the appearance reference images. Additionally, the stereoscopic information is added by lifting the 2D feature maps into 3D feature volumes through space projection. Without such a lifting process to enhance the 3D information, the encoded results suffer from loss of faithfulness, especially for hair and small details such as eyebrows, since it is hard to directly encode latent codes for 3D models from 2D inputs. In contrast, our full model generates the best results regarding both geometry faithfulness with the input sketches and appearance faithfulness with the appearance reference images.

As to the *Mask Fusion* module, without negatively affecting the editing effects, the fusion operation solves identity distortion in the original view and preserves unedited regions in other views. As shown in the first row in Fig. **??**, the baseline without the fusion strategy predicts images that exhibit subtle distortions on the background and facial shape in the original view (b). When we rotate it into the front view, the hair is also totally changed (d). Although part of the original features can be restored with the optimization approach, the final results (c)&(g) still have subtle differences from the original facial NeRF in unedited regions, such as hair details and background patterns. Our approach can well preserve the original features with fewer steps compared with the baseline approaches, proving the effectiveness of the *Mask Fusion* module.

An optimization process is included to address challenging editing cases and enhance the correspondence between the edited sketches and results in terms of details. In Fig. **??**, it is obvious that without such an optimization process, the encoded results are acceptable but differ from the desired sketches in small details, such as the area of hair in the first row, and the shape of eyeglasses in the second row. However, if we remove the encoding module, i.e., by directly optimizing the latent codes based on the initial latent codes and edited sketches, the optimized results have very low consistency with the desired sketches despite long optimization steps. For example, the hair barely changes in the first row, and the glasses cannot be added in the second row. We introduce several loss terms to ensure faithful and consistent editing effects. Without $\mathcal{L}_{\text{img}}$, the details, such as the beard in the first row, fail to be preserved. Without $\mathcal{L}_{\text{edit}}$ to guide the desired shape, the optimized results do not follow the edited sketches.

As illustrated in Fig. **??**, without the novel space sample loss term, acceptable editing results are still achieved at the original views, and the unedited regions are also well preserved. However, from the perspective of 3D human faces instead of 2D facial images, such 3D models are subject to substantial geometry changes, which can be detected from the frontal views. We also test the sketch generation approach by replacing it with Pix2PixHD [**?**] to directly predict sketches from the rendered images. However, since Pix2PixHD is quite heavy and not robust, the predicted sketches have a relatively indirect connection with the underlying 3D human faces, making the optimized results have similar effects to the initial predicted ones, as shown in Fig. **??**. In comparison, our full optimization setting makes up for the challenging details, such as the height of hair and

Fig. 11. Ablation study of the Mask Fusion module. The original image and drawn sketch are shown in (a). The mouth is closed in this example. As shown in the 1st row, without the *Mask Fusion* module, the predicted (b) and optimized images (c) have different backgrounds from the original images. When we rotate into the front view, the predicted (d) and optimized faces (e)~(g) have different haircuts compared with the original faces. As shown in the 2nd row, our method generates the same background (c) in the original view. The front-view faces well retain the original features with fewer optimization steps compared with baseline approaches.

Fig. 12. Ablation study of optimization approaches proposed in Sec. **??**. Input images (a) and modified sketches (b) are shown in the first two columns, with the gray regions indicating the inferred masks by our method. Direct prediction results (c) without optimization are acceptable but have a little bias with drawn sketches. Optimization started from the original latent (d), and without L_{edit} (f) has limited effects, and removing L_{img} (e) changes unedited regions. Utilizing Pix2PixHD [?] to calculate sketch loss has similar results as the initial predicted results, while our approach further improves the faithfulness to sketches as seen in the hair patterns in the first row and the shape of glasses in the second row.

	Pix2PixHD	UPD	w/o L_v	Ours
Inconsistency↓	0.108	0.084	0.064	**0.056**

Table 3. Consistency evaluation of 3D sketches. Our results have a lower inconsistency score than the baseline approaches Pix2PixHD [?] and UPD [?], meaning that our sketches have the best consistency across views.

the shape of eyeglasses, while preserving the unedited regions in 3D space.

We add a new sketch generation approach to synthesize 3D sketches based on EG3D's latent codes. As shown in Fig. **??**, the synthesized sketches are displayed in the UI system and rotated with the synthesized facial images simultaneously. So, sketch consistency during view changes affects users' interaction experience. We replace our sketch generation approach with two image-to-sketch translation methods, including Pix2PixHD and Unpaired Portrait Drawing (UPD for short) [?]. The regularization term we used in the training stage cannot be applied to these methods since they are not 3D-aware. A short-range consistency score is measured as in [?] to measure the inconsistency during view changes. We do not use a long-range consistency score since sketches are view-dependent and naturally vary with large view changes. We randomly generate 30 faces to calculate the metric and show the results in Table **??**. It can be seen that our sketches have the best view consistency compared with the alternative approaches. UPD generates sketches with three different styles, so we report the lowest value among them in Table **??**.

5.4 User Study

Given the above extensive qualitative and quantitative comparisons, we find it beneficial to conduct a perception study to fully testify our method from the perspective of human viewers. Specifically, we evaluate our method on two tasks: facial generation from hand-drawn

(a) Image (b) Sketch (c) w/o L_{space} (d) Ours (e) Front Image (f) Front w/o L_{space} (g) Front Ours

Fig. 13. Ablation study of the space sample loss during optimization. In the 1st row, the double chins of a girl are removed. In the 2nd row, the nose bridge is raised, and the hair bangs are added. The methods with and without L_{space} both generate good results observed from the editing views. However, when observed from other views, the results without L_{space} have undesirable changes in unedited regions, such as the hair patterns in the first row and the left hair in the second row.

2D sketches, and facial editing by modifying the corresponding 2D sketches.

For the facial generation, we compare our method against the same set of state-of-the-art methods in the qualitative comparison of sketch-based facial NeRF generation, i.e., PixelNeRF, DeepFaceEditing (denoted as "DFE"), DeepFaceEditing followed by NeRF projection (denoted as "DFE-Projection"), and pSp (denoted as "pSp-Ref"). We prepare 15 cases to cover as much diversity (such as the drawing style and personal attributes, including age, gender, hairstyle, etc.) as possible, each of which consists of an input 2D hand-drawn sketch, a reference appearance facial image and the facial images generated by the compared methods. Since these methods are not all 3D-aware, we only display the results rendered from the viewpoint of the input sketches, in random order. However, it is noteworthy that our method further supports face rotation due to the 3D-aware NeRF representation. Users are invited to rank the generated facial images in order (the lower, the better) from: the perspective of realism, geometry consistency with the input sketches, and appearance consistency with the reference appearance images. The scores are obtained by averaging the received rankings for each method of each case on each criterion. For each invited user, we randomly select 5 cases from all the available cases to save his/her time. Thus, we collect $5 \times 3 = 15$ answers from each user. In total, 39 people (28 males and 11 females in the age of $18 - 40$ participated in this study. Therefore, $39 \times 15 = 585$ answers were collected.

Fig. **??** (a) plots the statistics of the evaluation results. We found the significant effects for all three criteria through one-way ANOVA tests: realism ($F_{(2,42)} = 80.33, p < 0.001$), geometry consistency ($F_{(2,42)} = 14.88, p < 0.001$), and appearance consistency ($F_{(2,42)} = 61.57, p < 0.001$). We also conduct paired t-tests to confirm the superiority in terms of geometry consistency of our method (mean: 2.00) to PixelNeRF (mean: 3.23; $[t = -6.42, p < 0.001]$), DFE (mean: 3.18; $[t = -6.53, p < 0.001]$), DFE-Projection (mean: 3.09; $[t = -6.13, p < 0.001]$), and pSp-Ref (mean: 3.48; $[t = -7.36, p < 0.001]$). Besides its superior performance in geometry consistency, our method is also rated as one of the best in terms of realism, and it (mean: 1.87) is more preferred than PixelNeRF (mean: 4.79; $[t = -17.46, p < 0.001]$), DFE (mean: 3.17; $[t = -6.21, p < 0.001]$), and pSp-Ref (mean: 3.07; $[t = -11.77, p < 0.001]$). In terms of appearance consistency, it (mean: 2.30) also outperforms PixelNeRF (mean: 4.42; $[t = -13.44, p < 0.001]$) and pSp-Ref (mean: 3.74; $[t = -7.05, p < 0.001]$). However, since DFE-Projection projects the results back into the latent space of our backbone, the realism of DFE-Projection (mean: 2.08; $[t = -1.00, p = 0.32]$) is comparable to ours, as expected. Due to the well-disentangled property of geometry and appearance of DFE, the appearance consistency of DFE (mean: 2.28; $[t = 0.11, p = 0.90]$) and DFE-Projection (mean: 2.24; $[t = 0.31, p = 0.75]$) are again comparable to ours.

As to the facial NeRF editing, we also compare our method against the same set of state-of-the-art methods in the qualitative comparison of sketch-based facial editing, i.e., DeepPS, DeepFaceEditing (denoted as "DFE"), and SketchEdit. We prepare 15 cases to cover varying personal attributes of the original identities, different editing regions (such as nose, hair, mouth, etc.), and different editing angles (frontal or tilted). Each case consists of an original facial image, a modified sketch where edited regions are emphasized, and the edited facial images generated by each method. Again, we only display the results from the viewpoints of the original facial images since not all the compared methods are 3D-aware. For each invited user, we also randomly select 5 cases from all the available cases and ask the user to rank the edited facial images (presented in a random order) from the perspective of realism, retention of the unchanged regions, and faithfulness to the edited sketches. Thus, we collect $5 \times 3 = 15$ answers from each user. In total, 39 people (28 males and 11 females in the age of $18 - 40$ with normal vision) without any special experience successfully participated in this study. Therefore we collected $39 \times 15 = 585$ answers in total.

Fig. **??** (b) plots the statistics of the evaluation results. We found the significant effects for all three criteria through one-way ANOVA tests: realism ($F_{(2,42)} = 70.49, p < 0.001$), retention ($F_{(2,42)} = 32.50, p < 0.001$), and faithfulness ($F_{(2,42)} = 37.71, p < 0.001$). We also conduct paired t-tests to confirm the superiority in all three

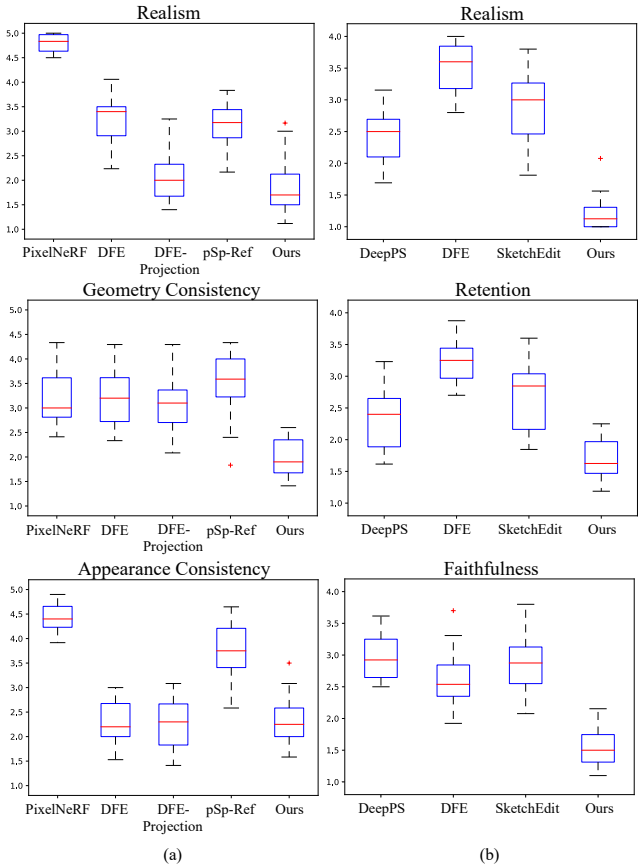


Fig. 14. Box plots of averaged perception rankings (the lower, the better). (a) The comparison of facial generation with five methods: PixelNeRF [?], DeepFaceEditing [?] (denoted as "DFE"), DeepFaceEditing-Projection (denoted as "DFE-Projection"), pSp [?] (denoted as "pSp-Ref"), and ours in terms of the realism, geometry consistency, and appearance consistency. (b) The comparison of facial editing with four methods: DeepPS [?], DeepFaceEditing [?], SketchEdit [?], and ours in terms of realism, retention, and faithfulness. The boxes are drawn from the first quartile to the third quartile, with a middle horizontal line denoting the median. The whiskers are the minimum and maximum values excluding any outliers.

criteria, i.e., realism, retention, and faithfulness in order, of our method (mean: 1.22, 1.71, 1.55) over DeepPS (mean: 2.41, 2.35, 2.96; $[t = -8.32, -4.33, -11.82, p < 0.001]$), DFE (mean: 3.49, 3.24, 2.64; $[t = -17.51, -12.69, -7.32, p < 0.001]$), and SketchEdit (mean: 2.86, 2.68, 2.83; $[t = -10.02, -5.76, -8.92, p < 0.001]$).

6 APPLICATION

In this section, we propose two novel applications of our system, namely, Semantic Facial NeRF Editing and Local Appearance Control.

Editing Propagation. As to the facial editing by sketches, since we carefully preserve our generative prior and achieve the editing by modifying the latent codes, our system can be further utilized to find editing directions for semantic controls. For example, as shown in Fig. **??**, we close the mouth in the first row and shorten the hair in the second row by editing the sketches. We subtract the latent codes for the original facial images from the derived latent codes for the edited facial images. Following this idea of latent vector arithmetic to GANs [?], the differences in the latent space are viewed as editing directions for closing the mouth and shortening the hair, and thus such editing directions can be applied to other smiling or long-hair cases. It is clear that the editing directions inferred by our system can generalize to other cases well, as seen from (d)~(g). Another interesting phenomenon is that by modifying the latent codes using our estimated semantic editing directions, not only are unedited regions well-preserved with small disturbances, such as those on the backgrounds but also the lighting effects are changed correspondingly. Such effects are more evident in the second case of shortening the hair: the right cheek becomes lit since the occlusion is removed by shortening the hair.

Local Appearance Control. Thanks to the 3D mask estimation and fusion strategy, our system can extend the appearance control by reference images from global space to local regions. Given the original 3D face, a sketch representing its facial geometry is rendered by the sketch generation approach. A new tri-plane representation that has the same geometry as the original face but has a new appearance is generated by the *Sketch Tri-plane Prediction* net in Sec. **??**. The original and new tri-plane features are fused by 3D masks estimated from the 2D masks M (drawn by users), similar to the function of the *Mask Fusion* module. The fused tri-plane feature is expected to preserve the original geometry of the entire face and appearance on untouched regions but change the appearance of the drawn regions, as shown in Fig. **??**. During the optimization, we further utilize a histogram loss to maintain the appearance faithfulness:

$$\mathcal{L} = \mathcal{L}_{hist}(\mathrm{H}(I' \odot M), \mathrm{H}(I_{ref} \odot M)) + \mathcal{L}_{hist}(\mathrm{H}(I' \odot \widetilde{M}), \mathrm{H}(I_{geo} \odot \widetilde{M})), \tag{12}$$

where H denotes the histogram features and $\mathcal{L}_{hist}$ denotes the feature distance as in [?]. I_{geo}, I_{ref}, and I' represent the original image, appearance reference image, and generated image, respectively, and $\widetilde{M}$ refers to unedited regions. As shown in Fig. **??**, local region appearance like hair and skin is modified effectively, while the features in other regions are retained. Since the fusion is conducted in the 3D space, our results can further be rotated to other viewpoints with 3D consistency.

7 CONCLUSIONS AND DISCUSSIONS

In this paper, we have presented the first novel sketch-based facial NeRF generation and editing method. The *Sketch Tri-plane Prediction* net is designed to supplement the appearance and stereoscopic information into 2D sketches, combined with a pretrained generator to synthesize high-quality faces NeRFs. Our system is robust against diverse drawing styles and allows appearance control. To preserve unedited 3D regions during local editing, we further propose the Mask Fusion module and a latent code optimization from sketch strategy, which can be performed repeatedly to support 3D-aware multi-step manipulations from different viewpoints. Our approach

Fig. 15. Results of Editing Propagation. In the first three columns (from left to right), we show the input images, edited sketches, and edited results. Mouth closing and hair cutting are applied by users. It can be seen that similar editing effects can be propagated to other persons.

Fig. 16. Results of local appearance control. Given faces rendered in certain views (a), users draw masks to indicate local editing regions (b). As shown in (c) and (d), with the reference images on the top-left corner, the local appearance in hair and skin regions is changed while the features in unedited regions are maintained.

Fig. 17. Failure cases. When the hand-drawn sketches (a) are too abstract and cartoonish, the generated faces (b) are still of good quality but cannot capture overly exaggerated characteristics. Our approach also cannot handle uncommon personal accessories such as the hat in (e).

outperforms existing sketch-based facial generation and editing approaches not only on faithfulness, and visual quality but also on 3D view consistency. We also adapted our system for two applications: semantic facial NeRF editing and local appearance control.

Thanks to the *Sketch Tri-plane Prediction* net and the pretrained generator, our system is robust for hand-drawn sketches. However, as shown in Fig. **??**, when users draw too abstract or cartoonish sketches, generated 3D faces might fail to capture overly exaggerated characteristics, though they are still of good quality. Besides, our system is designed to generate 3D faces from scratch with the input of front-view sketches because it is hard for novice users to draw free-view facial sketches, and the camera parameter prediction is very challenging. Designing a specific method to detect camera poses from hand-drawn sketches can partly solve this problem. Moreover, as shown in Fig. **??**, our system cannot deal with uncommon personal accessories such as hats since these examples are rare in our training dataset. Designing a specific approach to solving the data imbalance or augmenting the training data could alleviate this problem.

Ethical Discussion. Our work originates from and benefits positive real-world applications, such as digital character design, virtual meetings, and entertainment. However, the facial image generation and editing works have long suffered from potential harmful abuses. To prevent misuse, many works [**???**] in the fake detection community could discriminate between the synthesized and real faces. Besides, the generated free-view facial images of our method can also be utilized as a training dataset to benefit the fake detection research.

ACKNOWLEDGMENTS

Thanks to Ms. Xiaohong Wang for her valuable comments and suggestions on the use of OneITLab. This work was supported by grants from the National Natural Science Foundation of China (No. 62061136007 and No. 62102403), the Beijing Municipal Natural Science Foundation for Distinguished Young Scholars (No. JQ21013), China Postdoctoral Science Foundation (No. 2022M713205), the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CityU 11212119), Chow Sang Sang Group Research Fund (No. 9229119), and the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, CityU.