DOI: 10.1111/cgf.14890
Eurographics Symposium on Rendering 2023
T. Ritschel and A. Weidlich
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 4
e14890 *pp. 1 - 13*

PVP: Personalized Video Prior for Editable Dynamic Portraits using StyleGAN

K.-E. Lin[1], A. Trevithick[1], K. Cheng[2], M. Sarkis[2], M. Ghafoorian[2], N. Bi[2], G. Reitmayr[2], R. Ramamoorthi[1]

[1]University of California, San Diego [2]Qualcomm Technologies Inc.

Figure 1: ***Portrait reenactment and editing with our proposed method.*** *Our method takes a monocular video and learns a personalized video prior, which allows accurate reeanctment and editing through StyleGAN. In the first row, we show the selected pivots used to fine-tune the StyleGAN generator. We show different editing results in the second row: original, eye glasses, small eyes, chubby, fringe hair. The reenacted sequence is driven by the pose and expression of the third row.*

Abstract

Portrait synthesis creates realistic digital avatars which enable users to interact with others in a compelling way. Recent advances in StyleGAN and its extensions have shown promising results in synthesizing photorealistic and accurate reconstruction of human faces. However, previous methods often focus on frontal face synthesis and most methods are not able to handle large head rotations due to the training data distribution of StyleGAN. In this work, our goal is to take as input a monocular video of a face, and create an editable dynamic portrait able to handle extreme head poses. The user can create novel viewpoints, edit the appearance, and animate the face. Our method utilizes pivotal tuning inversion (PTI) to learn a personalized video prior from a monocular video sequence. Then we can input pose and expression coefficients to MLPs and manipulate the latent vectors to synthesize different viewpoints and expressions of the subject. We also propose novel loss functions to further disentangle pose and expression in the latent space. Our algorithm shows much better performance over previous approaches on monocular video datasets, and it is also capable of running in real-time at 54 FPS on an RTX 3080.

CCS Concepts

• ***Computing methodologies*** → ***Image-based rendering;***

1. Introduction

Digital avatars offer a compelling way to represent human appearance and expression, enabling various applications, like telepresence and virtual reality. Recent advances in StyleGAN [KLA19, KLA*20] and neural radiance fields (NeRFs) [MST*20] have accelerated the development of digital avatars [HPX*22, AXS*22]. Radiance fields have also spawned many extensions [HPX*22, AXS*22, ZLW*22, GZX*22, XWZ*22] which capture various characteristics of human portraits, such as expression, appearance and identity.

However, the NeRF-based methods mostly focus on reconstruct-

Table 1: ***Comparison of different methods.*** *Our proposed method bridges the gap between 2D and 3D methods by enabling reconstruction on extreme viewpoints through personalization. 2D methods allow for efficient image synthesis from single view inputs, but they often cannot handle larger motions. Even if the input video contains large motion changes, 2D methods [WYBD22, YZC*22] are not able to use the information and reconstruct more difficult viewpoints. On the other hand, 3D methods [GPL*22, GZX*22] can handle difficult viewpoints and NeRFBlendShape can achieve some form of personalization in expression. However, they do not allow full editability on appearance like the 2D methods. Our method achieves editable head poses, appearance editing via StyleGAN, monocular input, riggability through FLAME-like controls, and personalization with the help of a personalized video prior.*

Category	Methods	Head pose editing	Appearance editing	Monocular input	Riggable	Personalization
NeRF	HeadNeRF [HPX*22]	✗	✓	✗	✗	✗
	RigNeRF [AXS*22]	✓	✗	✓	✓	✓
	FDNeRF [ZLW*22]	✓	✗	✓	✓	✗
	cGOF [SWH*22]	✗	✓	✓	✓	✗
	NeRFBlendShape [GZX*22]	✓	✗	✓	✓	✓
	NeRFFaceEditing [JCL*22]	✗	✓	✓	✗	✓
Parametric Model	NHA [GPL*22]	✓	✗	✓	✓	✓
	ROME [KSLZ22]	✗	✗	✓	✓	✗
StyleGAN	FreeStyleGAN [LD21]	✗	✓	✗	✗	✗
	PIRenderer [RLC*21]	✓	✗	✓	✓	✗
	LIA [WYBD22]	✓	✗	✓	✓	✗
	StyleHEAT [YZC*22]	✓	✓	✓	✓	✗
	Ours	✓	✓	✓	✓	✓

ing the input sequence and do not provide a good way to edit or manipulate various properties. Many of these approaches rely on learning a 3D generative prior, thus requiring inversion into the latent space, which requires generator finetuning and an inevitable loss of 3D representation ability. These methods are also non-trivial to extend to video sequences of a subject. We argue that editability also plays an important part in making the digital avatars engaging, and it offers the user more control over their desired looks (see Fig. 1). A recent paper, FreeStyleGAN [LD21], enables editing on static human portraits, but it requires multiple carefully-shot images with the subject holding still for several seconds. Concurrent work [SLHG22,JCL*22] provides an editing interface to allow user inputs, but these methods are not animatable (e.g. enabling 3DMM-like control [BV99]) and thus do not offer the same level of interactivity. Lastly, while ClipFace [ATDN22] provides editability on 3D morphable models (3DMM) and their texture, it does not handle other facial features like hair.

Our proposed method offers a new way to tackle portrait synthesis using personalized StyleGAN from monocular portrait videos. First, we carefully sample several frames from the input videos to use as pivots, and the pivots are used to perform PTI [RMBCO21] and fine-tune the StyleGAN generator to produce a personalized manifold, allowing for faithful reconstruction of the subject under extreme poses and smooth transitions between different head poses. Then, we employ lightweight pose and expression encoders to enable finer control over the representation. We can render novel head poses and expressions in real-time (at 54 FPS) by feeding in the corresponding pitch, yaw angles and FLAME coefficients [LBB*17]. We also propose a novel expression matching loss and pose consistency loss to disentangle the editing directions in the latent space, making it easier to change one attribute at a time. Additionally, exploiting various methods in StyleGAN editing [PWS*21,SYTZ20], our method can provide good editing capability. Project website and code: https://cseweb.ucsd.edu/~viscomp/projects/EGSR23PVP/

We summarize our contributions as follows:

1. a personalized video prior derived from monocular portrait video of a given subject (Sec. 4.1);
2. a novel algorithm that enables control of a dynamic portrait within the personalized manifold, allowing editing on pose, expression and appearance (Sec. 4.2, Fig. 2);
3. expression matching and pose consistency loss to better disentangle the poses and expressions given only a short portrait video of a subject (Sec. 4.3).

2. Related Work

Previous methods have shown promising results in reconstructing photorealistic face renderings. In particular, there exist 4 promising directions: (a) StyleGAN models which can synthesize 2D face renderings and allow for user control by traversing in their latent space (Sec. 2.1); (b) neural radiance fields (NeRF) that handle the dynamic facial expression and complex visual effects through volumetric rendering (Sec. 2.2); (c) parametric models, like 3DMM and FLAME, which provide explicit pose and expression control through skinning (Sec. 2.3). (d) facial reenactment methods which enable facial expressions to be transferred to different subjects (Sec. 2.4). We give an overview of these methods in the following subsections and a comparison of our method against previous methods in Table 1.

2.1. Face Synthesis with StyleGAN

Generative models like StyleGAN2 [KLA*20] have demonstrated an impressive capability of synthesizing photorealistic por-

traits while only trained on in-the-wild imagery. In order to reconstruct human faces, a common method is to invert the underlying latent code. GAN inversion methods, like e4e [TAN*21] and pSp [RAP*21], have shown remarkable results in finding the most representative latent code. With the latent codes, PTI [RMBCO21] and MyStyle [NAH*22] further optimize the StyleGAN generator to learn a personalized latent space, enabling editing while staying faithful to the same identity. In terms of controlling the pose and expression of the StyleGAN renderings, StyleRig [TEB*20] predicts modified latent codes directly from semantic control parameters. StyleHEAT [YZC*22] performs warping on the intermediate features layers of the StyleGAN synthesis network. Although these methods show promising results, there are some shortcomings. For example, StyleRig could introduce a shift in identity after rotating the head pose. The warping scheme used by StyleHEAT could not handle large viewpoint changes well enough (see Fig. 5). The aforementioned StyleGAN methods are mainly handling 2D images.

While 3D GAN-based methods [CMK*21, CLC*22, GLWT22, OELS*22, DYXT22] generate photorealistic 3D representations from 2D images, the editability of such models requires further research to reach its full potential. SofGAN [CLX*22], IDE-3D [SWS*22], FENeRF [SWZ*22] and CIPS-3D [ZXNT21] demonstrated some level of editability on the appearance and expression of 3D GANs. Additionally, inverting a video into their latent space is non-trivial as each frame is inverted independently, and 3D GAN PTI often collapses to a 2D representation.

As a result, our paper looks at lifting 2D StyleGAN renderings to a 3D representation that allows editing of both the facial appearance and expression. One related work is FreeStyleGAN [LD21]. However, it requires the subject to be static and multi-view input images. Our method seeks to handle dynamic portraits with only a single-view input video. Please refer to Table 1 for a comparison of different methods.

2.2. 3D Methods for Dynamic Portraits

NeRF [MST*20] brought about exciting advancements in the field of image-based rendering. Recently, a plethora of work tackling portrait reconstruction [HPX*22, AXS*22, ZLW*22, GZX*22, SLB*21, ZAB*22, PSB*21, PSH*21] has emerged. First, NeRFace [GTZN21] conditions the MLP with learnable codes and expression coefficients to represent dynamic facial motions. HeadNeRF [HPX*22] encodes identity, expression, albedo and illumination into latent codes and uses them to condition the NeRF MLP, providing some level of user control. RigNeRF [AXS*22] combines the 3DMM deformation fields with NeRF to allow explicit control of the expression and head poses. Nerfies [PSB*21] and HyperNeRF [PSH*21] provide ways to encode deformable NeRFs, enabling capture of 3D facial movements. Following the above methods, FDNeRF [ZLW*22] further extends to few-shot inputs and NeRFBlendShape [GZX*22] blends hashgrids with different expression coefficients to allow for fast and expressive portrait reconstruction. While the above approaches mostly focus on reenactment and pose manipulation, they do not provide comprehensive appearance editing.

On the other hand, NeRFFaceEditing [JCL*22] focuses on the editability of different facial parts. SofGAN [CLX*22] provides finer control over different facial parts, including appearance, shape and lighting effects. However, these methods do not offer explicit facial pose control. Our work aims to achieve granular pose control and appearance editing at the same time.

2.3. Parametric Model for Human Faces

There has been a lot of interest in using morphable face models to enable facial animation [TL18, GVR*14, WBLP11]. The FLAME model [LBB*17] offers a good trade-off between controllability and expressiveness. Derivatives like DECA [FFBB21] can infer FLAME parameters given only a single image. ROME [KSLZ22] further expands DECA to include displacements for the head mesh and preserve more details in non-facial regions (e.g. hair, neck and shoulders). A notable work in this category is NHA [GPL*22], which optimizes a head model given a single-view video. NHA provides a good geometry estimate of the subject while keeping explicit control over the FLAME parameters. Although NHA supports editing the facial expression and head pose through the FLAME parameters, it does not have an interface for appearance editing. Our proposed method focuses on this aspect and seeks to combine the editability of StyleGAN images with FLAME-like control parameters.

2.4. Facial Reenactment Techniques

In addition to the above methods, there are some other techniques that focus on the facial reenactment tasks. In other words, these methods transfer the expressions from one subject to another. Face2Face [TZS*16] achieves real-time face capture and reenactment by fitting a parametric 3D model to monocular RGB input videos. Deep Video Portraits [KGT*18] utilizes a translation network to convert coarse facial renderings into photorealistic video portraits. Different from these methods, our work enables reenactment and editing simultaneously through the use of pose and expression encoder networks and the StyleGAN latent space.

3. Background

In this section, we give a brief overview on StyleGAN and its latent space design. Then, as we seek to edit real portraits, we introduce pivotal tuning inversion (PTI), which is the state-of-the-art method to perform GAN inversion, while keeping good editability. PTI serves as an important foundation for our method, since we would like to explore the idea of a personalized video prior by fine-tuning the StyleGAN generator on selected frames of a given video (more in Sec. 4). Lastly, we briefly go through editing on the StyleGAN.

StyleGAN [KLA*20] demonstrates high-quality image synthesis results, and it has a well-structured latent space, which allows smooth transition between different latent codes through linear interpolation. To be more specific, it takes a latent code $z \in \mathbb{R}^{512}$ as input (often referred to as $\mathcal{Z}$ space) to a mapper network, which maps the input to the intermediate latent code $w \in \mathcal{W}$. The latent

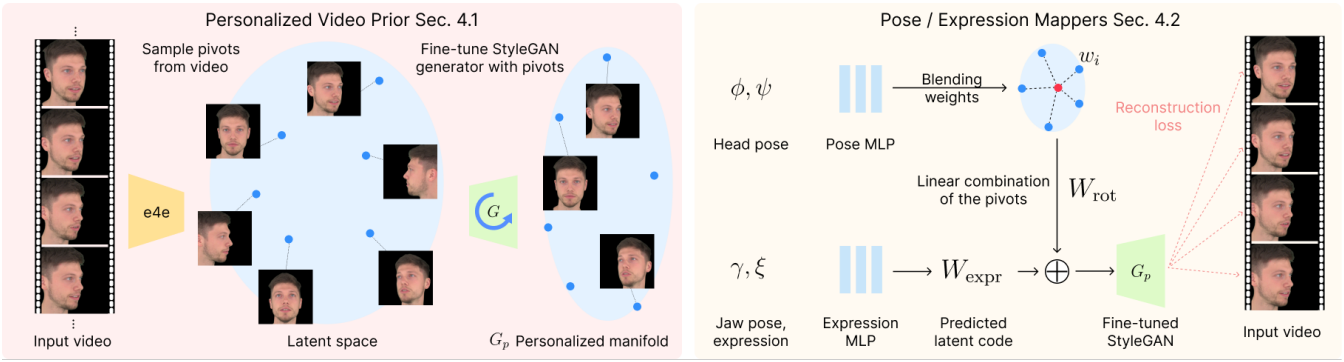

Figure 2: ***Overview of our proposed method.*** *Our algorithm is divided into two stages. First, we use PTI to fine-tune the StyleGAN generator with the selected frames from the input video (see Sec. 4.1). The fine-tuned StyleGAN now has a personalized manifold that reconstructs the pivots faithfully, and we utilize its smoothness to interpolate between the pivots and represent different head poses. Then, we train the pose MLP to output the blending weights of the personalized manifold to provide a latent code that represents the rotated head. We use the expression MLP to calculate the W+ latent code residuals (see Sec. 4.2). Finally, the combined latent code is sent to the fine-tuned StyleGAN to synthesize the final rendering, and we supervise it with reconstruction loss and other supervision (see Sec. 4.3).*

code w is then affine transformed and fed to each convolutional layer in the synthesis network via adaptive instance normalization (AdaIN) [HB17]. The affine transformed latent code is often referred to as the $\mathcal{W}+$ space. We can then write the StyleGAN image synthesis as $I' = G(w^+(z);\theta)$, for a StyleGAN generator G with weights θ.

A way to adapt StyleGAN to editing real images is through GAN inversion. The idea is to freeze the StyleGAN weights and optimize for the latent code in either $\mathcal{W}$ [KLA*20], or $\mathcal{W}+$ [AQW19]. However, there is a tradeoff between the two methods. $\mathcal{W}$ space inversion offers better editability but poor reconstruction quality, whereas $\mathcal{W}+$ provides superior reconstruction, yet produces inferior editing results. To address this tradeoff, the better way is to find pivot latent codes in the $\mathcal{W}+$ space via e4e [TAN*21] and then fine-tune the StyleGAN generator based on the pivots, namely pivotal tuning inversion. With PTI, it is possible to extend StyleGAN to handle samples which differ greatly from the training distribution, for instance, persons with makeup and faces viewed at large angles ($\geq 60°$). To be more specific, we can define the pivots as

$$w^+ = E(I), \tag{1}$$

where E is the e4e encoder, I the input image, and w^+ the inverted latent code in $\mathcal{W}+$ space. And we can synthesize the rendering with the StyleGAN G by

$$I' = G(w^+;\theta), \tag{2}$$

where θ denotes the weights of a StyleGAN generator. Once we acquire the pivots, we can then optimize the StyleGAN generator G and obtain the fine-tuned weights θ_p with the following objective:

$$\theta_p = \arg\min_{\theta} \mathcal{L}_{\text{LPIPS}}(I', I) + \lambda^{\text{P}}_{L2}\mathcal{L}_{L2}(I', I) + \lambda^{\text{P}}_{R}\mathcal{L}_{R}, \tag{3}$$

where $\mathcal{L}_{\text{LPIPS}}$ is the perceptual loss [ZIE*18], $\mathcal{L}_{L2}$ denotes the MSE loss, λ^{P}_{L2} denotes the weight for MSE the loss, $\mathcal{L}_R$ is the locality regularization loss [RMBCO21], and λ^{P}_{R} the weight for the regularizer. The p superscript denotes personalization, since we use $L2$ loss later for training our mapping networks as well. Aside from the LPIPS and MSE loss, the locality regularization is enforced to make sure the PTI changes stay local without affecting other parts in the latent space. Once, the network is fine-tuned, it can be seen as a personalized generator which now incorporates a prior based on the given subject, instead of the domain prior it originally has [NAH*22].

Finally, we can edit an inverted image by manipulating the latent code, which can be described as:

$$I'_{\text{edit}} = G(w^+ + \Delta w^+;\theta_p), \tag{4}$$

where Δw^+ could be any editing directions from InterfaceGAN [SYTZ20] or GANSpace [HHLP20].

In the next section, we discuss our proposed algorithm, starting by sampling frames from the input video, performing PTI on the selected frames to learn a personalized video prior, and then learning pose and expression mapper networks to represent different head pose and facial expressions on the personalized manifold.

4. Proposed Method

Given a monocular video of a dynamic portrait, we seek to produce an editable representation with fine controls over the pose, expression and appearance of the subject. To achieve this goal, we develop an optimization pipeline with two stages: (a) learning the personalized video prior in StyleGAN, and (b) training pose and expression mapper networks. To start, we discuss the first stage in Sec. 4.1. Then, we describe the second stage, namely, how we train the pose and expression mappers, and how we use them to control different head poses and expressions in Sec. 4.2. Last, we describe our loss design for the mappers in Sec. 4.3. Please refer to Fig. 2 for an overview of our optimization pipeline.

4.1. Personalized Video Prior in StyleGAN

Our main goal is to generate an edited portrait video with StyleGAN2 [KLA*20]. A naive way to address this is to perform GAN

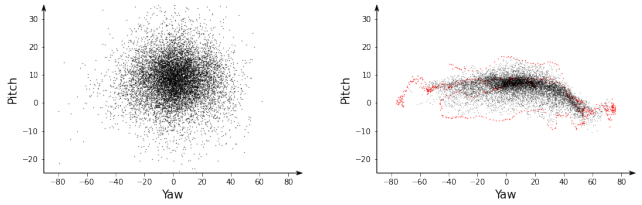

Figure 3: ***Head pose distribution of vanilla StyleGAN2 and personalized manifold.*** *We randomly synthesize 10k facial samples with the truncation parameter set to 0.5. Then we use DECA [FFBB21] to detect the yaw and pitch of the head. Most head poses have yaw inside* $[-60, 60]$ *and pitch in* $[-20, 30]$*, making it difficult to invert to head pose outside of this distribution. In the right figure, we randomly sample a linear combination of the pivots and synthesize novel views with StyleGAN. We highlight the head poses from the input video in red and the samples from the personalized manifold in black. The learned personalized manifold can reconstruct extreme yaw poses (*$\geq 60°$*) in the input video.*

inversion on a frame-by-frame basis. However, previous methods generally struggle with extreme viewing angles ($\sim 90°$). This is because large head rotations ($> 40°$ in yaw, $> 25°$ in pitch) are less represented in the StyleGAN latent space, as shown in the left diagram in Fig. 3. We notice that these out-of-domain (OOD) samples can be better represented with PTI [RMBCO21] as discussed in Sec. 3. Another property we discovered is that interpolation between the pivots offers a smooth transition between different head poses (see the right diagram of Fig. 3), and we can represent the head motions as a linear combination of different pivot vectors, similar to the notion of linear blending in animation. As a result, our idea is to seek out these pivots in the given portrait video, and construct a manifold to allow animating the portrait by user control.

As discussed, when presented an image sequence, our task is to seek a personalized video prior p that forms a local manifold W_p within the latent space of the fine-tuned StyleGAN $G(\cdot;\theta_p)$. First, we preprocess the images by aligning and cropping the face following FFHQ [KLA19]. As the facial landmark detection could introduce slight inconsistencies and cause the results to flicker, we employ Gaussian filters to smooth out the noise [FTET21, TMG*22]. Since there are many frames in a video, we seek to select the most useful frames as pivots to ensure efficient learning of the personalized manifold. We design a sampling strategy to uniformly sample different head poses and expressions throughout the video to ensure good coverage of all possible facial changes. To be more specific, we utilize an off-the-shelf face detector, DECA [FFBB21], to estimate the yaw ψ, pitch ϕ, neck pose κ, jaw pose γ and expression parameters ξ of a given sequence. We then stack the ψ, ϕ and ξ channel-wise and perform K-means clustering [Llo82] to select the most prominent K clusters. While uniform sampling sometimes yields similar results, our sampling strategy would avoid oversampling cases where the pose or expression stay similar. We find that this strategy provides good coverage of different expressions and head poses in the input video. We define the K samples from the video as $I = \{I_1, I_2, ..., I_K\}$. Finally, we optimize the StyleGAN generator, similar to Eq. 3, with the following objective:

$$\theta_p = \arg\min_{\theta} \sum_{i=1}^{K} \frac{1}{K} \left[\mathcal{L}_{\text{LPIPS}}(I'_i, I_i) + \lambda^{\text{P}}_{L2} \mathcal{L}_{L2}(I'_i, I_i) \right] + \lambda^{\text{P}}_{R} \mathcal{L}_R. \quad (5)$$

The main difference from Eq. 3 is that we are optimizing over the K images instead of only one image. From this point forward, we introduce a shorthand $G_p = G(w^+;\theta_p)$. Once we acquire the fine-tuned StyleGAN G_p, we can define the β-dilated convex hull [NAH*22] in the $\mathcal{W}+$ space as the local manifold W_p:

$$W_p = \sum_{i=1}^{K} \alpha_i w_i, \qquad \text{s.t.} \sum_{i=1}^{K} \alpha_i = 1, \text{ and } \alpha_i \geq -\beta, \quad (6)$$

where α_i is the blending weight for pivot w_i. As discussed in Sec. 3, interpolation between StyleGAN latent codes provides smooth transitions between different pivots. We also notice that interpolating between poses at $90°$ and $-90°$ in W_p gives stable and high-quality renderings of the poses in between, while maintaining the same identity of the subject.

Discussion. Our proposed fine-tuning stage has two differences from MyStyle [NAH*22]. First, MyStyle uses 100 to 200 images of the given subject under different settings, including lighting, ages, expressions and hairstyles, whereas our method focuses on frames from the same input video where the subject can have similar appearance. This difference make redundant samples and overfitting more likely to happen due to less variety in the training samples. However, our proposed sampling strategy ensures the samples are different enough in poses or expressions by clustering and selecting the representative frames from the input video. This way, our method can represent a wide variety of facial expressions and poses given a short video of around 20 seconds. Another difference is that fine-tuning the StyleGAN without any regularizer $\mathcal{L}_R$ could lead to degraded editability (see Fig. 9). We theorize that this is because the fine-tuned StyleGAN overfits to the input video and causes other parts in the latent space to lose the smoothness of the original StyleGAN. To prevent this, it is important to apply the locality constraint in our setting to keep the structure in the StyleGAN latent space relatively unchanged after the fine-tuning. It is also beneficial to use StyleGAN's expressive latent space to synthesize expressions which are not observed in the input video. We design a loss function that utilizes random expression objectives to encourage the network to synthesize these unseen expressions. More details are in Sec. 4.2 and Sec. 4.3.

4.2. Pose and Expression Mapping Networks

A key contribution of our proposed method is the ability to easily control the face with parameters like yaw, pitch and a 50-dimensional expression vector from DECA [FFBB21]. To this end, we implement the pose and expression mappers, which are MLPs taking pose and expression coefficients as input and producing the corresponding latent code (see Fig. 4 for examples).

First, in order to change the head pose of the rendering, we could manipulate the latent code by moving in the personalized manifold W_p. Specifically, we employ an MLP $\mathcal{F}_{\text{rot}}$ to calculate the latent

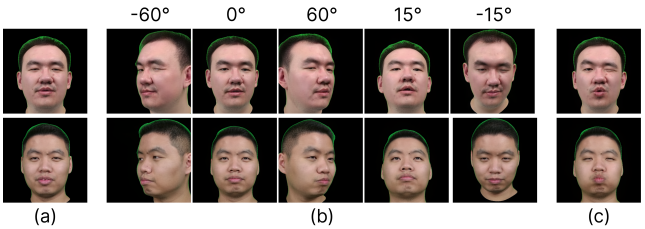

Figure 4: ***Examples of latent code shown in Sec. 4.2.*** *(a) The personalized manifold W_p provides a good representation of the subject, and it allows us to further edit the pose and expression. (b) The rotation code w_{rot} enables free pose control of the synthesized portraits. Here we show specific yaw ($-60°$ to $60°$) and pitch ($-15°$ to $15°$) controls. (c) The expression code w_{expr} allows finer control over the facial expressions.*

code w_{rot} as a linear combination of the pivots $\{w_i\}_{i=1}^K$ by

$$w_{\text{rot}} = \sum_{i=1}^{K} \alpha_i^{\text{rot}} w_i, \quad \text{and} \quad \{\alpha_i^{\text{rot}}\}_{i=1}^K = \mathcal{F}_{\text{rot}}(\phi, \psi; \theta_{\text{r}}), \tag{7}$$

where ϕ denotes the pitch, ψ the yaw of the head and θ_{r} represents the weights of $\mathcal{F}_{\text{rot}}$. We use the MLP to predict the blending weights instead of the latent code in $\mathcal{W}$ space or $\mathcal{W}+$ space, because the blending weights effectively constrain the latent code to be inside the manifold where all points are meaningful and would represent the subject (see the video results in supplementary materials for an example). On the other hand, it is possible for editing directions in $\mathcal{W}$ and $\mathcal{W}+$ space to go beyond the personalized manifold and degrade the rendering significantly.

Lastly, we introduce expression control by learning a mapping network that takes as input the jaw pose γ and the expression parameters ξ and outputs the difference of the latent code Δw_{expr} in $\mathcal{W}+$ space. Specifically, we have

$$\Delta w_{\text{expr}} = \mathcal{F}_{\text{expr}}(\gamma, \xi; \theta_{\text{e}}). \tag{8}$$

We then add it to the latent code w_{rot}:

$$w_{\text{final}} = w_{\text{rot}} + \Delta w_{\text{expr}}. \tag{9}$$

In practice, we only predict the first 8 layers of the latent code instead of the full 18 layers. This is because we would like to focus on geometry changes instead of other appearance changes. We are effectively predicting a local change in the latent space to represent changes like lip motions during a talking sequence and facial expressions like raising of the eyebrows. The local neighborhood of the latent space contains a rich and disentangled representation of the subject's possible facial expressions. Note that it is possible to move the latent code slightly outside the convex hull. To regularize, we only use the first 8 layers of $\mathcal{W}+$ to reduce degrees-of-freedom, and we apply $\mathcal{L}_{\text{local}}$ (see Sec. 4.3) to keep the changes small.

Finally, we can then render the novel view I' with the fine-tuned StyleGAN generator G_p:

$$I' = G_p(w_{\text{final}}). \tag{10}$$

Note that once the mapping network is trained, it can be run in real-time given the pose and expression parameters.

4.3. Loss Design for Mapping Networks

Next, we fix the StyleGAN generator weight θ_p, and train the mapping networks $\mathcal{F}_{\text{rot}}$ and $\mathcal{F}_{\text{expr}}$ with the loss design discussed in this section, optimizing for their weights θ_r and θ_e. As we aim to reconstruct an input video, we supervise the output with rendering losses like LPIPS [ZIE*18] and $L2$ loss, which are denoted by $\mathcal{L}_{\text{LPIPS}}$ and $\mathcal{L}_{L2}$, respectively. Additionally, to ensure the synthesized identities are the same, we apply an identity loss $\mathcal{L}_{\text{id}}$ [TAN*21], which uses a pretrained ArcFace [DGXZ19] network to obtain identity features.

Since the input video would often show one combination of the head pose and expression, it is insufficient to train on the input video alone. To avoid overfitting, we introduce regularization by synthesizing images with a slightly perturbed expression input. Precisely, for each target view I and its corresponding jaw pose γ and expression ξ, we render an additional rendering I^{e} with G_p by changing the input to the perturbed parameters $\gamma' = \gamma + \epsilon, \xi' = \xi + \epsilon$, where we add a normal distribution $\epsilon \sim \mathcal{N}(0, \sigma)$. Then we feed the new rendering I^{e} to the encoder of DECA [FFBB21] to predict yaw ψ^e, pitch ϕ^e, jaw pose γ^e, expression ξ^e, and neck pose κ^e. and define the expression matching loss as:

$$\mathcal{L}_{\text{expr}} = \|(\gamma^e, \xi^e) - (\gamma', \xi')\|_2^2. \tag{11}$$

We found this loss to improve the disentanglement of expression and head pose, since the synthesized portrait has the same head pose but a slightly different facial expression. This additional objective increases the diversity of facial expressions that the expression mapper can generate. Visualization of the perturbed renderings is shown in Appendix A. Furthermore, we enforce a pose consistency loss and an RGB consistency loss to make sure I^{e} does not show large head motions. We define the pose consistency loss as:

$$\mathcal{L}_{\text{pose}} = \|\kappa^e - \kappa\|_2^2, \tag{12}$$

Similarly, κ is the neck pose for the input frame, and κ^e denotes the DECA-estimated neck pose of the perturbed rendering. Note that we do not feed the neck pose κ as an input to the pose MLP. For the RGB consistency loss, we calculate a mask m excluding the eyes and mouth regions, and then minimize the loss between the perturbed rendering I^{e} and the original rendering I':

$$\mathcal{L}_{\text{cons}} = \|m \odot (I^{\text{e}} - I')\|_2^2 \tag{13}$$

To further encourage the expression network to explore local regions in the manifold, we apply a regularization loss on the predicted latent Δw_{expr}, namely,

$$\mathcal{L}_{\text{local}} = \|\Delta w_{\text{expr}}\|_2^2. \tag{14}$$

Lastly, our training objective is defined as follows:

$$\begin{aligned}(\theta_{\text{r}}, \theta_{\text{e}}) = \arg\min_{\theta_{\text{r}}, \theta_{\text{e}}} & \lambda_{\text{LPIPS}} \mathcal{L}_{\text{LPIPS}} + \lambda_{L2} \mathcal{L}_{L2} + \lambda_{\text{id}} \mathcal{L}_{\text{id}} \\ & + \lambda_{\text{cons}} \mathcal{L}_{\text{cons}} + \lambda_{\text{pose}} \mathcal{L}_{\text{pose}} + \lambda_{\text{expr}} \mathcal{L}_{\text{expr}} + \lambda_{\text{local}} \mathcal{L}_{\text{local}},\end{aligned} \tag{15}$$

where we update the pose network weights θ_{r} and the expression network weights θ_{e}.

5. Experiments

We demonstrate the efficacy of our proposed method against SOTA 2D methods [RLC*21, WYBD22, YZC*22] and a 3D

Figure 5: ***Visual results of different methods.*** *We show reconstruction results on the held-out views. Our method can predict better geometry and appearance of the subject. For example, in column (a), (b) and (c), we can see the ear reconstruction failed for NHA [GPL*22]. For 2D methods, they mostly fail to reconstruct the correct details and expression. To be more specific, PIRenderer [RLC*21] and StyleHEAT [YZC*22] both fail for large viewpoint changes, with StyleHEAT generating overly-smoothed results, as shown by the red arrow in column (c). LIA [WYBD22] overall has a tone shift and fails to reconstruct the details correctly, like the earring shown in (d). Our method offers faithful reconstruction of the head pose and expressions of the subject, as well as details, like the tint on the eyeglasses in (f).*

Table 2: ***Evaluation of ours and baseline methods.*** *Our renderings offer the best visual quality across all metrics. 2D-based methods struggle to handle viewpoints at larger angles, resulting in poor visual quality. While 3D-based methods like NHA can handle larger head motions, they fail to reconstruct good 3D geometry for in-the-wild videos, leading to inferior results across all metrics.*

Methods	PSNR↑	LPIPS↓	SSIM↑
PIRenderer	21.49	0.2431	0.7540
LIA	23.47	0.1796	0.8545
StyleHEAT	20.29	0.2494	0.7685
NHA	25.31	0.1538	0.8746
Ours	**26.63**	**0.1119**	**0.8803**

method [GPL*22] in this section. We first describe implementation details in Sec. 5.1. We show quantitative comparison in Sec. 5.2 and Table 2. Then we provide visual results in Sec. 5.3, **Figs.** 5 to 8. Finally, in Sec. 5.4, we discuss limitations and possible future directions.

5.1. Implementation Details

We implement our algorithm using PyTorch [PGM*19]. We first run PTI [NAH*22] with the LPIPS threshold set to 0.03, and max PTI steps set to 350, on the samples from the input video to acquire a personalized manifold. Then we train our pose and expression mappers for 50k steps. We set our learning rate for the MLPs to be 5×10^{-4}. For the loss functions in Sec. 4.3, we use $\lambda_{\text{LPIPS}} = 10$, $\lambda_{L2} = 10$, $\lambda_{\text{id}} = 0.5$, $\lambda_{\text{pose}} = 0.1$, $w_{\text{expr}} = 0.1$, $\lambda_{\text{cons}} = 1.0$, and $\lambda_{\text{local}} = 0.5$. We set $\sigma = 0.5$ for the perturbed expression parameters. Further details can be found in Appendix B.

For the dataset, we use the monocular video dataset from NHA [GPL*22] and NeRFBlendShape [GZX*22]. For the NHA dataset, we choose the first 750 frames as training data. Due to the last few frames being corrupted in the original video data provided by the author, we choose frames 751 to 1450 as the evaluation dataset instead of frames 751 to 1500. The NeRFBlendShape dataset has videos with 3000 to 4000 frames, and we choose the last 500 frames as evaluation data. We briefly describe dataset composition in Appendix C. Source code can be found on our website: https://cseweb.ucsd.edu/~viscomp/projects/EGSR23PVP/.

5.2. Quantitative Results

To evaluate the visual quality, we used metrics like peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and perceptual loss (LPIPS). We evaluate all methods on the held-out views from the evaluation dataset discussed in Sec. 5.1. As for the baseline methods, we evaluate the resolution at 512×512 and remove the background as a preprocessing stage to ensure fairness when compared to methods like NHA [GPL*22]. We also set the first frame of the video as the source frame for 2D-based methods and train NHA on the same training set as our method. We show the quantitative results in Table 2. Our method offers the best performance among all the methods across all metrics. Specifically, it can handle difficult head poses like 90° to the left and to the right.

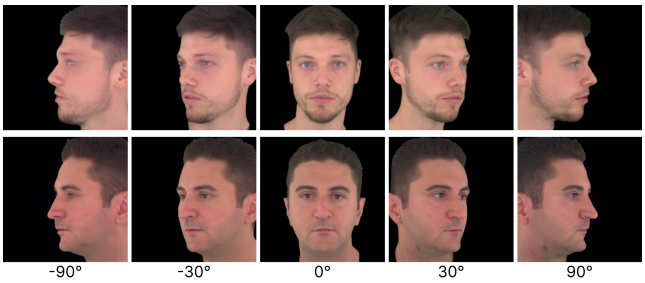

Figure 6: ***Visual results of our method with extreme head poses.*** *A key contribution of our method is to enable direct control over head poses with StyleGAN renderings. Our method can synthesize renderings with yaw from −90° to 90°, which previous 2D-based methods cannot achieve.*

Moreover, the personalized manifold enhances the details in the reconstructed images as the StyleGAN generator is fine-tuned to produce highly-similar images as the input video. Note that although our method is a 2D-based method, it still can produce multi-view consistent imagery. This is because the head rotations are mostly represented by interior points of the personalized manifold. And we observe that the interpolation between the pivots show good consistency for the identity and the geometry. As a result, our method can provide better visual quality than 3D methods, such as NHA. Finally, our method supports real-time rendering thanks to its lightweight mapping network. The inference time for our method is 0.018s, which is about 54 FPS, on an RTX 3080 GPU.

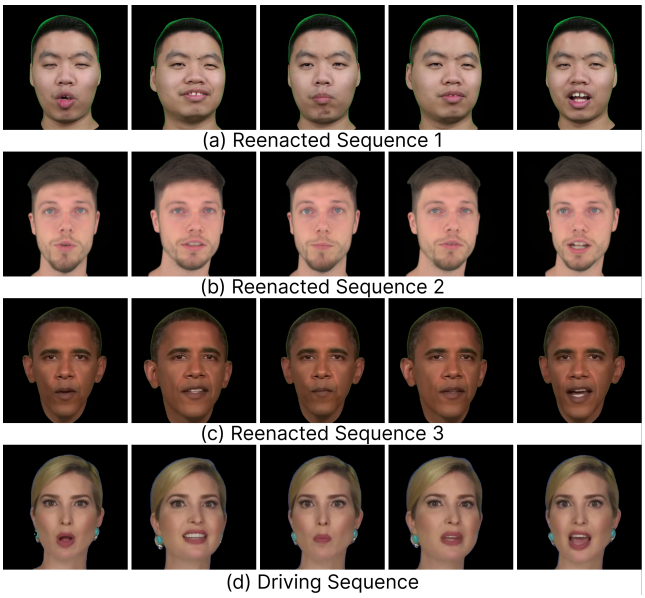

Figure 7: ***Reenactment results of our method.*** *We show reenacted sequences in (a), (b) and (c), and the driving sequence in (d). Our method is able to transfer the expressions across different subjects.*

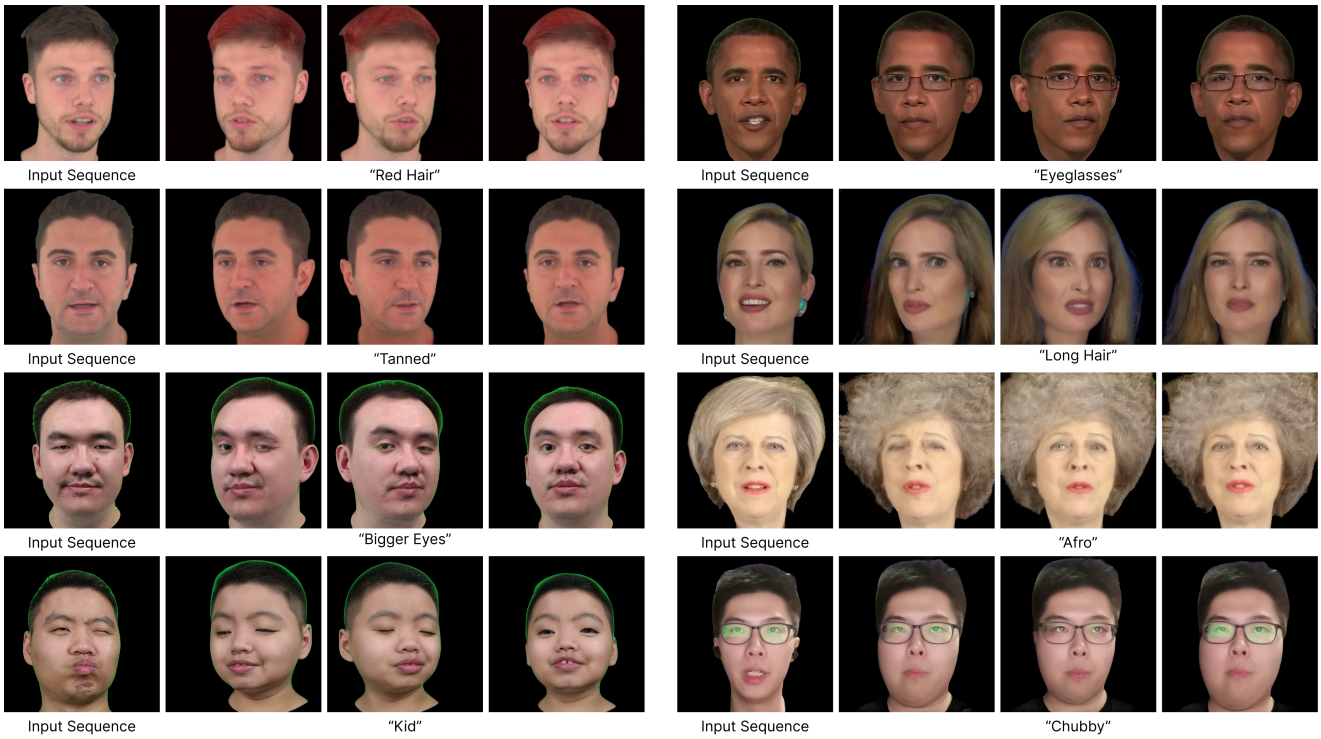

Figure 8: ***Editing results of our method.*** *We show a random frame from the input video sequence and the edited renderings with different head pose and facial expressions. Below each inset, we show the StyleCLIP prompt to edit the latent code and produce different editing results. Our method preserves the same identity and edits across different expression and pose changes. For example, in the "Kid" prompt, we show various expressions of the subject as a younger avatar and keep similar identities. Also, note the tint in the "Chubby" prompt is preserved and shows view-dependent changes across different viewpoints.*

5.3. Qualitative Results

Figure 5 demonstrates the visual results of our method against other baseline methods. Our algorithm is able to reconstruct face poses at extreme viewpoints, while previous methods like StyleHEAT and PIRender fail to generate reasonable results. We can observe distorted face cheeks in (a) for both methods and even some repeating patterns near the back of the head. Moreover, StyleHEAT often produces changes in the subject's eyes, as can be seen in (b) and (c). The StyleHEAT results are overly smoothed, leading to missing details like wrinkles seen in (d) and (e). On the other hand, PiRenderer, while keeping some details, misses the mouth texture, such as missing teeth shown in (d) and (f). As for LIA, it produces tone shifts in all the results, as well as missing details, for example the earrings in (d) and the tint on the eyeglasses in (f). LIA also fails to reconstruct the expression correctly, and it can be observed in (b), where the subject's eyes are fully open instead of half open as shown in the GT. NHA has a hard time reconstructing faces captured with fewer head poses. As shown in (a), (b) and (c), we can observe that the ears are heavily distorted due to NHA's optimization process. Moreover, it fails to reconstruct the teeth region in (f) and it does not match the expression in (d) and (e). We show additional comparison with NeRFBlendShape in Appendix D.

In Figs. 4 and 6, we demonstrate how our method can be used to generate rotated views of the subject, while fixing the expression. Previous methods [YZC*22] often have a difficult time handling extreme viewpoints as these viewpoints are out of the distribution of the pretrained StyleGAN generator. Since we construct a personalized manifold, our method can represent extreme head poses well, as long as they are presented in the input views. Note that we choose to learn the blending weights in the manifold, instead of an editing direction which controls the head motion. This design choice is because we found that while the linear interpolation can be smoothly changing between pivots, it does not fully represent the desired head motion. In other words, the personalized manifold might still contain some nonlinearity which causes the actual rotation to follow along a curved path, instead of a linear path. The learned MLP ensures that we can follow a correct trajectory to represent different head poses in a high-dimensional latent space.

Moreover, we show reenactment results in Fig. 7. Our method can be used to reenact different subjects with the provided expression parameters. We extract the FLAME parameters ψ, ϕ, γ and ξ from the driving sequence in (d) and feed it to the learned mapping networks for each identity in (a), (b) and (c). Then, we can produce reenactment results with good accuracy. For instance, in the rightmost column, we show reenactment results of all avatars with the mouth open expression. Each avatar can do this expression properly with their unique and accurate teeth texture, which is often difficult to recover in previous methods. We notice that since the FLAME

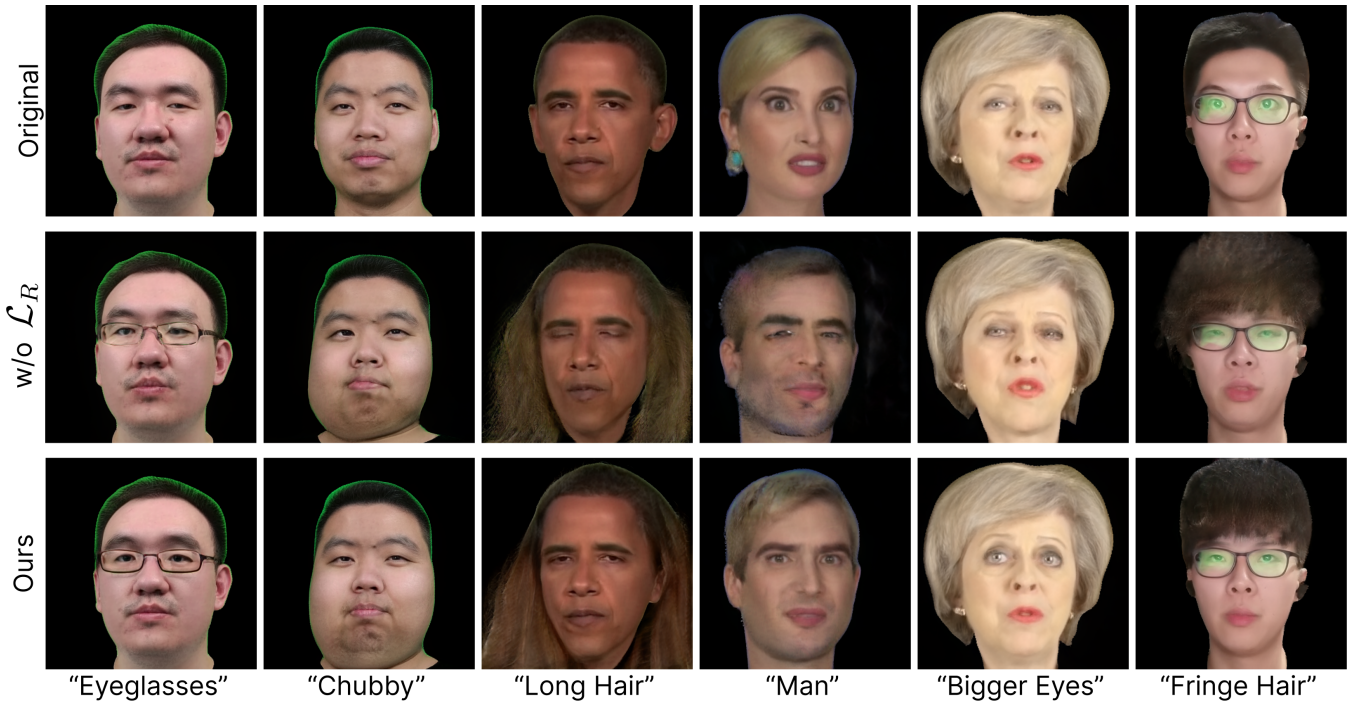

Figure 9: ***Effects of the regularizer on the editing results.*** *We observe that using selected frames from the input video could make it easier to overfit. Consequently, different from MyStyle [NAH*22], we apply a regularizer during the fine-tuning stage to maintain the editing ability of StyleGAN.*

parameters might have different distributions for each individual, it is better to renormalize the input parameters with the mean and standard deviation of the source and target.

Finally, we show editing results in Fig. 8. In the figure, we demonstrate various editing tasks using StyleCLIP [PWS*21]. Our method can be adapted to use other editing methods like InterfaceGAN [SYTZ20] and GANSpace [HHLP20]. Note that our method provides consistent renderings and maintains the edits even after the head is rotated. Furthermore, we show that the edited avatar can still produce various expressions, as shown in the "Kid" inset. The edited latent code also shows good multi-view consistency. Most notably, the "Eyeglasses" inset shows rotated versions of the subject with the eyewear with good geometry across different head poses. Some view-dependent effects are retained, for instance, the tint on the eyeglasses in the "Chubby" inset.

It is worth noting that while we fine-tune the StyleGAN generator on selected video frames, the editing capability is not impacted thanks to the regularizer $\mathcal{L}_R$. Comparison of the regularizer can be found in Fig. 9. Since we only have a handful of views of the given subject, it is highly likely for the fine-tuning stage to overfit to the distribution of these views. Therefore, it is critical to enforce the regularizer to avoid degraded results in editing. For instance, without the regularizer, the edited images often show color shift artifacts as shown in "Eyeglasses" and "Long Hair". Additionally, the details and expressions are not retained after the edits, as shown in the "Chubby" and "Man" insets.

5.4. Limitations and Future Work

While our proposed method shows promising results in enabling editing of pose, expression, and appearance of digital avatars, there are several limitations that we would like to address in future work.

First, our method only handles the regions within the face alignment bounding box, excluding the back of the head and upper body. Future work could focus on improving the StyleGAN generator or cropping to incorporate these regions, allowing for a more comprehensive representation of the subject.

In addition, our approach requires learning a personalized manifold for each subject, which can take some time for optimization. To be more specific, learning the manifold takes around 3 hours for 200 pivot images on an RTX 3080 and training the mapping networks takes around 4 hours on an NVIDIA A10 24 GB GPU. Note that after the optimization stage, our pipeline runs in real-time at 54 FPS on an RTX 3080. It could be interesting to explore meta-learning and have a network that outputs a personalized manifold directly by looking at images. Also, the mapping network could be pretrained on a large-scale dataset and apply to each avatar directly or with a fast fine-tuning stage. This could potentially remove the optimization overhead and enable faster adaptation to new subjects.

We also note that sometimes the gaze or eye regions are not perfect. However, this is not bound by our network design, but rather the performance of the DECA algorithm. For future work, it is possible to swap out DECA for better facial expression detectors like EMOCA [DBB22] and add some gaze regularization.

Finally, our method works best for interpolation, but may not perform well when extrapolating beyond the training data. Future work could explore additional regularization methods or data augmentation techniques to enable extrapolation and improve the overall generalization of the model.

5.5. Ethical considerations

The success of recent approaches in synthesizing photorealistic editable representations of a given subject, as achieved in this paper, has necessitated the introduction of various methods to detect if an image is fake [MNM*22]. However, the best methods can often be used as critics in the training paradigm of the state-of-the-art generative or editing models, in order to avoid detection. Additionally, as models become better, existing detection methods may be unable to scale. To prevent the misuse of editing methods, the development of more robust detection and verification techniques is paramount.

6. Conclusions

We propose a novel algorithm that encodes a monocular portrait video into a personalized manifold to enable editing on pose, expression, and appearance. Our approach selects useful pivots from the video sequence, allowing for efficient learning of the personalized manifold. We also design loss functions to learn pose and expression mapping networks, enabling granular control of the rendering given only a single video. Moreover, our method provides good editing capability through various StyleGAN editing methods. Overall, our work significantly contributes to the development of digital avatars by making them more interactive, engaging, and personalized.

Acknowledgement

This work was funded in part by a Qualcomm FMA Fellowship, ONR grants N000142012529, N000142312526, an NSF Graduate Fellowship, NSF grants 2100237 and 2120019, and the Ronald L. Graham Chair. We also acknowledge gifts from Adobe, Google, Meta and a Sony Research Award. All of the datasets mentioned in this paper were solely downloaded and used by University of California San Diego.

References

[AQW19] Abdal R., Qin Y., Wonka P.: Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 4432–4441. 4

[ATDN22] Aneja S., Thies J., Dai A., Niessner M.: ClipFace: Text-guided Editing of Textured 3D Morphable Models. In *ArXiv preprint arXiv:2212.01406* (2022). 2

[AXS*22] Athar S., Xu Z., Sunkavalli K., Shechtman E., Shu Z.: Rignerf: Fully controllable neural 3d portraits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 20364–20373. 1, 2, 3

[BV99] Blanz V., Vetter T.: A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), pp. 187–194. 2

[CLC*22] Chan E. R., Lin C. Z., Chan M. A., Nagano K., Pan B., Mello S. D., Gallo O., Guibas L., Tremblay J., Khamis S., Karras T., Wetzstein G.: Efficient geometry-aware 3D generative adversarial networks. In *CVPR* (2022). 3

[CLX*22] Chen A., Liu R., Xie L., Chen Z., Su H., Yu J.: Sofgan: A portrait image generator with dynamic styling. *ACM Transactions on Graphics (TOG) 41*, 1 (2022), 1–26. 3

[CMK*21] Chan E., Monteiro M., Kellnhofer P., Wu J., Wetzstein G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proc. CVPR* (2021). 3

[DBB22] Danecek R., Black M. J., Bolkart T.: EMOCA: Emotion driven monocular face capture and animation. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 20311–20322. 10

[DGXZ19] Deng J., Guo J., Xue N., Zafeiriou S.: Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 4690–4699. 6

[DYXT22] Deng Y., Yang J., Xiang J., Tong X.: Gram: Generative radiance manifolds for 3d-aware image generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022). 3

[FFBB21] Feng Y., Feng H., Black M. J., Bolkart T.: Learning an animatable detailed 3D face model from in-the-wild images. vol. 40. URL: https://doi.org/10.1145/3450626.3459936. 3, 5, 6

[FTET21] Fox G., Tewari A., Elgharib M., Theobalt C.: Stylevideogan: A temporal generative model using a pretrained stylegan, 2021. URL: https://vcai.mpi-inf.mpg.de/projects/stylevideogan. 5

[GLWT22] Gu J., Liu L., Wang P., Theobalt C.: Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations* (2022). 3

[GPL*22] Grassal P.-W., Prinzler M., Leistner T., Rother C., Niessner M., Thies J.: Neural head avatars from monocular rgb videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 18653–18664. 2, 3, 7, 8

[GTZN21] Gafni G., Thies J., Zollhöfer M., Niessner M.: Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021), pp. 8649–8658. 3

[GVR*14] Garrido P., Valgaerts L., Rehmsen O., Thormahlen T., Perez P., Theobalt C.: Automatic face reenactment. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 4217–4224. 3

[GZX*22] Gao X., Zhong C., Xiang J., Hong Y., Guo Y., Zhang J.: Reconstructing personalized semantic facial nerf models from monocular video. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 41*, 6 (2022). doi:10.1145/3550454.3555501. 1, 2, 3, 8, 13

[HB17] Huang X., Belongie S.: Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV* (2017). 4

[HHLP20] Härkönen E., Hertzmann A., Lehtinen J., Paris S.: Ganspace: Discovering interpretable gan controls. In *Proc. NeurIPS* (2020). 4, 10

[HPX*22] Hong Y., Peng B., Xiao H., Liu L., Zhang J.: Headnerf: A real-time nerf-based parametric head model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022). 1, 2, 3

[JCL*22] Jiang K., Chen S.-Y., Liu F.-L., Fu H., Gao L.: Nerffaceediting: Disentangled face editing in neural radiance fields. In *SIGGRAPH Asia 2022 Conference Papers* (2022), pp. 1–9. 2, 3

[KGT*18] Kim H., Garrido P., Tewari A., Xu W., Thies J., Niessner M., Pérez P., Richardt C., Zollhöfer M., Theobalt C.: Deep video portraits. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–14. 3

[KLA19] Karras T., Laine S., Aila T.: A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 4401–4410. 1, 5

[KLA*20] Karras T., Laine S., Aittala M., Hellsten J., Lehtinen J., Aila T.: Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR* (2020). 1, 2, 3, 4

[KSLZ22] Khakhulin T., Sklyarova V., Lempitsky V., Zakharov E.: Realistic one-shot mesh-based head avatars. In *European Conference of Computer vision (ECCV)* (2022). 2, 3

[LBB*17] Li T., Bolkart T., Black M. J., Li H., Romero J.: Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia) 36*, 6 (2017), 194:1–194:17. URL: https://doi.org/10.1145/3130800.3130813. 2, 3

[LD21] Leimkühler T., Drettakis G.: Freestylegan: Free-view editable portrait rendering with the camera manifold. doi:10.1145/3478513.3480538. 2, 3

[Llo82] Lloyd S.: Least squares quantization in pcm. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137. doi:10.1109/TIT.1982.1056489. 5

[MNM*22] Masood M., Nawaz M., Malik K. M., Javed A., Irtaza A., Malik H.: Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward. *Applied Intelligence* (2022), 1–53. 11

[MST*20] Mildenhall B., Srinivasan P., Tancik M., Barron J., Ramamoorthi R., Ng R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)* (2020), pp. I–405–I–421. 1, 3

[NAH*22] Nitzan Y., Aberman K., He Q., Liba O., Yarom M., Gandelsman Y., Mosseri I., Pritch Y., Cohen-Or D.: Mystyle: A personalized generative prior. *ACM Transactions on Graphics (TOG) 41*, 6 (2022), 1–10. 3, 4, 5, 8, 10

[OELS*22] Or-El R., Luo X., Shan M., Shechtman E., Park J. J., Kemelmacher-Shlizerman I.: StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), pp. 13503–13513. 3

[PGM*19] Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L., Desmaison A., Kopf A., Yang E., DeVito Z., Raison M., Tejani A., Chilamkurthy S., Steiner B., Fang L., Bai J., Chintala S.: Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 8

[PSB*21] Park K., Sinha U., Barron J. T., Bouaziz S., Goldman D. B., Seitz S. M., Martin-Brualla R.: Nerfies: Deformable neural radiance fields. *ICCV* (2021). 3

[PSH*21] Park K., Sinha U., Hedman P., Barron J. T., Bouaziz S., Goldman D. B., Martin-Brualla R., Seitz S. M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph. 40*, 6 (dec 2021). 3

[PWS*21] Patashnik O., Wu Z., Shechtman E., Cohen-Or D., Lischinski D.: Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2021), pp. 2085–2094. 2, 10

[RAP*21] Richardson E., Alaluf Y., Patashnik O., Nitzan Y., Azar Y., Shapiro S., Cohen-Or D.: Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021). 3

[RLC*21] Ren Y., Li G., Chen Y., Li T. H., Liu S.: Pirenderer: Controllable portrait image generation via semantic neural rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 13759–13768. 2, 6, 7

[RMBCO21] Roich D., Mokady R., Bermano A. H., Cohen-Or D.: Pivotal tuning for latent-based editing of real images. *ACM Trans. Graph.* (2021). 2, 3, 4, 5

[SLB*21] Sun T., Lin K.-E., Bi S., Xu Z., Ramamoorthi R.: Nelf: Neural light-transport field for portrait view synthesis and relighting. In *Eurographics Symposium on Rendering* (2021). 3

[SLHG22] Sun C., Liu Y., Han J., Gould S.: Nerfeditor: Differentiable style decomposition for full 3d scene editing. *arXiv preprint arXiv:2212.03848* (2022). 2

[SWH*22] Sun K., Wu S., Huang Z., Zhang N., Wang Q., Li H.: Controllable 3d face synthesis with conditional generative occupancy fields. In *NeurIPS* (2022). URL: http://papers.nips.cc/paper_files/paper/2022/hash/67b0e7c7c2a5780aeefe3b79caac106e-Abstract-Conference.html. 2

[SWS*22] Sun J., Wang X., Shi Y., Wang L., Wang J., Liu Y.: Ide-3d: Interactive disentangled editing for high-resolution 3d-aware portrait synthesis. *ACM Transactions on Graphics (TOG) 41*, 6 (2022), 1–10. 3

[SWZ*22] Sun J., Wang X., Zhang Y., Li X., Zhang Q., Liu Y., Wang J.: Fenerf: Face editing in neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 7672–7682. 3

[SYTZ20] Shen Y., Yang C., Tang X., Zhou B.: Interfacegan: Interpreting the disentangled face representation learned by gans. *TPAMI* (2020). 2, 4, 10

[TAN*21] Tov O., Alaluf Y., Nitzan Y., Patashnik O., Cohen-Or D.: Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–14. 3, 4, 6

[TEB*20] Tewari A., Elgharib M., Bharaj G., Bernard F., Seidel H.-P., Pérez P., Zöllhofer M., Theobalt C.: Stylerig: Rigging stylegan for 3d control over portrait images, cvpr 2020. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (june 2020), IEEE. 3

[TL18] Tran L., Liu X.: Nonlinear 3d face morphable model. In *In Proceeding of IEEE Computer Vision and Pattern Recognition* (Salt Lake City, UT, June 2018). 3

[TMG*22] Tzaban R., Mokady R., Gal R., Bermano A., Cohen-Or D.: Stitch it in time: Gan-based facial editing of real videos. In *SIGGRAPH Asia 2022 Conference Papers* (2022), pp. 1–9. 5

[TZS*16] Thies J., Zollhöfer M., Stamminger M., Theobalt C., Niessner M.: Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2016). 3

[WBLP11] Weise T., Bouaziz S., Li H., Pauly M.: Realtime performance-based facial animation. *ACM transactions on graphics (TOG) 30*, 4 (2011), 1–10. 3

[WYBD22] Wang Y., Yang D., Bremond F., Dantcheva A.: Latent image animator: Learning to animate images via latent space navigation. In *International Conference on Learning Representations* (2022). 2, 6, 7

[XWZ*22] Xu Y., Wang L., Zhao X., Zhang H., Liu Y.: Manvatar : Fast 3d head avatar reconstruction using motion-aware neural voxels, 2022. URL: https://arxiv.org/abs/2211.13206, doi:10.48550/ARXIV.2211.13206. 1

[YZC*22] Yin F., Zhang Y., Cun X., Cao M., Fan Y., Wang X., Bai Q., Wu B., Wang J., Yang Y.: Styleheat: One-shot high-resolution editable talking face generation via pre-trained stylegan. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII* (2022), Springer, pp. 85–101. 2, 3, 6, 7, 9

[ZAB*22] Zheng Y., Abrevaya V. F., Bühler M. C., Chen X., Black M. J., Hilliges O.: I M Avatar: Implicit morphable head avatars from videos. In *Computer Vision and Pattern Recognition (CVPR)* (2022). 3

[ZIE*18] Zhang R., Isola P., Efros A. A., Shechtman E., Wang O.: The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR* (2018). 4, 6

[ZLW*22] Zhang J., Li X., Wan Z., Wang C., Liao J.: Fdnerf: Few-shot dynamic neural radiance fields for face reconstruction and expression editing. In *SIGGRAPH Asia 2022 Conference Papers* (2022), pp. 1–9. 1, 2, 3

[ZXNT21] Zhou P., Xie L., Ni B., Tian Q.: CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis. arXiv:2110.09788. 3

Appendix

Appendix A: Perturbed Renderings for $\mathcal{L}_{\text{expr}}$

We show the renderings from the perturbed parameters γ'_i in Fig. 10. The expression matching loss $\mathcal{L}_{\text{expr}}$ helps the network learn unseen expressions through perturbing the parameters and matching the predictions from DECA.

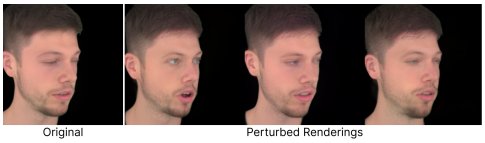

Figure 10: ***Renderings from the perturbed parameters γ'_i.***

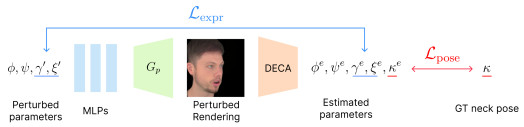

Figure 11: ***Illustration of our supervision.***

Appendix B: Implementation Details

We discuss the implementation details of our algorithm in this section. $\mathcal{F}_{\text{rot}}$ is a 2-layer MLP with leaky ReLU as the nonlinear activation layer in the middle and Tanh as the output layer. There are 128 channels for the hidden layer. Instead of outputting the blending weights directly, the network predicts the residual weights from the centroid, which has equal weights from all pivots, to ensure the initialization is around the centroid (since network output is small initially). Then it linearly combines the blending weights into a latent code in the $\mathcal{W}+$ space. Additionally, to prevent the MLP from generating values lower than negative beta, we shift the output values by beta and apply a SoftPlus function. This would effectively bound the values. $\mathcal{F}_{\text{expr}}$ is composed of the same 2-layer MLPs, except that there are 8 of them to control the first 8 layers of the latent code in the $\mathcal{W}+$ space. The predicted values are added to the latent code output from $\mathcal{F}_{\text{rot}}$.

Another implementation detail is that we perform a normalization stage for reenactment. The idea is that the distributions of expressions are different for each video. Since we only train on a single input video, it is possible to overfit to the given sequence. As a result, we normalize the driving features with standard deviation and mean of the expressions from the source and driving video before input it to the $\mathcal{F}_{\text{expr}}$ for reenactment.

For the clustering stage discussed in Sec. 4.1, we use K=200 in our experiments. It is important when the expression distribution in the data is uneven (e.g. same expressions for a long time) as the clustering can select more representative frames.

In terms of rendering resolution, we downsample our results from 1024×1024 to 512×512 for fair comparison with other baseline methods. However, our method can run in 1024×1024 resolution with similar efficiency (removing the downsampling stage), given input videos with 1024×1024 resolution.

Appendix C: Dataset Composition

We show the range of head poses in Table 3. Dataset names starting with "id" denote the subjects from the NeRFBlendShape dataset, whereas names starting with "person" denote subjects from the NHA dataset. It is difficult to determine the number of expressions as they are often transitioning from one to another (e.g. from smiling to grinning). However, in the NHA and NeRFBlendShape dataset, there are different expressions like smiling, winking and puffing.

Table 3: ***Composition of our dataset.*** *We show the range of head poses (in degrees) of each video sequence. Our evaluation dataset contains a wide range of head poses.*

Dataset	Min Yaw	Max Yaw	Min Pitch	Max Pitch
id1	-45.36	32.29	-23.25	17.06
id2	-57.74	35.85	-39.35	15.70
id3	-13.43	14.38	-20.60	12.44
id4	-29.09	12.39	-14.98	14.95
id5	-17.22	12.27	-5.42	11.41
id6	-9.62	2.31	-9.67	-1.38
id7	-39.96	36.23	-22.38	13.24
id8	-4.98	10.09	-20.16	-4.51
person0000	-75.56	75.92	-35.00	10.49
person0004	-70.62	72.65	-37.2	17.85

Appendix D: Additional Comparison with NeRFBlendShape

We show additional experiments with NeRFBlendShape [GZX*22] in Table 4 and Fig. 12. In Table 4, we provide two different numbers: original and masked. The original set is the same as Table 2, where we evaluate the whole image. The masked set uses the alpha values from NeRFBlendShape as the mask and only evaluate the face regions. While our method is purely 2D, it shows comparable performance to SOTA 3D methods. In Fig. 12, we demonstrate that our method produces sharper details, whereas NeRFBlendShape could produce ghosting artifacts.

Table 4: ***Evaluation of ours and NeRFBlendShape.*** *We show quantitative results from the NeRFBlendShape data. Our method demonstrates comparable performance. Moreover, our method offers better and sharper image in terms of the LPIPS.*

Methods	Masked	LPIPS↓	PSNR↑	SSIM↑
NeRFBlendShape	✗	0.1645	20.20	0.8689
Ours	✗	**0.1119**	**26.63**	**0.8803**
NeRFBlendShape	✓	0.0981	**32.05**	**0.9323**
Ours	✓	**0.0856**	31.76	0.9256

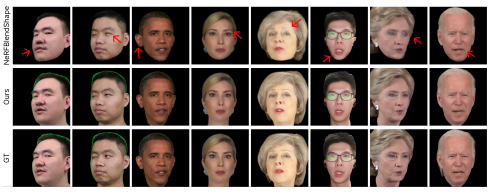

Figure 12: ***Qualitative comparison with NeRFBlendShape*** *Our method provides sharper details compared to NeRFBlendShape [GZX*22]. While NeRFBlendShape shows more accurate geometry, it produces artifacts when transitioning between different expressions, as highlighted above. Please zoom in to better see the details.*