PVP: Personalized Video Prior for Editable Dynamic Portraits using StyleGAN

K.-E. Lin\textsuperscript{1}, A. Trevithick\textsuperscript{1}, K. Cheng\textsuperscript{2}, M. Sarkis\textsuperscript{2}, M. Ghafoorian\textsuperscript{2}, N. Bi\textsuperscript{2}, G. Reitmayr\textsuperscript{2}, R. Ramamoorthi\textsuperscript{1}

\textsuperscript{1}University of California, San Diego \hspace{1em} \textsuperscript{2}Qualcomm Technologies Inc.

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
\begin{itemize}
  \item Computing methodologies \rightarrow Image-based rendering;
\end{itemize}

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-
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 users 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 users to edit on trajectory 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-

### Table 1: Comparison of different methods

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
<th>Head pose editing</th>
<th>Appearance editing</th>
<th>Monocular input</th>
<th>Riggable</th>
<th>Personalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF</td>
<td>HeadNeRF [HPX∗22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NeRF</td>
<td>RigNeRF [AXS∗22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NeRF</td>
<td>FDNeRF [ZLW∗22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NeRF</td>
<td>cGOF [SWH∗22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NeRF</td>
<td>NeRFBlendShape [GZX∗22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NeRF</td>
<td>NeRFFaceEditing [JCL∗22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parametric Model</td>
<td>NHA [GPL∗22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parametric Model</td>
<td>ROME [KSLZ∗22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parametric Model</td>
<td>FreeStyleGAN [LD21]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parametric Model</td>
<td>PIRendering [RLC∗21]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parametric Model</td>
<td>LIA [WYBD22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parametric Model</td>
<td>StyleHEAT [YZC∗22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ours</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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).
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, OEL’S2, DYXT22] generate photorealistic 3D representations from 2D images, the editability of such models requires further research to reach its full potential. SoiGAN [CLX’22], IDE-3D [SW’S2], FENeRF [SWZ’22] and CIPS-3D [XZNT21] 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 NeRFBBlendShape [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 [KSLZ’22] 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 \( Z \) space) to a mapper network, which maps the input to the intermediate latent code \( w \in \mathcal{W} \). The latent
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 ‘\( W^+ \) space. We can then write the StyleGAN image synthesis as \( I' = G(w^+; \theta) \), for a StyleGAN generator \( G \) with weights \( \theta \).

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 ‘\( W \)’ [KLA20], or ‘\( W^+ \)’ [AQW19]. However, there is a tradeoff between the two methods. ‘\( W \)’ space inversion offers better editability but poor reconstruction quality, whereas ‘\( W^+ \)’ provides superior reconstruction, yet produces inferior editing results. To address this tradeoff, the better way is to find pivot latent codes in the ‘\( W^+ \)’ space via e4e [TAN21] 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 (\( \geq 60^\circ \)). To be more specific, we can define the pivots as

\[
\begin{align*}
    w^+ = E(I),
    \end{align*}
\]

where \( E \) is the e4e encoder, \( I \) the input image, and \( w^+ \) the inverted latent code in ‘\( W^+ \)’ space. And we can synthesize the rendering with the StyleGAN \( G \) by

\[
\begin{align*}
    I' = G(w^+; \theta),
    \end{align*}
\]

where \( \theta \) 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 \( \theta_p \) with the following objective:

\[
\begin{align*}
    \theta_p = \arg \min_{\theta} L_{\text{LPIPS}}(I', I) + \lambda^p_{L_2} L_2(I', I) + \lambda^p_{L_2} L_2,
    \end{align*}
\]

where \( L_{\text{LPIPS}} \) is the perceptual loss [ZIE18], \( L_2 \) denotes the MSE loss, \( \lambda^p_{L_2} \) denotes the weight for MSE, the loss, \( L_2 \) is the locality regularization loss [RMBCO21], and \( \lambda^p_{L_2} \) the weight for the regularizer. The p superscript denotes personalization, since we use \( L_2 \) 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 [NAH22].

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

\[
\begin{align*}
    I'_\text{edit} = G(w^+ + \Delta w^+; \theta_p),
    \end{align*}
\]

where \( \Delta w^+ \) 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 poses 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 [KLA20]. A naive way to address this is to perform GAN
inversion on a frame-by-frame basis. However, previous methods generally struggle with extreme viewing angles (~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, TMG22]. 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 \( \psi \), pitch \( \phi \), neck pose \( \kappa \), jaw pose \( \gamma \) and expression parameters \( \xi \) of a given sequence. We then stack the \( \psi, \phi \) and \( \xi \) 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 \( f = \{I_1, I_2, \ldots, I_K\} \). Finally, we optimize the StyleGAN generator, similar to Eq. 3, with the following objective:

\[
\theta_p = \arg \min_{\theta} \frac{1}{K} \sum_{i=1}^{K} \left[ L_{L1}(I_i^f, I_i^r) + \lambda_{L2}^p L_{L2}(I_i^f, I_i^r) + \lambda_R^p L_R \right].
\]  

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 \( G_p = G(w^*_i; \theta_p) \). Once we acquire the fine-tuned StyleGAN \( G_p \), we can define the \( \beta \)-dilated convex hull [NAH22] in the \( W^+ \) space as the local manifold \( W_p \):

\[
W_p = \sum_{i=1}^{K} \alpha_i w_i, \quad \text{s.t.} \quad \sum_{i=1}^{K} \alpha_i = 1, \quad \text{and} \quad \alpha_i \geq -\beta.
\]

where \( \alpha_i \) 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 [NAH22]. 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 makes 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 \( L_R \) 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 settings 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 \( F_{rot} \) to calculate the latent

\[
W_p = \sum_{i=1}^{K} \alpha_i w_i, \quad \text{s.t.} \quad \sum_{i=1}^{K} \alpha_i = 1, \quad \text{and} \quad \alpha_i \geq -\beta.
\]
code $w_{rot}$ as a linear combination of the pivots $\{w_i\}_{i=1}^K$ by
\[
    w_{rot} = \sum_{i=1}^K a_i^{rot} w_i, \quad \text{and} \quad \{a_i^{rot}\}_{i=1}^K = \mathcal{F}_{rot}(\phi, \psi; \theta_t),
\]
where $\phi$ denotes the pitch, $\psi$ the yaw of the head and $\theta_t$ represents the weights of $\mathcal{F}_{rot}$. We use the MLP to predict the blending weights instead of the latent code in $\mathcal{W}$ space or $\mathcal{W}^+$ space. Because of 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 $\gamma$ and the expression parameters $\xi$ and outputs the difference of the latent code $\Delta w_{expr}$ in $\mathcal{W}^+$ space. Specifically, we have
\[
    \Delta w_{expr} = \mathcal{F}_{expr}(\gamma, \xi; \theta_e). \tag{8}
\]
We then add it to the latent code $w_{rot}$:
\[
    w_{final} = w_{rot} + \Delta w_{expr}. \tag{9}
\]
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 $\mathcal{L}_{local}$ (see Sec. 4.3) to keep the changes small.

Finally, we can then render the novel view $I'$ with the fine-tuned StyleGAN generator $G_p$:
\[
    I' = G_p(w_{final}). \tag{10}
\]
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 $\theta_p$ and train the mapping networks $\mathcal{F}_{rot}$ and $\mathcal{F}_{expr}$ with the loss design discussed in this section, optimizing for their weights $\theta_t$ and $\theta_e$. As we aim to reconstruct an input video, we supervise the output with rendering losses like LPIPS [ZIE18] and $L_2$ 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 $\mathcal{L}_{id}$ [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 $\gamma$ and expression $\xi$, we render an additional rendering $I'$ with $G_p$ by changing the input to the perturbed parameters $\gamma' = \gamma + \epsilon, \xi' = \xi + \epsilon$, where we add a normal distribution $\epsilon \sim N(0, \sigma)$. Then we feed the new rendering $I'$ to the encoder of DECA [FFBB21] to predict yaw $\psi'$, pitch $\phi'$, jaw pose $\gamma'$, expression $\xi'$, and neck pose $\kappa'$. And define the expression matching loss as:
\[
    \mathcal{L}_{\text{expr}} = ||(\gamma', \xi') - (\gamma', \xi')||^2_2. \tag{11}
\]
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 $I'$ does not show large head motions. We define the pose consistency loss as:
\[
    \mathcal{L}_{\text{pose}} = ||\kappa' - \kappa||^2_2. \tag{12}
\]
Similarly, $\kappa$ is the neck pose for the input frame, and $\kappa'$ denotes the DECA-estimated neck pose of the perturbed rendering. Note that we do not feed the neck pose $\kappa$ as an input to the pose MLP. For the RGB consistency loss, we calculate a mask $\mu$ excluding the eyes and mouth regions, and then minimize the loss between the perturbed rendering $I'$ and the original rendering $I$
\[
    \mathcal{L}_{\text{cons}} = ||m \odot (I' - I)||^2_2. \tag{13}
\]
To further encourage the expression network to explore local regions in the manifold, we apply a regularization loss on the predicted latent $\Delta w_{expr}$, namely,
\[
    \mathcal{L}_{\text{local}} = ||\Delta w_{expr}||^2_2. \tag{14}
\]
Lastly, our training objective is defined as follows:
\[
    (\theta_t, \theta_e) = \arg \min_{\theta_t, \theta_e} \mathcal{L}_{\text{LPIPS}} + \lambda_{L2} \mathcal{L}_{L2} + \lambda_{id} \mathcal{L}_{id} + \lambda_{\text{cons}} \mathcal{L}_{\text{cons}} + \lambda_{\text{pose}} \mathcal{L}_{\text{pose}} + \lambda_{\text{expr}} \mathcal{L}_{\text{expr}} + \lambda_{\text{local}} \mathcal{L}_{\text{local}}, \tag{15}
\]
where we update the pose network weights $\theta_t$ and the expression network weights $\theta_e$.

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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>PSNR↑</th>
<th>LPIPS↓</th>
<th>SSIM↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIRenderer</td>
<td>21.49</td>
<td>0.2431</td>
<td>0.7540</td>
</tr>
<tr>
<td>LIA</td>
<td>23.47</td>
<td>0.1796</td>
<td>0.8545</td>
</tr>
<tr>
<td>StyleHEAT</td>
<td>20.29</td>
<td>0.2494</td>
<td>0.7685</td>
</tr>
<tr>
<td>NHA</td>
<td>25.31</td>
<td>0.1538</td>
<td>0.8746</td>
</tr>
<tr>
<td>Ours</td>
<td>26.63</td>
<td>0.1119</td>
<td>0.8803</td>
</tr>
</tbody>
</table>

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.
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). 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 \( \psi, \phi, \gamma \) and \( \xi \) 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
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 edit-able 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 monocural 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 function to learn pose and expression mapping networks, enabling granular control of the rendered 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


t-Confere.html. 2


dings of IEEE Computer Vision and Pattern Recognition (Salt Lake City, UT, June 2018). 3


© 2023 The Authors.
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.
Appendix

Appendix A: Perturbed Renderings for $L_{\text{expr}}$

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

Figure 10: Renderings from the perturbed parameters $\gamma'_i$.

Appendix B: Implementation Details

We discuss the implementation details of our algorithm in this section. $F_{\text{rot}}$ 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 expression distribution. The initial weights are set such that the training in the expression space. Additionally, to prevent the MLP from overfitting to the given sequence, we add the masked loss $L_{\text{mask}}$ to the expression loss $L_{\text{expr}}$.

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 $F_{\text{expr}}$ 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 $1024 \times 1024$ to $512 \times 512$ for fair comparison with other baseline methods. However, our method can run in $1024 \times 1024$ resolution with similar efficiency (removing the downsampling stage), given input videos with $1024 \times 1024$ 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Min Yaw</th>
<th>Max Yaw</th>
<th>Min Pitch</th>
<th>Max Pitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>id1</td>
<td>-65.76</td>
<td>52.29</td>
<td>-23.25</td>
<td>17.00</td>
</tr>
<tr>
<td>id2</td>
<td>-57.34</td>
<td>35.65</td>
<td>-39.25</td>
<td>15.70</td>
</tr>
<tr>
<td>id3</td>
<td>-15.41</td>
<td>14.38</td>
<td>-20.60</td>
<td>12.44</td>
</tr>
<tr>
<td>id4</td>
<td>-20.69</td>
<td>12.79</td>
<td>-14.00</td>
<td>14.95</td>
</tr>
<tr>
<td>id5</td>
<td>-11.22</td>
<td>12.27</td>
<td>-5.42</td>
<td>11.41</td>
</tr>
<tr>
<td>id6</td>
<td>-9.62</td>
<td>2.31</td>
<td>-9.62</td>
<td>-1.36</td>
</tr>
<tr>
<td>id7</td>
<td>39.36</td>
<td>36.23</td>
<td>-22.38</td>
<td>13.24</td>
</tr>
<tr>
<td>id8</td>
<td>-4.98</td>
<td>10.09</td>
<td>-20.16</td>
<td>-4.51</td>
</tr>
<tr>
<td>person0000</td>
<td>75.56</td>
<td>75.92</td>
<td>-35.00</td>
<td>10.49</td>
</tr>
<tr>
<td>person0001</td>
<td>70.64</td>
<td>72.65</td>
<td>-37.52</td>
<td>17.95</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Masked</th>
<th>LPIPS</th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRFBlendShape</td>
<td>✓</td>
<td>0.0949</td>
<td>20.30</td>
<td>0.8609</td>
</tr>
<tr>
<td>Ours</td>
<td>✓</td>
<td>0.0890</td>
<td>21.76</td>
<td>0.8703</td>
</tr>
<tr>
<td>NeRFBlendShape</td>
<td>✓</td>
<td>0.0949</td>
<td>20.30</td>
<td>0.8609</td>
</tr>
<tr>
<td>Ours</td>
<td>✓</td>
<td>0.0890</td>
<td>21.76</td>
<td>0.8703</td>
</tr>
</tbody>
</table>

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.

© 2023 The Authors.
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.