# PIRenderer: Controllable Portrait Image Generation via Semantic Neural Rendering

## *Supplementary Material*

## A . Additional Results of PIRenderer

In this section, we provide additional results of the proposed model. To achieve more intuitive comparisons, we show the results in a *Supplementary Video*. In this video, the following materials are provided:

- **Results of the intuitive portrait image editing task**. Although many commercial softwares are available for portrait image editing, complex and high-level modifications (e.g. modifying head posture or expression) are not supported or require professional skills. We show that our model can achieve interactive real-world facial image editing, which will greatly reduce the difficulty of image editing and help users to obtain satisfactory images.
- **Results of the motion imitation task**. In the same-identity reconstruction task, we show that our model can generate coherent videos with realistic details. Meanwhile, compared with the results of GFLA and FOMM, our model is more robust to the occlusions in the driving videos. In the cross-identity motion imitation task, we show that the proposed model can generate realistic results while preserving the source identity.
- **Results of the audio-driven facial reenactment task**. The generated videos as well as the input audios are provided. It can be seen that our model can generate accurate mouth motions and realistic other motions (eyes, head poses, *etc*.) for the given audios. Meanwhile, we can generate various motions from only single input audios and transform these motions into coherent videos.
- **Results of the ablation study**. We provide the results of the ablation study described in Sec. B . Using the coefficients of a window of continuous frames as the motion descriptors of the center frame can help the proposed model to achieve coherent results.
- **Results of the facial interpolation task**. The results of the facial interpolation task described in Sec. C are reported. It can be seen that our model learns a linear latent-space $\mathcal{Z}$, which enables interpolating images with smooth-varying motions.

## B . Analysis of the Target Motion Descriptor

In this paper, we extract 3DMM coefficients using an off-the-shelf 3D face reconstruction model [1]. Although this method produces relatively accurate results, errors and noise are inevitable. In our main paper, we mentioned that the estimation errors cause mismatches between the extracted motions and the real motions of target images, which will lead to performance degradation. Meanwhile, failing to model the temporal correlations of videos will cause incoherent videos. To alleviate these problems, we propose to use the coefficients of a window of continuous frames as the motion descriptors of the center frame. In this section, we prove the effectiveness of this choice.

An ablation model is trained by using the 3DMM coefficients of a single input frame as the target motion descriptors. The evaluation results of the same-identity reconstruction task are shown in Tab. B.1. It can be seen that our PIRenderer can generate images with more accurate target motions. This indicates that our network models the temporal correlations from the coefficients of the continuous frames and thus reduces the errors. Meanwhile, the subjective comparisons are provided in our supplementary videos. It can be seen that the ablation model cannot generate coherent videos, which reduces the reality of the results. Our PIRenderer can generate coherent results with accurate motions.

| | FID | AED | APD | LPIPS |
|---|---|---|---|---|
| Ablation | 8.752 | 0.1169 | 0.0182 | 0.1370 |
| Ours | **8.260** | **0.1106** | **0.0164** | **0.1285** |

Table B.1. Analysis of the target motion descriptor. The ablation model is trained by using the 3DMM coefficients of a single input frame, while our model uses the coefficients of a window of continuous frames as the motion descriptors of the center frame.


Figure C.1. The interpolation results of latent-space $\mathcal{Z}$. Our model can generate images with smooth-varying motions.

## C . Interpolation of Latent-space $\mathcal{Z}$

In this paper, we use a mapping network $f_m : \mathcal{P} \rightarrow \mathcal{Z}$ to map motion descriptors $\mathbf{p}$ to latent vectors $\mathbf{z}$. In this section, we show that our model can learn a linear latent-space $\mathcal{Z}$ which supports the task of facial motion interpolation. The interpolated images are generated by latent vectors $\mathbf{z}'$ calculated by:

$$\mathbf{z}' = \alpha f_m(\mathbf{p}_1) + (1 - \alpha) f_m(\mathbf{p}_2) \tag{1}$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are two different motions and $f_m$ is our mapping function. We first implement the interpolation task with real-world motions. The generated results can be found in Fig. C.1. It can be seen that our model can generate images with smooth-varying motions. Both expressions and poses are linearly transformed from the motion $\mathbf{p}_1$ to the motion $\mathbf{p}_2$ as the $\alpha$ increasing. Then we show that the interpolation can also be performed with a specific motion attribute. Fig. C.2 provides the generated results. Given motions $\mathbf{p}_1$ and $\mathbf{p}_2$ with the same expressions but different poses, the interpolated latent vectors $\mathbf{z}'$ control to generate images with the same expressions and smoothly-varying poses. Meanwhile, given motions $\mathbf{p}_1$ and $\mathbf{p}_2$ with the same poses but different expressions, only the facial expressions change in the interpolated results. The facial motion interpolation task can enable many applications such as exemplar-based portrait expression manipulation.

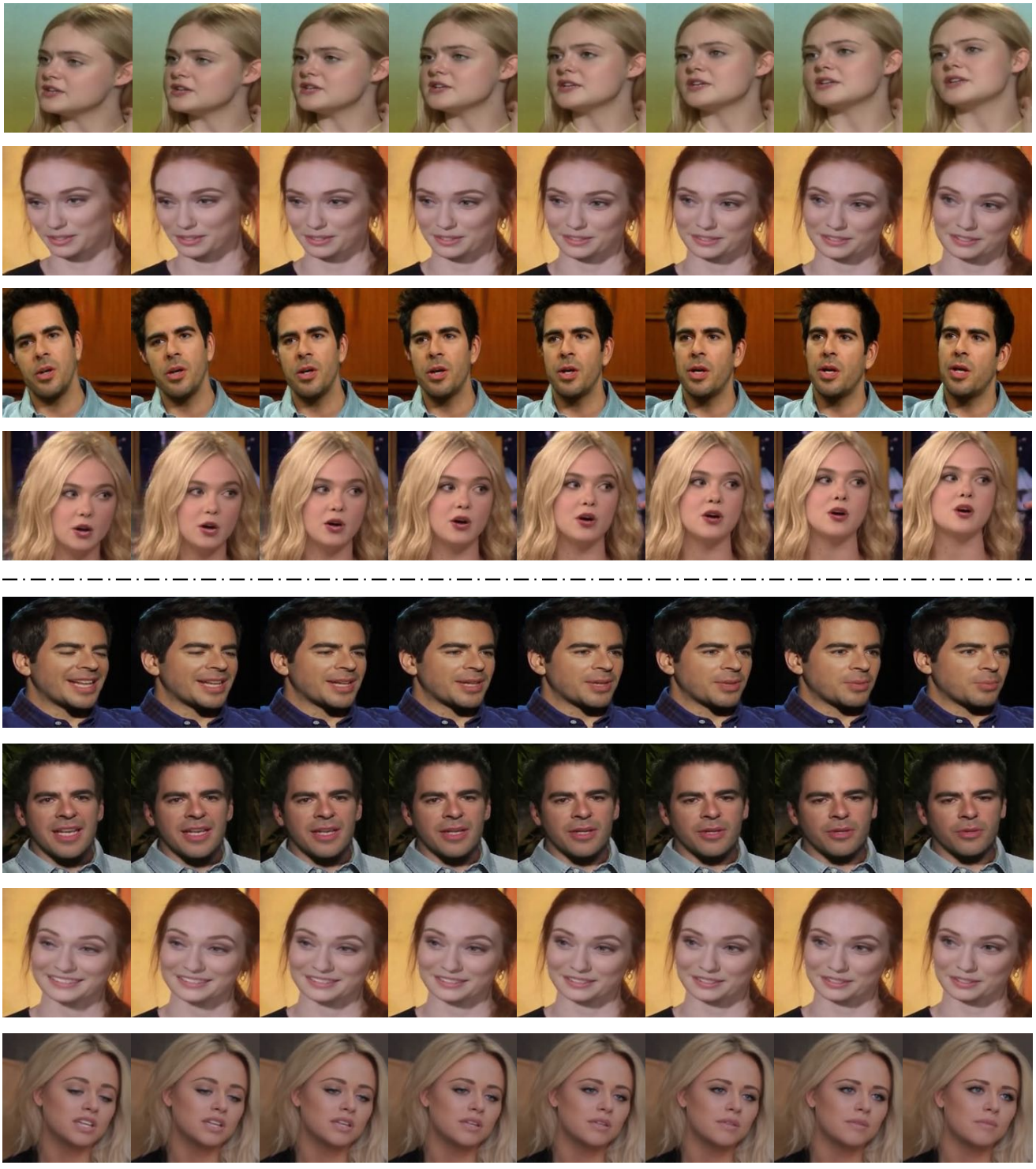


Figure C.2. The interpolation results of latent-space $\mathcal{Z}$. The top four rows show the results generated by interpolating motions with the same expressions and different poses. The bottom four rows show the results generated by interpolating motions with the same pose and different expressions.

# D . Implementation Details

## D .1. Implementation Details of PIRenderer

**Model Architecture.** The architecture of our PIRenderer is shown in Fig. D.4. The mapping network is responsible for transform the target motions $\mathbf{p} \in \mathcal{P}$ into latent vectors $\mathbf{z} \in \mathcal{Z}$. As discussed in our main paper, to alleviate the problem brought by the coefficient estimation errors, we use the coefficients of a window of $k$ continuous frames as the motion descriptor of the center frame. We set $k = 27$ for all experiments. Meanwhile, the 3D face reconstruction models always employ face alignment as the pre-processing method to crop input images such that faces of inputs have similar size and position to improve their performance. Therefore, instead of providing absolute translation parameters $\mathbf{t}$, these methods only estimate relative translations $\mathbf{t}'$. To describe the absolute face positions, we use the cropping parameters $\mathbf{t}_c$ together with the relative translations $\mathbf{t}'$ as our translations. The architecture of the mapping network is shown in Fig. D.4. We use 1D convolution layers to process the input motions. Leaky-ReLU is used as the activation function in this network. The architectures of the warping and editing network are shown in Fig. D.4. Auto-encoder structures are used to design these networks. Skip connections are employed to skip the high-resolution features. We use the architectures shown in Fig. D.3 as the basic components. ADAIN operation is used after each convolution layer of (a) ConvDown, (b) ResBlock, and (c) ResBlockUp to inject the latent vectors $\mathbf{z}$. Layer normalization is used as the activation normalization method of the other convolution layers. We use Leaky-ReLU as the non-linear function in our model.

**Training and Inference.** We train our model in stages. The mapping network and the warping network are first pre-trained for $200k$ iterations. Then we train the whole model for another $200k$ iteration in an end-to-end manner. We adopt the ADAM optimizer with an initial learning rate as $10^{-4}$. The learning rate is decreased to $2 \times 10^{-5}$ after $300k$ iterations. The batch size is set to 20 for all experiments. We set $\lambda_w = 2.5$, $\lambda_c = 4$, and $\lambda_s = 1000$. In the inference phase, we use coefficients of $k = 27$ continuous frames as the motion descriptors for the reenactment task. In the intuitive image editing task, we repeat the target motion $k = 27$ times as the motion descriptors.

## D .2. Implementation Details of $f_\theta$

We extent our PIRenderer to tackle the audio-driven facial reenactment task by training an additional mapping function $f_\theta$. The mapping function $f_\theta$ is responsible for generating sequential 3DMM coefficients from audios. As discussed in the main paper, *Normalizing flow* is employed to design $f_\theta$. The core idea of normalizing flow is to train an invertible and differentiable nonlinear mapping function that maps samples from a simple known distribution (*e.g.* Gaussian) to a more complex distribution. In our conditional setting, the function is trained to map motion-condition pairs $(\mathbf{p}, \mathbf{c})$ to latent variables $\mathbf{n}$ with $\mathbf{n} = f_\theta^{-1}(\mathbf{p}, \mathbf{c})$. Function $f_\theta$ is composed of a sequence of invertible transformations: $f_\theta = f_1 \circ f_2 \circ \cdots \circ f_K$, such that the relationship between $\mathbf{p}$ and $\mathbf{n}$ can be written as:

$$\mathbf{n} \xleftrightarrow{f_1(*,\mathbf{c})} \mathbf{h}_1 \xleftrightarrow{f_2(*,\mathbf{c})} \mathbf{h}_2 \cdots \xleftrightarrow{f_K(*,\mathbf{c})} \mathbf{p} \tag{2}$$

The key aspect of normalizing flows is that the probability density function $p_{\mathbf{p}|\mathbf{c}}$ can be explicitly computed as:

$$\begin{aligned} \log p_{\mathbf{p}|\mathbf{c}}(\mathbf{p}|\mathbf{c}, \theta) &= \log p_{\mathbf{n}}(f_\theta^{-1}(\mathbf{p}, \mathbf{c})) + \log \left| \det \frac{\partial f_\theta^{-1}}{\partial \mathbf{p}}(\mathbf{p}, \mathbf{c}) \right| \\ &= \log p_{\mathbf{n}}(f_\theta^{-1}(\mathbf{p}, \mathbf{c})) + \sum_{j=1}^{K} \log \left| \det \frac{\partial f_j^{-1}}{\partial \mathbf{h}_j}(\mathbf{h}_j, \mathbf{c}) \right| \end{aligned} \tag{3}$$

where we define $\mathbf{h}_K \equiv \mathbf{p}$ for conciseness. We design $f_n$ using a similar architecture as that of Glow [2]. Each transformation contains three sub-steps: an actnorm function; a linear transformation; and an affine coupling layer; Let $\mathbf{x}$ signifies the input of each layer, and $\mathbf{y}$ signifies the output. Both $\mathbf{x}$ and $\mathbf{y}$ are tensors of shape $[c \times t]$ with channel dimension $c$ and time dimension $t$. Actnorm is an affine transformation of the activations using a scale and bias parameter per channel with $\mathbf{y}_t = \mathbf{s} \odot \mathbf{x}_t + \mathbf{b}$. The linear transformation is used to transform the input tensor with the trainable parameters $\mathbf{W} \in \mathbb{R}^{C \times C}$ with $\mathbf{y}_t = \mathbf{W}\mathbf{x}_t$. The affine crouping layer first splits the input tensor $(\mathbf{x}_a, \mathbf{x}_b) = \mathrm{Split}(\mathbf{x})$. Then, a neural network is used to predict the affine parameters $(\log \mathbf{s}, \mathbf{t}) = \mathrm{NN}(\mathrm{concat}(\mathbf{x}_b, \mathbf{c}))$. Finally the output is obtained by $\mathbf{y} = \mathrm{concat}(\mathbf{s} \odot \mathbf{x}_a + \mathbf{t}, \mathbf{x}_b)$.

Our task requires generating sequential motion descriptors $\mathbf{p}$. Thus, modeling the temporal correlations is a crucial challenge for this task. To handle this, we generate the motions in a recurrent manner. The previously generated $k$ motions $\mathbf{p}_{i-k:i-1}$ are used as a part of conditional information for the current generation. Meanwhile, we design the neural network

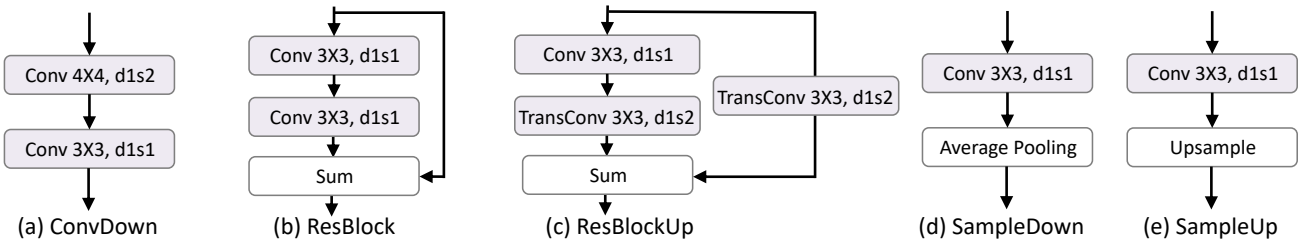


Figure D.3. The components used in our networks. The ADAIN operation is used after each convolution layer of (a) ConvDown, (b) ResBlock, and (c) ResBlockUp.

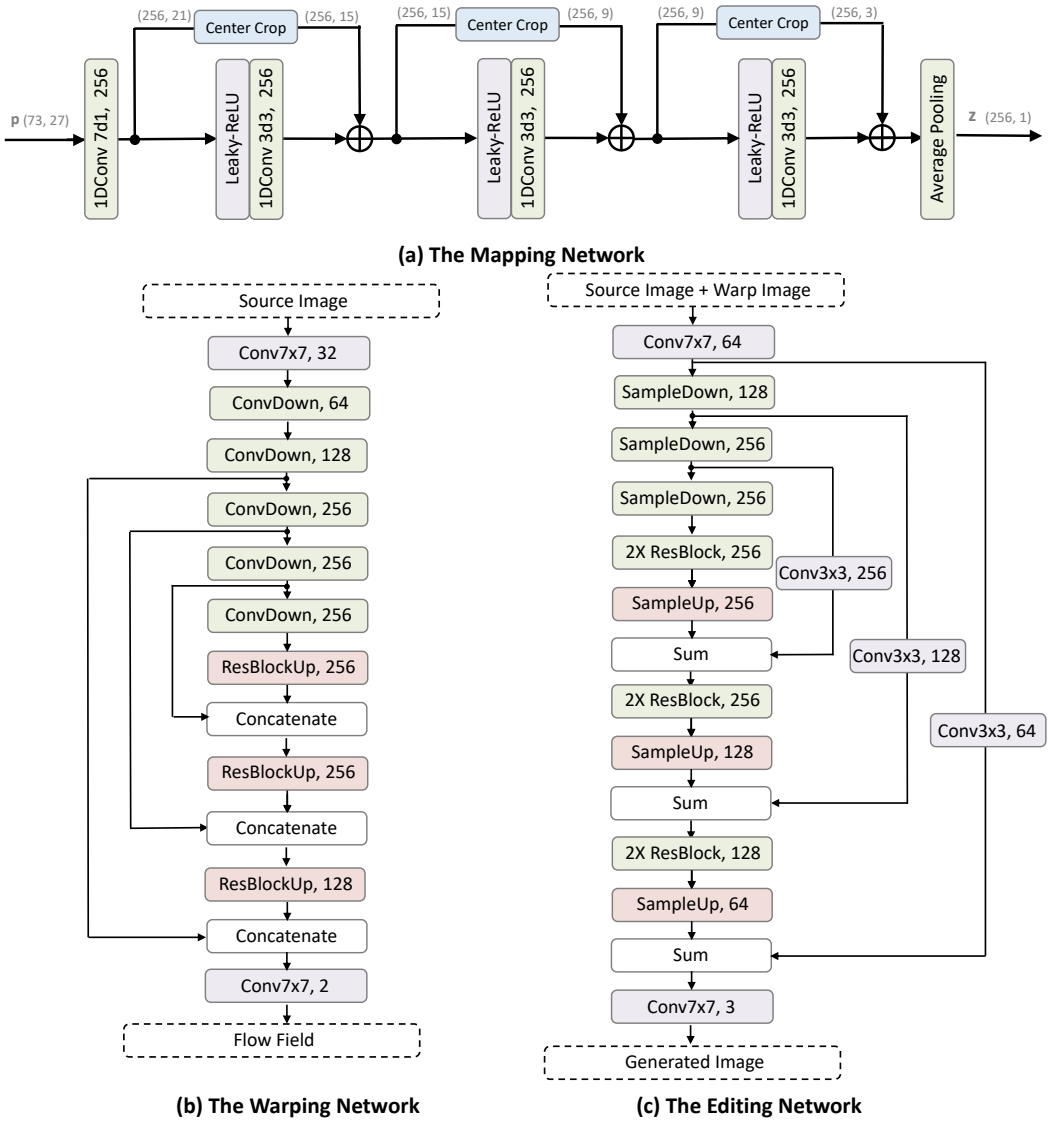


Figure D.4. The architecture of our PIRenderer.

$\mathrm{NN}(*)$ in the affine coupling layers as LSTM modules to further model the temporal correlations. Instead of generating the motions $\mathbf{p}$ using a single $f_\theta$, we design two mapping functions $f_{\theta_1}$ and $f_{\theta_2}$ to generate expressions $\boldsymbol{\beta} \in \mathbb{R}^{64}$ and positions $\mathbf{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3$ respectively. For the expression mapping function $f_{\theta_1}$, the conditional information $\mathbf{c}_i$ consists of two parts: a window of previous expressions $\boldsymbol{\beta}_{i-k:i-1}$ and a window of audio single $\mathbf{a}_{i-k:i+\tau}$. For the position mapping function $f_{\theta_2}$, in addition to the previously generated motions and the input audios, we further add the initial position of the first frame to the conditional information to help the model building the long-term relationship. We design $K = 10$ for $f_{\theta_1}$ and $K = 8$ for $f_{\theta_2}$. For all experiments, we set $k = 5$ and $\tau = 6$.

In the training phase, we train our mapping functions to generate latent vectors $\mathbf{n}_i$ from the ground truth motion $\mathbf{p}_i$ and the corresponding conditional information $\mathbf{c}_i$. The negative log-likelihood loss is used as the training loss.

$$\mathcal{L}_{nll} = -\log p_n(\mathbf{n}_i) - \sum_{j=1}^{K} \log \left| \det \frac{\partial f_j^{-1}}{\partial \mathbf{h}_j}(\mathbf{h}_j, \mathbf{c}_i) \right| \tag{4}$$

In the inference phase, we randomly sample latent vectors $\mathbf{n} \sim p(\mathbf{n})$ and generate the sequential motions using $\mathbf{p} = f_\theta(\mathbf{n}, \mathbf{c})$. The motions in the conditional information $\mathbf{c}_0$ are initialized as the motion of the source image.

## References


[1] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 1

[2] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018. 4