PERSE: Personalized 3D Generative Avatars from A Single Portrait

Supplementary Material

A. Implementation Details

A.1. Avatar Model

A.1.1. Avatar Model Architecture

To model diverse attributes with a single model, our avatar model follows three-stage deformations proposed in PEGASUS [2] with a few modifications. First, we initialize the learnable generic canonical points P^{gc} with the vertices of a FLAME [18] mesh with an open mouth:

$$P^{gc} = \{x_i^{gc}\}_{i=\{1\cdots N\}}, \tag{18}$$

where N is the number of points. The generical canonical points P^{gc} are shared start points for all subjects in the synthetic dataset. By deforming the points with subject-specific latent z as a condition, we obtain subject-specific canonical points P^{sc} containing the shape of a specific attribute, such as having long hair or grey cap. The mapping between two states is defined as an offset of each point $\mathcal{O}_i^{gc\rightarrow sc}$, which is regressed using coordinate-based deformation MLP as follows:

$$\{\mathcal{O}_i^{gc\rightarrow sc}, \mathcal{O}_i^{sc\rightarrow fc}, \mathcal{E}_i, \mathcal{P}_i, \mathcal{W}_i\} = \text{MLP}_d(\mathbf{z}, \mathbf{x}_i^{gc}). \tag{19}$$

It regresses FLAME LBS weight $\mathcal{W}_i$ and blendshapes $\{\mathcal{E}_i, \mathcal{P}_i\}$ of each point jointly, which is crucial to reenact our avatars into any novel pose and expression. Subsequently, our avatar model defines a mapping of subject-specific canonical points P^{sc} to the FLAME canonical points P^{fc} for better fidelity following the previous work [2, 35]. The mappings between two points are defined as another point offset $\mathcal{O}_i^{sc\rightarrow fc}$ which is also regressed by the deforming MLP jointly. The transformation between each state are summarized as follows:

$$\mathbf{x}_i^{sc} = \mathbf{x}_i^{gc} + \mathcal{O}_i^{gc\rightarrow sc}, \tag{20}$$
$$\mathbf{x}_i^{fc} = \mathbf{x}_i^{sc} + \mathcal{O}_i^{sc\rightarrow fc}. \tag{21}$$

Finally, the points in the FLAME-canonical space P^{fc} are deformed into the final posed space P^d using Linear Blend Skinning (LBS) and FLAME parameters $\{\beta, \theta, \psi\}$ as follows:

$$\mathbf{x}^{d-} = \mathbf{x}^{fc} + B_S(\boldsymbol{\beta}; \mathcal{S}) + B_P(\boldsymbol{\theta}; \mathcal{P}) + B_E(\boldsymbol{\psi}; \mathcal{E}) \tag{22}$$
$$\mathbf{x}^d = \text{LBS}(\mathbf{x}^{d-}, \mathbf{J}(\psi), \theta, \mathcal{W}), \tag{23}$$

where $\mathbf{x}^{d-}$ denotes the point after applying the blendshapes and before applying transformation via linear blend skinning.

Similar to PEGASUS [2], we infer the attributes of each Gaussian, $\mathbf{o}_i$ (opacity), $\mathbf{r}_i$ (rotation), $\mathbf{s}_i$ (scale), and $\mathbf{c}_i$ (color) using a coordinated-based MLP as follows:

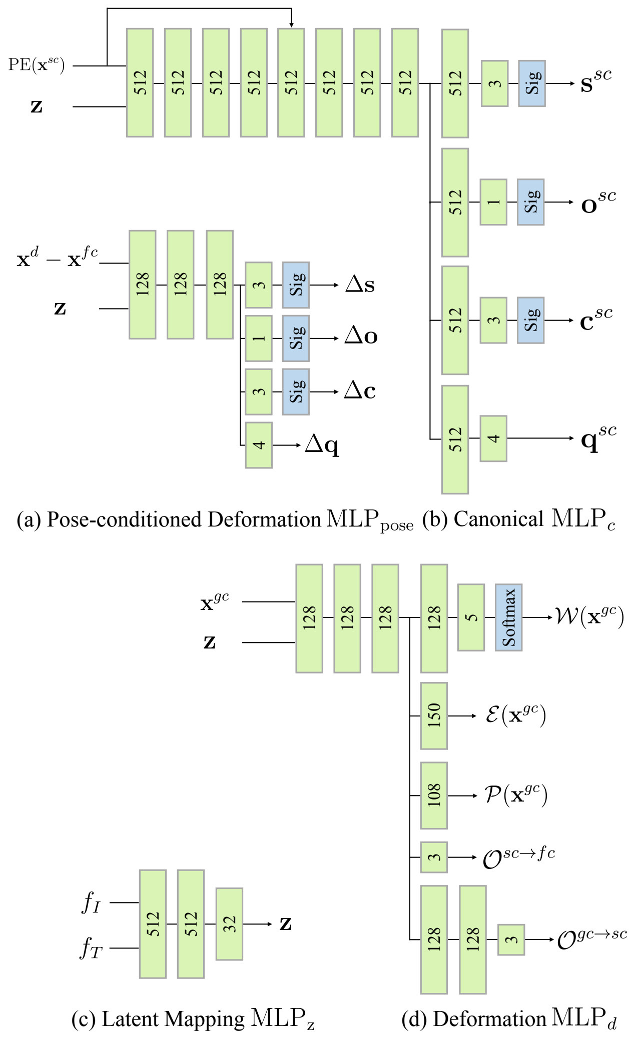

PE($\mathbf{x}^{sc}$)
$\mathbf{z}$
512
3
Sig
1
4
$\mathbf{s}^{sc}$
$\mathbf{o}^{sc}$
$\mathbf{c}^{sc}$
$\mathbf{q}^{sc}$
$\mathbf{x}^d - \mathbf{x}^{fc}$
128
$\Delta\mathbf{s}$
$\Delta\mathbf{o}$
$\Delta\mathbf{c}$
$\Delta\mathbf{q}$
(a) Pose-conditioned Deformation MLP_{pose}
(b) Canonical MLP_c
$\mathbf{x}^{gc}$
5
Softmax
150
108
$\mathcal{W}(\mathbf{x}^{gc})$
$\mathcal{E}(\mathbf{x}^{gc})$
$\mathcal{P}(\mathbf{x}^{gc})$
$\mathcal{O}^{sc\rightarrow fc}$
$\mathcal{O}^{gc\rightarrow sc}$
f_I
f_T
32
(c) Latent Mapping MLP_z
(d) Deformation MLP_d

Figure 12. **Network Configuration.** We show a detailed structure of the networks of our avatar model: pose-conditioned deformation MLP_{pose}, canonical MLP_{c}, latent mapping MLP_{z}, and deformation MLP_{d}.

$$\{\mathbf{o}_i^{sc}, \mathbf{r}_i^{sc}, \mathbf{s}_i^{sc}, \mathbf{c}_i^{sc}\} = \text{MLP}_c(\mathbf{z}, \mathbf{x}_i^{sc}). \tag{24}$$

This canonical MLP_c is defined against subject-specific canonical points and conditioned by latent code $\mathbf{z}$. We model additional 3D Gaussian change depending on the pose changes following MonoGaussianAvatar [3]. We calculate the deviation of each Gaussian center between before and after LBS deformation of (23) and query the change of each center to an MLP network together with latent $\mathbf{z}$ to estimate pose-conditioned deformation:

$$\Delta \mathbf{x}_i = \mathbf{x}_i^d - \mathbf{x}_i^{fc}, \tag{25}$$

$$\{\Delta \mathbf{r}_i, \Delta \mathbf{s}_i, \Delta \mathbf{o}_i, \Delta \mathbf{c}_i\} = \text{MLP}_{\text{pose}}(\Delta \mathbf{x}_i, \mathbf{z}). \tag{26}$$

We change all Gaussian parameters except the center $\mathbf{x}_i$. The final deformed Gaussians which are queried in the Gaussian Rasterizer [16] are as follows:

$$\mathbf{o}_i^d = \Delta \mathbf{o}_i + \mathbf{o}_i^{sc}, \tag{27}$$

$$\mathbf{s}_i^d = \Delta \mathbf{s}_i + \mathbf{s}_i^{sc}, \tag{28}$$

$$\mathbf{c}_i^d = \Delta \mathbf{c}_i + \mathbf{c}_i^{sc}, \tag{29}$$

$$\mathbf{r}_i^d = \Delta \mathbf{r}_i + \text{Rot}(\mathbf{r}_i^{sc}, \frac{\partial \mathbf{x}_i^d}{\partial \mathbf{x}_i^{fc}}), \tag{30}$$

where Rot$(\cdot)$ denotes multiplying a corresponding rotation $\frac{\partial \mathbf{x}_i^d}{\partial \mathbf{x}_i^{fc}}$ on each quaternion $\mathbf{r}_i^{sc}$ occurred during LBS of (23). The overall optimizable parameters of our avatar model are summarized below:

$$\Theta = \{\text{MLP}_c, \text{MLP}_d, \text{MLP}_z, \text{MLP}_{\text{pose}}, \{\mathbf{x}_i^{gc}\}_{i\in\{1\cdots N\}}\}. \tag{31}$$

The detailed network structure is shown in Fig. 12.

A.1.2. Training Strategy

We set the first epoch as a warm-up stage for stable optimization. During this stage, the pose-conditioned deformation MLP is disabled, and only the remaining MLPs and points are optimized. It encourages the deformation module of the avatar network to generate valid offsets from the generic canonical space to the final deformed space. We optimize our avatar model for 112 epochs using DDP with 8 A6000 GPUs, which takes around 2 days.

We follow prior work [3, 35] to iteratively densify the Gaussians via upsampling every 5 epochs until the number of points reaches 130,000. Once this target is reached, we reduce the length of the existing Gaussian attributes' 3D covariance by a factor of 0.75, and prune Gaussian attributes with opacity lower than 0.5 every 5 epochs. To maintain the point count at 130,000, we additionally upsample new Gaussian attributes with a fixed radius of 0.004.

A.1.3. Loss Functions

The FLAME loss [3, 35] included in total loss $\mathcal{L}_{tot}$ is regularization enforcing the inferred FLAME blendshapes and LBS weights $(\hat{\mathcal{E}}, \hat{\mathcal{P}}, \hat{\mathcal{W}})$ of each Gaussian to be close to the FLAME mesh's one:

$$\mathcal{L}_{\text{FLAME}} = \frac{1}{N}\sum_{i=1}^{N}(\lambda_e \|\mathcal{E}_i - \hat{\mathcal{E}}_i\|_2 + \lambda_p \|\mathcal{P}_i - \hat{\mathcal{P}}_i\|_2 + \lambda_w \|\mathcal{W}_i - \hat{\mathcal{W}}_i\|_2), \tag{32}$$

where $\mathcal{E}, \mathcal{P},$ and $\mathcal{W}$ are the pseudo ground truth from the k-nearest neighbor vertices of the FLAME [18]. This regularization is important to obtain better reenactment with unseen pose.

A.2. Finetuning for Interpolated Samples

A.2.1. Preliminaries: DiffMorpher

By viewing a diffusion sampling process as a solution of ODE, we obtain a deterministic mapping between a latent variable in the Gaussian distribution $\xi_T \in \mathcal{N}$ and an image $\mathbf{I}$ through DDIM forward and inversion [27]:

$$\xi = \text{DDIM}_{\text{inv}}(\mathbf{I}; \mathbf{W}),$$
$$\mathbf{I} = \text{DDIM}(\xi; \mathbf{W}),$$

where $\mathbf{W}$ means a pre-trained image diffusion model. By interpolating latents (ξ_a, ξ_b) inverted from two images $(\mathbf{I}_a, \mathbf{I}_b)$, we obtain semantically meaningful smooth interpolation as follows:

$$\xi_{interp,\alpha} = slerp(\xi_b, \xi_a, \alpha),$$
$$\mathbf{I}_{interp,\alpha} = \text{DDIM}(\xi_{interp,\alpha}; \mathbf{W}),$$

where α is an interpolation weight and $slerp(\cdot)$ is spherical linear interpolation [26].

DiffMorpher [32] uses personalized diffusion models for DDIM sampling and inversion, resulting in smoother and better natural image interpolation. For two images $(\mathbf{I}_a, \mathbf{I}_b)$, it trains LoRA [11] on UNet $(\Delta \mathbf{W}_a, \Delta \mathbf{W}_b)$ for each image and uses the LoRA-integrated UNet for DDIM inversion:

$$\xi_a = \text{DDIM}_{inv}(\mathbf{I}_a; \mathbf{W} + \Delta \mathbf{W}_a),$$
$$\xi_b = \text{DDIM}_{inv}(\mathbf{I}_b; \mathbf{W} + \Delta \mathbf{W}_b).$$

For the forward process on interpolated latent $\xi_{interp,\alpha}$, it uses interpolated LoRA with attention interpolation:

$$\mathbf{I}_{interp,\alpha} = \text{DDIM}(\xi_{interp,\alpha}; \Theta_{interp,\alpha}), \tag{33}$$

where $\mathbf{W}_{interp,\alpha}$ is an interpolated LoRA derived as $\mathbf{W}_{interp,\alpha} = \mathbf{W} + \Delta \mathbf{W}_a(1-\alpha) + \Delta \mathbf{W}_b \alpha$. For brevity, we denote the overall interpolation process with DiffMorpher from two images $(\mathbf{I}_a, \mathbf{I}_b)$ and a weight α as follows:

$$\mathbf{I}_{\alpha_i} = \text{DiffMorpher}_{\alpha_i}\Big(\mathbf{I}_a, \mathbf{I}_b\Big). \tag{34}$$

A.2.2. DiffMorpher LoRA Optimization

We use DiffMorpher [32] to generate interpolated images, which serve as pseudo ground truth to fine-tune our avatar model. Specifically, we select two subjects from the synthetic dataset and fine-tune the model for interpolated renderings between them. To obtain the corresponding pseudo ground truth images with DiffMorpher, we require a LoRA for each image.

Training a LoRA for each posed image is computationally prohibitive considering the number of images in our synthetic dataset. Therefore, unlike vanilla DiffMorpher [32], which uses a single image, we train LoRA subject-wise using all animated frames in each subject. The LoRA training objective is equal to the standard diffusion training objectives [24] as follows:

$$\mathcal{L}(\Delta\Theta) = \mathbb{E}_{\epsilon,\tau,i}[||\epsilon - \epsilon_{\Theta+\Delta\Theta}(\xi_{\tau i}, \tau, \mathbf{c}_i)||^2], \tag{35}$$

$$\xi_{\tau i} = \sqrt{\bar{\alpha}_\tau}\xi_{0i} + \sqrt{1-\bar{\alpha}_\tau}\epsilon, \tag{36}$$

where $\xi_{0i} = \mathcal{E}(\mathbf{I}_i)$ represents the latent encoded by the VAE encoder of diffusion model, $\mathbf{I}_i$ is the i^{th} animated image of the subject randomly selected at each iteration, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is Gaussian noise, and $\xi_{\tau i}$ denotes the perturbed latent at diffusion step τ. To avoid confusion with our model's latent variable $\mathbf{z}$, we use ξ to refer to the VAE-encoded latents of the diffusion model here. We train the subject-specific LoRA with batch size 8 for 5 epochs per subject.

A.2.3. Interpolation Loss Details

To enhance the quality of the interpolated sample and ensure interpolation smoothness, we calculate reconstruction loss on the interpolated samples. In every iteration, we randomly sample two subjects (a, b) from the same category of our synthetic dataset, referred to here as pivots. Then, we generate 5 interpolated samples using linear interpolation as follows:

$$\mathbf{z}_{\alpha,i} = \mathbf{z}_a(1-\alpha_i) + \mathbf{z}_b\alpha_i, \tag{37}$$

where $\{\alpha_i\}_{[i=1\cdots5]}$ are 5 equally distributed interpolation weights from $1/6$ to $5/6$. For all 5 interpolated samples, we compare the rendering with DiffMorpher [32] generated images as follows:

$$\hat{\mathbf{I}}_{\alpha_i} = \text{GSR}\Big(\mathcal{M}_\Theta(\mathbf{z}_{\alpha,i})\Big), \tag{38}$$

$$\mathbf{I}_{\alpha_i} = \text{DiffMorpher}_{\alpha_i}\Big(\mathbf{I}_a, \mathbf{I}_b\Big), \tag{39}$$

$$\mathcal{L}_{\text{interp}} = \sum_{i=1}^{5} \mathcal{L}_{\text{part}}\Big(\mathbf{M}_{\text{part}} \circ \mathbf{I}_{\alpha_i}, \mathbf{M}_{\text{part}} \circ \hat{\mathbf{I}}_{\alpha_i}\Big). \tag{40}$$

As the image $\mathbf{I}_\alpha$ generated by DiffMorpher [32] fails to preserve the identity of the remaining regions, we apply the loss only to the subpart region M_{part} is that changes during interpolation.

All DiffMorpher inferences and target part segmentations are performed online during optimization, as the number of possible pairs is too large to process in advance. We fine-tune our avatar model using an interpolation loss applied to 40 arbitrary pairs per subject, resulting in a total of 38,600 pairs. In each iteration, we also apply the total loss $\mathcal{L}_{tot}$ to the pivot subjects (a, b) to preserve their quality.

Category	# of attributes	w/ *portrait-Champ*
Hair	395	✓
Beard	69	✓
Cloth	57	-
Earrings	59	✓
Eyebrows	58	-
Headphones	59	✓
Hat	110	✓
Mouth	75	-
Nose	75	-
Total	957	-

Table 5. **Number of Attributes in Our Synthetic Dataset.** We use *portrait-Champ* to animate the portrait images when 'w/ *portrait-Champ* ' is indicated; otherwise, we use LivePortrait [9].

A.3. Synthetic Dataset

A.3.1. Attribute-Edited Portrait Image Generation

The number of attributes in each category is shown in Tab. 5. While we generate approximately $1k$ samples to demonstrate the effectiveness of PERSE, the pipeline can be extended to produce any desired amount, as our synthetic dataset generation process is fully automated. We use FLUX with inpainting controlnet [5] for Image-to-Image (I2I) inpainting and SDXL with pose controlnet [22, 33] for attribute mask generation.

A.3.2. Training *portrait-Champ*

Our *portrait-Champ* builds upon the architecture introduced in Champ [36], incorporating modifications to enhance 3D-awareness and improve reenactment performance. Specifically, we integrate an additional Variational Autoencoder (VAE) encoder-decoder pair dedicated to normal maps, drawing inspiration from MagicMan [10]. Adopting the dual-branch strategy proposed in MagicMan [10], we introduce an additional U-Net for the normal maps. This U-Net shares all weights with the original RGB U-Net except for the first layer. The shared layers between the two U-Nets enable cross-domain feature integration, allowing the model to fuse features from both normal map and RGB image. By combining geometric and visual information, our approach enhances the geometric awareness of model, resulting in improved structural coherence.

We replace the original SMPL [19] rendered motion guidance in vanilla Champ [36] with FLAME rendering. Specifically, we employ a monocular face capture method [6] to extract FLAME parameters [18]. Using these parameters, we render the FLAME depth map and FLAME normal map. To provide motion guidance for the body, including shoulders, which are not covered by FLAME rendering, we supplement the guidance with full-body keypoints and facial landmarks inferred from RGB videos using DWPose [30].

We use 5,196 videos from CelebV-Text [31] datasets to train our *portrait-Champ*. Following previous work [36], we train *portrait-Champ* using 8 A6000 GPUs in two stages:

58,732 iterations with a batch size of 32 in stage 1, and 26,450 iterations with a batch size of 8 in stage 2. In stage 1, we optimize the model using randomly sampled frames from videos as an image diffusion model. In stage 2, we train only the temporal motion module with videos while freezing other modules.

A.3.3. Animating Portrait Images

Enhancing the reenactment capability of our avatar model requires training videos that cover a wide range of facial expressions and head poses. We achieve this by animating portrait images with a motion sequence containing diverse expressions and poses. To obtain a motion sequence that satisfies both continuity and the minimal number of frames required by *portrait-Champ*, we record a video for this motion sequence ourselves.

Using a reference portrait image and a predefined motion sequence in an RGB video, we first generate an animated portrait video centered on the reference image using LivePortrait [9]. From this video, we extract normal maps, depth maps, and facial keypoint motion guidance using EMOCA [6] and DWPose [30]. With this guidance, we animate images edited in the hair, hat, and beard attributes using *portrait-Champ*. For other facial attributes, we directly generate RGB videos using LivePortrait [9].

A.4. Attribute Transfer

To transfer facial attributes from in-the-wild images, we incorporate LoRA layers [11] into the MLP network of the avatar model and optimize these layers. The LoRA layers are trained using animated videos generated from input in-the-wild images. We generate the animated videos following the procedure outlined in Sec. 4.2 of the main paper. To ensure only the desired attribute is transferred, we segment the relevant sub-part using an off-the-shelf segmentation network [17] and apply a part-wise loss as described in Eq. (15) of the main paper:

$$\mathcal{L}_{\text{partwise, lora}} = \mathcal{L}_{\text{recon}}\Big(\mathbf{M}_{\text{part}} \circ \mathbf{I}_{\text{itw}}, \mathbf{M}_{\text{part}} \circ \hat{\mathbf{I}}_{\text{attr}}\Big), \tag{41}$$

where I_{itw} represents the image from video animated in-the-wild portrait image, $\hat{I}_{\text{attr}}$ denotes the rendered image with latent $\mathbf{z}_{itw}$ regressed by latent mapping MLP_{z} from CLIP features of input in-the-wild image.

We observe that using only the partwise loss fails to preserve reference identity of our avatar model and collapse the pretrained latent space. To address this, we introduce a 3D loss. The 3D loss encourages the LoRA layers in the avatar model to produce the same output as when the LoRA layers are absent. Specifically, Gaussian random latent codes $\mathbf{z}_{\text{random}}$ from the pretrained latent space are sampled and used as inputs along with the FLAME parameters of an animatable portrait video. The model is trained to minimize the difference between the outputs of the avatar model with and without the LoRA layers, ensuring consistency in 3D Gaussian parameters and 3D positions. Specifically, for the Gaussian attributes inferred with and without LoRA layers:

$$\{\boldsymbol{x}_i^d, \boldsymbol{r}_i^d, \boldsymbol{s}_i^d, \boldsymbol{o}_i^d, \boldsymbol{c}_i^d\} = \mathcal{M}_{\Theta}(\mathbf{x}_i^{gc}, \mathbf{z}_{\text{random}}, \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{\psi}), \tag{42}$$

$$\begin{aligned}\{\boldsymbol{x}_{i,\text{lora}}^d, \boldsymbol{r}_{i,\text{lora}}^d, \boldsymbol{o}_{i,\text{lora}}^d, \boldsymbol{s}_{i,\text{lora}}^d, \boldsymbol{c}_{i,\text{lora}}^d\}& \\ = \mathcal{M}_{\Theta+\Delta\Theta}(\mathbf{x}_i^{gc}, \mathbf{z}_{\text{random}}, \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{\psi}),&\end{aligned} \tag{43}$$

we calculate the distance between them as follows:

$$\begin{aligned}\mathcal{L}_{3\text{d}} = \|\boldsymbol{x}_{i,\text{lora}}^d - \boldsymbol{x}_i^d\|_1 + \|\boldsymbol{r}_{i,\text{lora}}^d - \boldsymbol{r}_i^d\|_1 \\ + \|\boldsymbol{o}_{i,\text{lora}}^d - \boldsymbol{o}_i^d\|_1 + \|\boldsymbol{s}_{i,\text{lora}}^d - \boldsymbol{s}_i^d\|_1 + \|\boldsymbol{c}_{i,\text{lora}}^d - \boldsymbol{c}_i^d\|_1.\end{aligned} \tag{44}$$

The total loss for LoRA layer optimization is defined as follows:

$$\mathcal{L}_{\text{total, lora}} = \mathcal{L}_{3\text{d}} + \mathcal{L}_{\text{partwise, lora}} \tag{45}$$

We perform LoRA layer optimization with a learning rate of $1e^{-4}$ for 5 epochs.

B. Evaluation Details

B.1. Baseline Implementation Details

To demonstrate our pipeline's effectiveness, we evaluated our methods compared to three different methods.

B.1.1. PEGASUS

We train PEGASUS [2] with our synthetic dataset using publicly available code, strictly following the settings described in the paper, including the latent space configuration and network configurations. The model is trained using DDP across 8 RTX 6000 GPUs until convergence. After point rendering with PyTorch3D [23], no additional denoising steps are applied.

B.1.2. Conditional INSTA (Cond.TA)

To train INSTA with multiple subjects, we introduce a latent condition to the density MLP network, referred to as Conditional INSTA (Cond.TA). We adopt the PEGASUS [2] latent configuration to achieve similar sub-part disentangled control. Since the original density MLP network of INSTA is too small to encode a thousand of attributes, we increase the MLP width from 64 to 512 and the depth from 2 to 4. As this adjustment sacrifices rendering speed and increases training time, we focus our comparisons solely on quality, excluding rendering speed. The final Conditional INSTA model is trained using DDP with 8 RTX 4090 GPUs until convergence.

B.1.3. Conditional SplattingAvatar (Cond.SA)

Since SplattingAvatar [25] does not include any network for receiving conditioning, we incorporate an MLP to deform a single set of shared canonical 3D Gaussians into subject-specific canonical 3D Gaussians, similar to the approach in

PEGASUS [2]. To ensure a fair comparison, we configure the MLP with the same size as PEGASUS's canonical MLP, providing sufficient capacity to represent all subjects in the synthetic dataset. The densification interval is increased from vanilla SplattingAvatar [25] to address the low stability of optimization in early stages. Densification is halted after 5 epochs, as the gathered gradients do not converge, possibly due to exposure to different subjects in each iteration. We adopt the same latent configuration as the PEGASUS model, and the final Conditional SplattingAvatar model is trained using DDP on 8 RTX 4090 GPUs until convergence.

B.2. Interpolation Evaluation Details

To evaluate the rendering quality of avatars with unseen attributes and interpolation smoothness, we sample avatars from our model using interpolated latent codes. For each of the 9 categories in our synthetic dataset, we randomly select 200 subject pairs and generate 9 interpolated latent codes per pair, following (37). The intervals between the sampled latent codes are evenly spaced. Each interpolated latent code is used to render the corresponding avatar in 5 different poses. This process produces 9,000 images per category and a total of 81,000 images across all categories for evaluation.

Metrics. We compute FID and KID scores by comparing our renderings with two different image sets: FFHQ [14] and our synthetic evaluation dataset, which is built with the same input reference individual. Specifically, we use $(\text{FID}_{\text{FFHQ}}, \text{KID}_{\text{FFHQ}})$ to asses the realism and quality of the renderings by comparing with real face images, and $(\text{FID}_{\text{SYN}}, \text{KID}_{\text{SYN}})$ to evaluate identity preservation by comparison with the synthetic evaluation dataset.

Since the rendered outputs do not include backgrounds, we remove the backgrounds of all portrait images in FFHQ using MODNet [15] before calculating metrics. The synthetic evaluation image sets are constructed with the same reference image, following our edited portraits generation pipeline. To prevent potential information leaks, we synthesize $2k$ novel images using text prompts not included in the training dataset. This approach provides a more reliable measurement of identity preservation during attribute editing, particularly for changes that partially alter identity features, such as the eyes, eyebrows, and nose, which are challenging to evaluate with existing identity metrics like ArcFace [7].

B.3. User Studies

We also conduct a user study to evaluate the rendering quality of interpolated samples, as shown in Fig. 22. Since only PEGASUS [2] and our method receive votes among the four methods in preliminary study, we exclude *Cond.TA* and *Cond.SA* from the options. Participants are asked to choose the better images based on interpolation smoothness and image quality for 20 pairs of interpolations. The pairs are randomly selected from the hair category. We collect responses from 229 participants via CloudResearch [4].

Method	Accuracy			Naturalness	
	PSNR↑	SSIM↑	LPIPS↓	FID↓	KID↓
Moore-AnimateAnyone [12, 20]	17.77	0.6841	*0.2536*	**146.59**	**0.0530**
MimicMotion [34]	17.27	0.6641	0.3012	178.87	0.0980
MegActor-Σ [28, 29]	*17.89*	*0.6986*	0.2599	155.04	0.0572
Ours (*portrait-Champ*)	**20.58**	**0.7417**	**0.1878**	*150.59*	*0.0555*

Table 6. **Quantitative Comparisons for Image-to-Video Models.** We evaluate our *portrait-Champ* with recent diffusion based baselines in face reenactment scenarios. Ours *portrait-Champ* obtain the best scores in accuracy and comparable FID and KID.

Input Type	2D Video	Rendering Quality			
	Subject Consistency ↑	PSNR↑	SSIM↑	LPIPS↓	Imaging Quality↑
Real Video	**0.9761**	**22.26**	0.9045	**0.1352**	0.5366
Synthetic Video $_{\text{self-driving}}$	0.9719	21.23	**0.9241**	0.1582	**0.5896**

Table 7. **Quantitative Comparison of Impact of Inconsistency.** Quantitative comparison of PERSE avatar models trained on real and synthetic videos. Note that 2D video evaluated for subject consistency is used for training, and rendering quality is evaluated on unseen head poses and facial expressions using a test sequence.

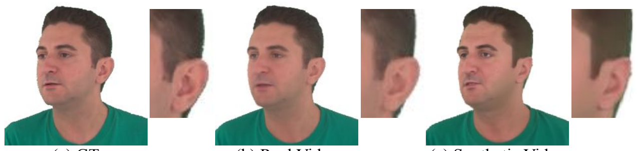

(a) GT (b) Real Video (c) Synthetic Video

Figure 13. **Qualitative Comparison of Impact of Inconsistency.** We show qualitative comparision of impact of inconsistency between real and 2D generated video.

C. More Experiments

C.1. Additional Results

We present additional sample results of attribute-edited portrait image generation, providing seven results for each attribute in Fig. 14 and Fig. 15. Furthermore, we demonstrate the rendering results of our personalized 3D generative avatar on unseen poses, trained with synthetic datasets created using additional portrait images in Fig. 16, Fig. 17, Fig. 18, Fig. 19, and Fig. 20. Finally, we provide the interpolation results between two latent codes for each attribute in Fig. 21.

C.2. Impact of Video Inconsistency

The different between real and generated 2D video is negligible, as monocular avatar-building pipeline handles temporal deformations and inconsistencies. To assess this, we present evaluations by building a 3D avatar from each single video, as demonstrated in Tab. 7 and Fig. 13. We measure subject consistency and imaging quality following VBench [13], comparing real video and generated video from *portrait-*

Champ by animating the first frame in a self-driven manner, where they show minor differences. After building 3D avatars from each 2D real and generated video separately, we also compare the rendering quality under novel head poses and facial expressions. As shown in Tab. 7 and Fig. 13, the avatar renderings also show negligible differences in quality, with comparable PSNR, LPIPS, and SSIM scores.

C.3. Synthetic Monocular Dataset Generation from Single Image

To demonstrate the effectivness of our *portrait-Champ*, we evaluate the reconstruction quality and rendering realism compared to diffusion based baselines. Moore AnimateAnyone [20] is open-source repository fine-tuned AnymateAnyone [12] to be specialized on facial reenactment. MimicMotion [34] is a full body animating model based on Stable Video Diffusion [1] also capable of reenactment using facial landmarks in DWPose. MegActor-Σ is Diffusion Transformer [21] based approach to solve reenactement problem. We disable the additional audio input option of MegActor-Σ during test here.

We test the methods using 20 sequence randomly selected from CelebV-Text dataset [31] not seen during the trainining. We animate the first frame to make other frames and compare with ground truth frames in the video to compute accuracy. We additionally calculate FID and KID against FFHQ dataset [14] to evaluate the naturaless of the animated images. As shown in Tab. 6, our approach achieves the highest reconstruction score across all metrics compared to previous SOTA methods.

D. Rights

All portrait reference images used in this work are sourced from the *FreePik* [8] website under a free license. Note that all of our portraits to show our results are not AI-generated images. Our code and samples of synthetic datasets are publicly released for research purposes only. For more details, refer to `https://github.com/snuvclab/perse` about our implementations.

E. Notations

Refer to Tab. 8 for an overview of the notations used in this paper.

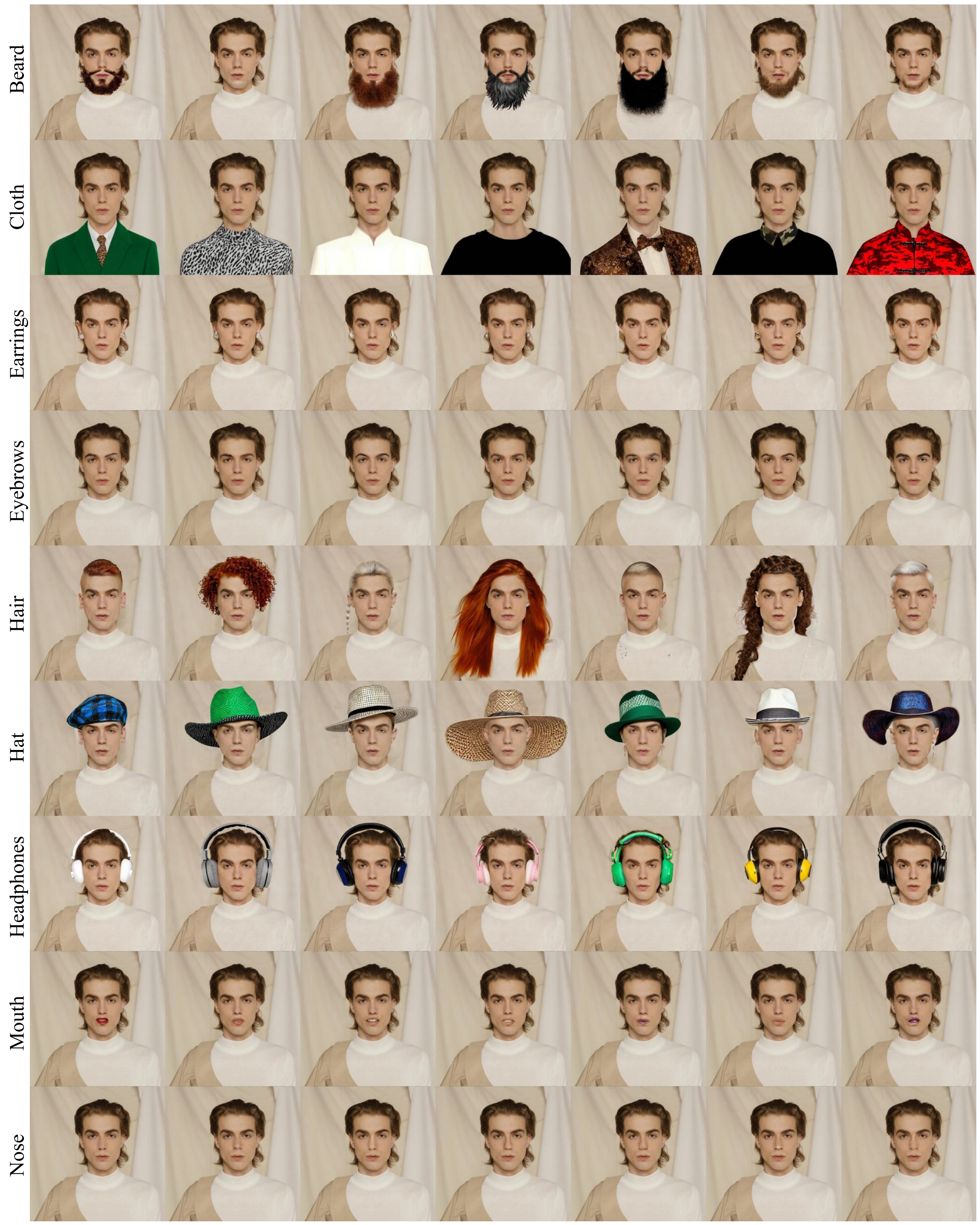


Figure 14. **Example of Attribute-Edited Portrait Image Generation (1).** We present samples of attribute-edited portrait image generation. For each attribute, we display results obtained through random sampling.

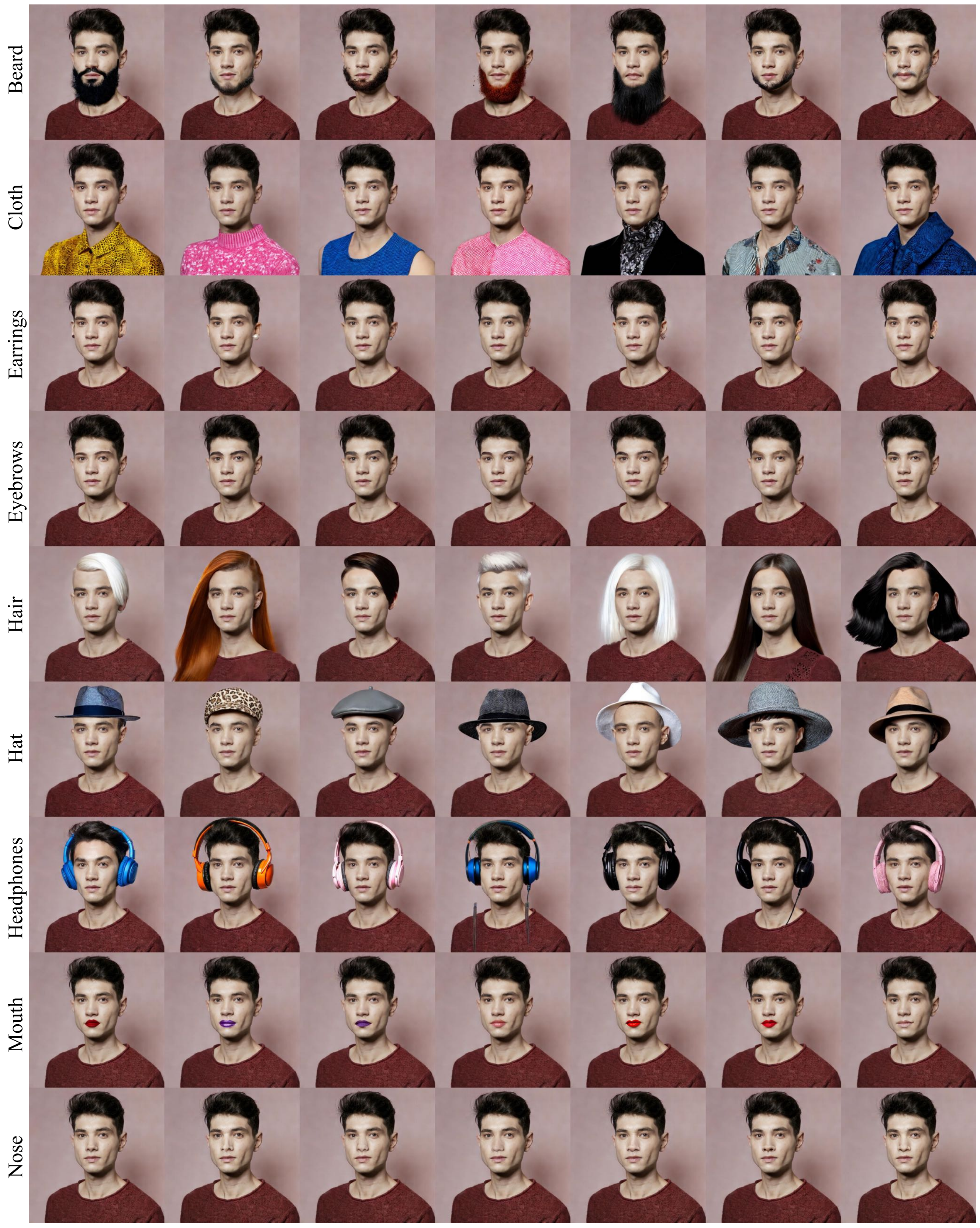


Figure 15. **Example of Attribute-Edited Portrait Image Generation (2).** Our method can be applied to various portrait images

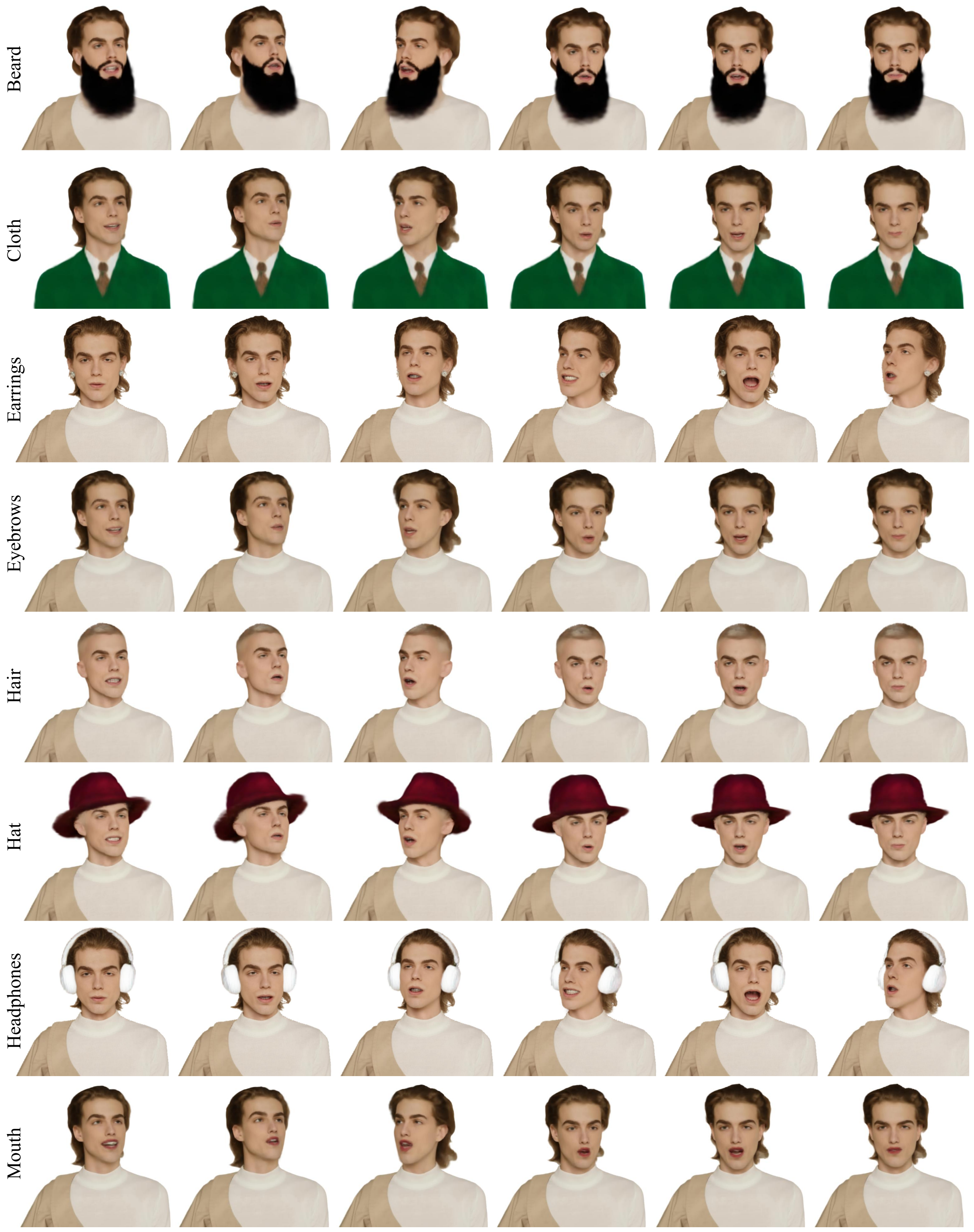


Figure 16. **Unseen Pose Rendering Results (1).** We present the rendering results using latent codes for novel head poses and facial expressions not included in the training dataset, categorized by each attribute.

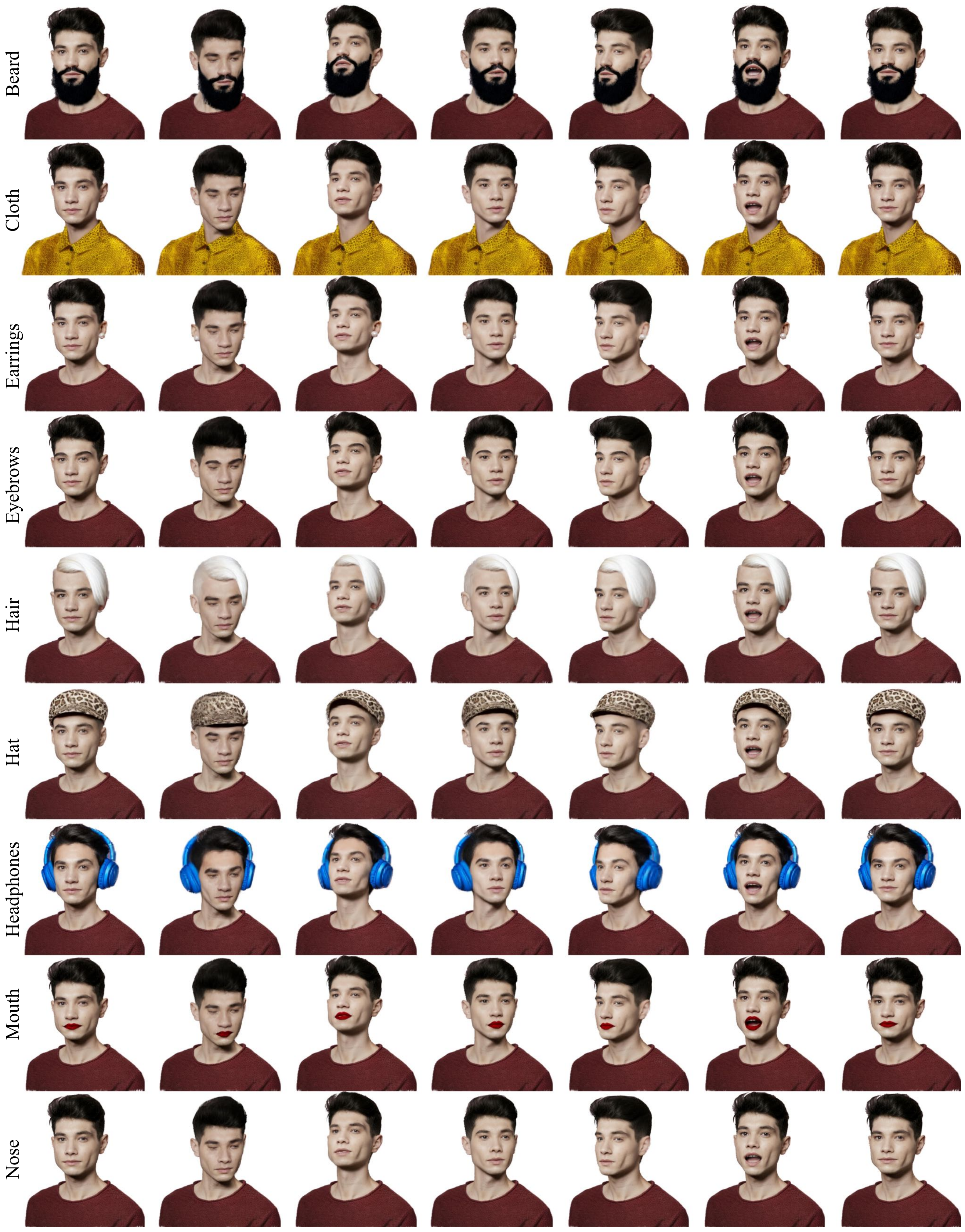


Figure 17. **Unseen Pose Rendering Results (2).**

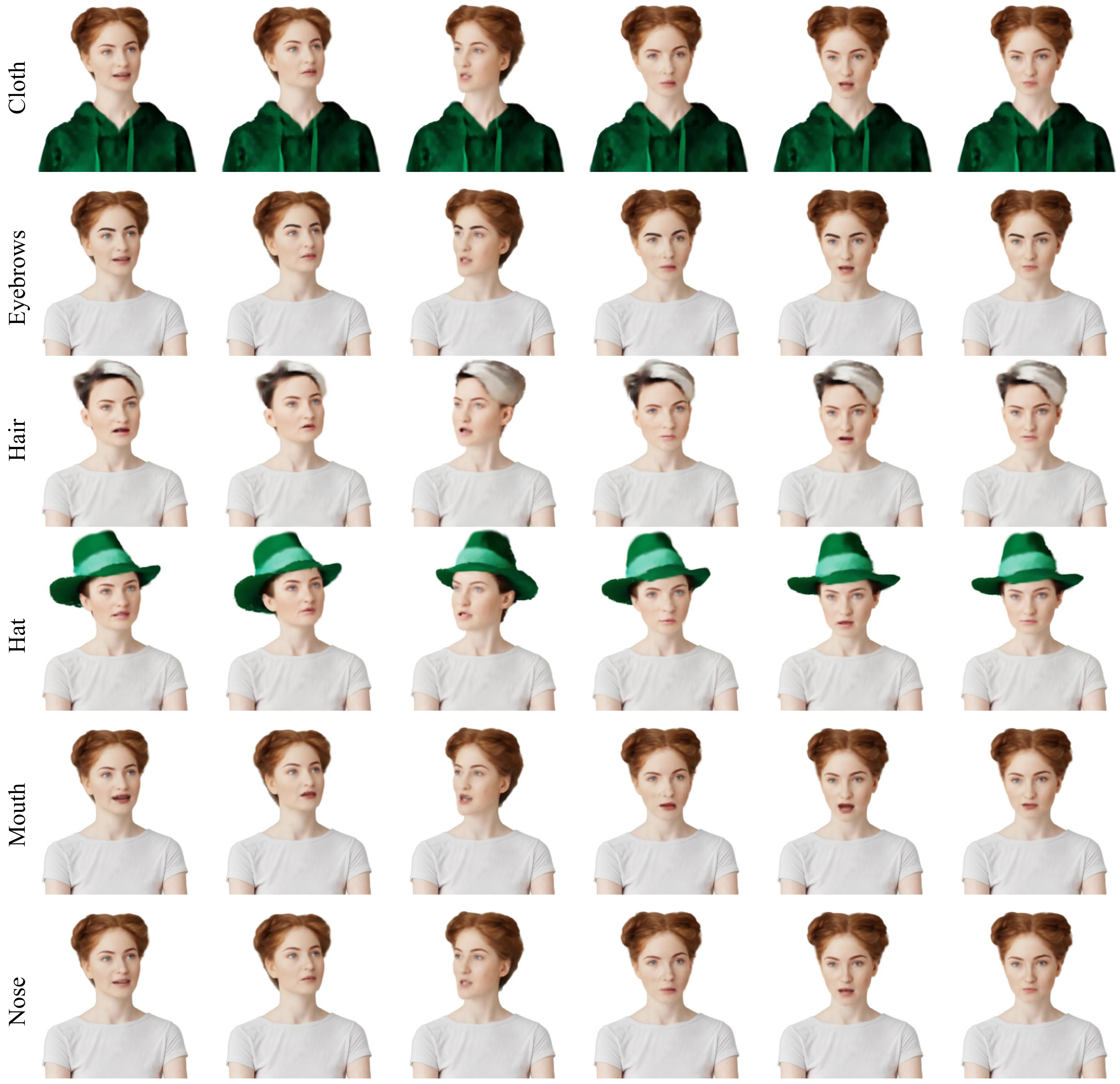


Figure 18. **Unseen Pose Rendering Results (3).**

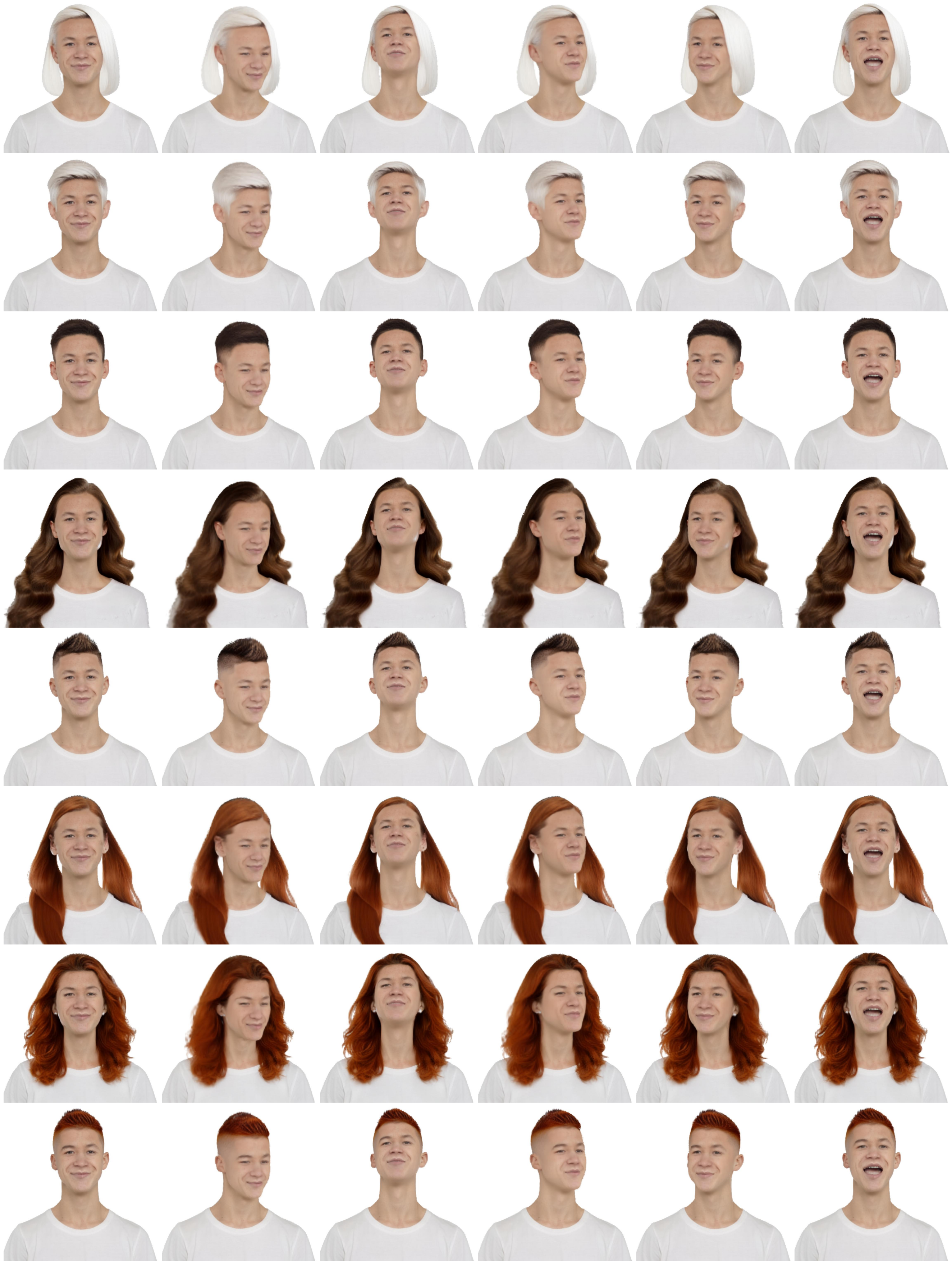


Figure 19. **Unseen Pose Rendering Results (4).** We show hair-only rendering results for unseen poses.

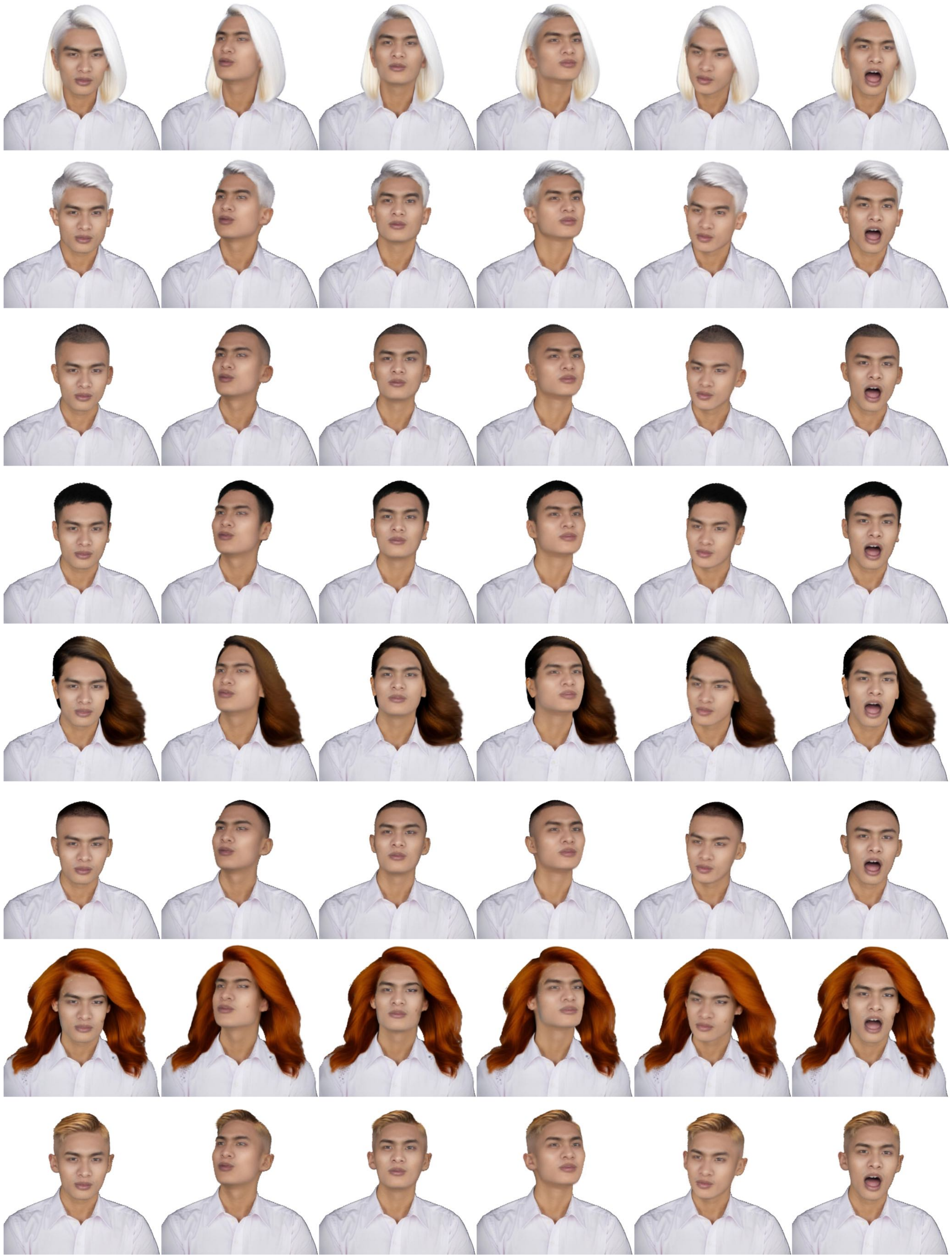


Figure 20. **Unseen Pose Rendering Results (5).** We show hair-only rendering results for unseen poses.

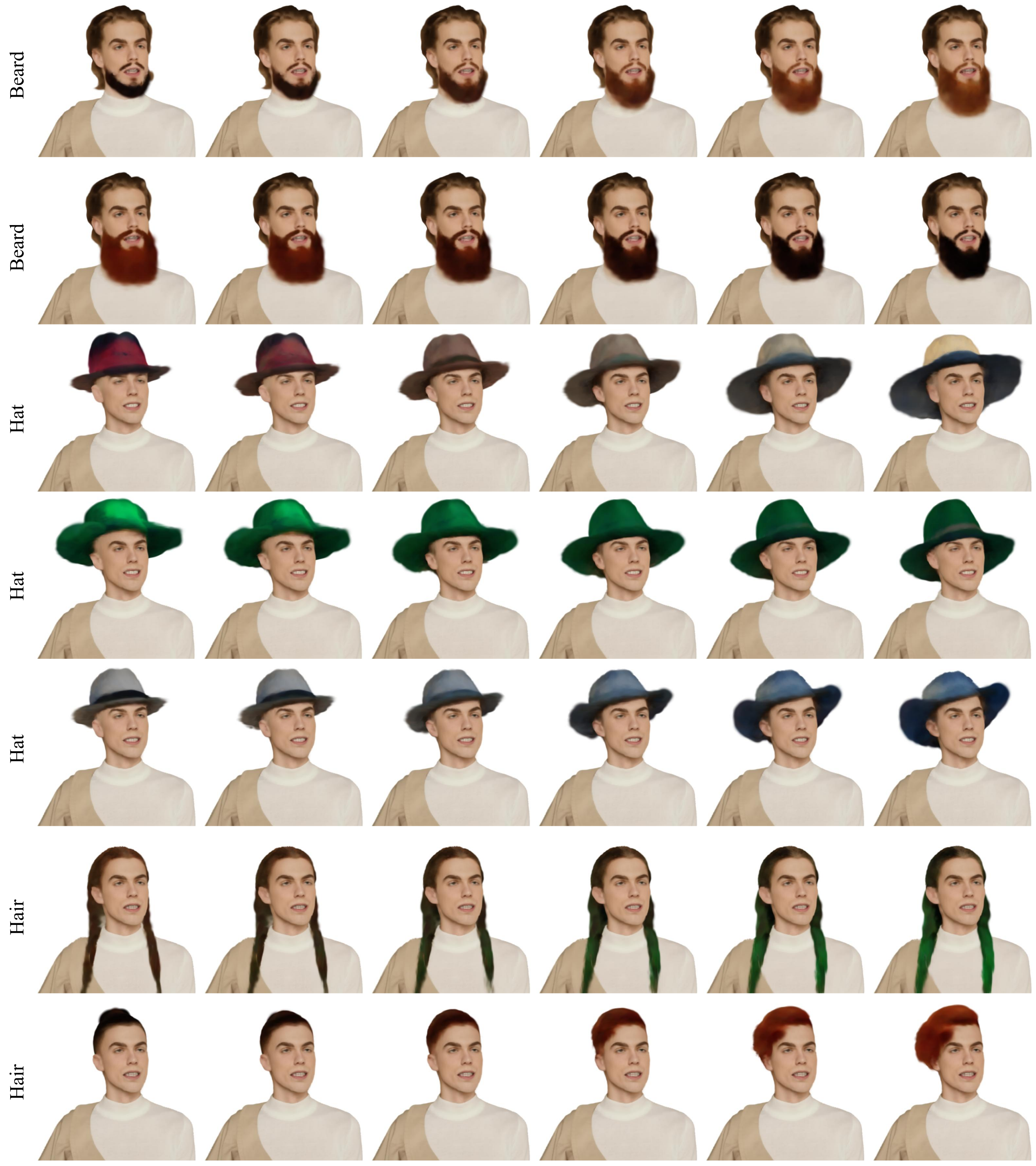
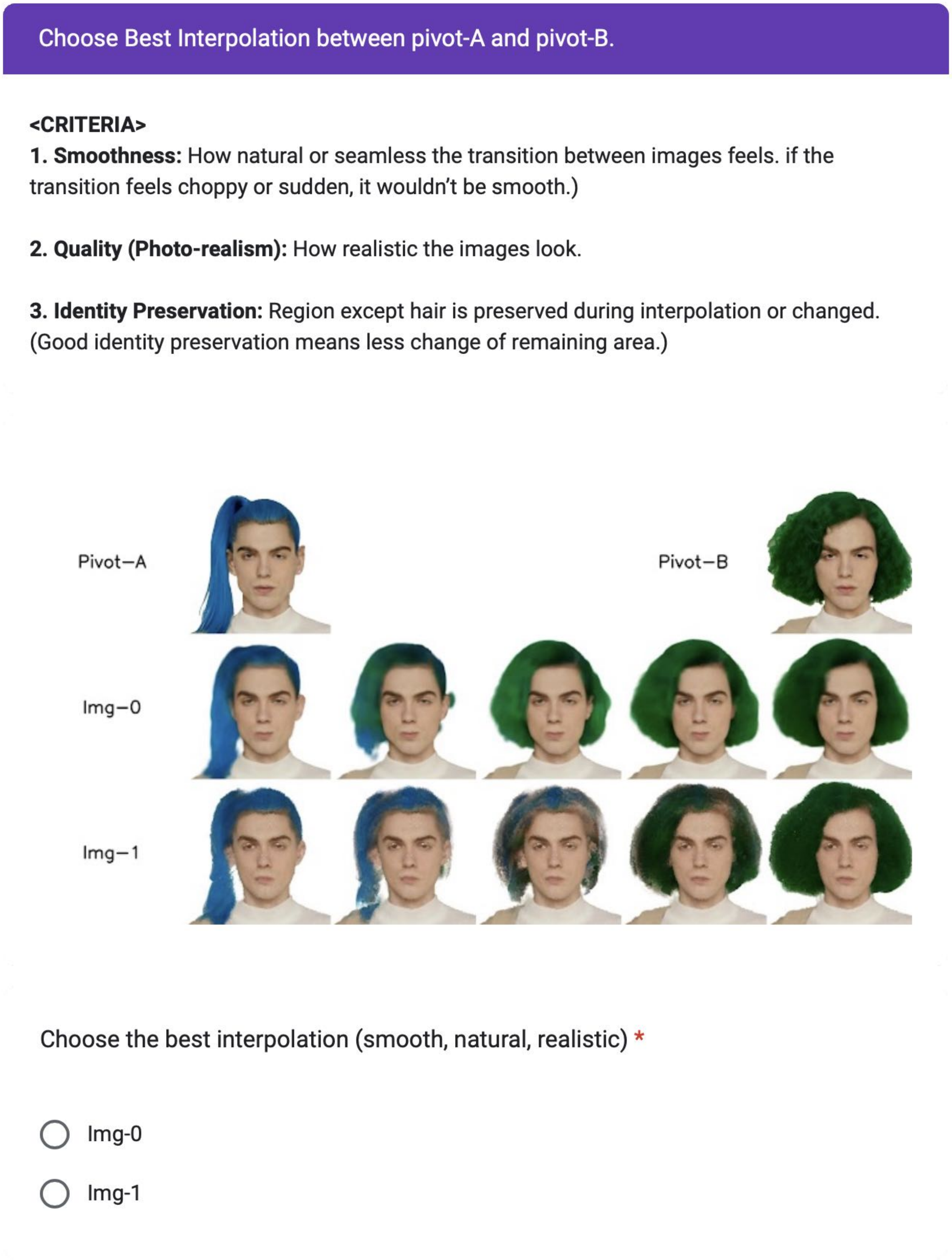


Figure 21. **Interpolation Between Two Latent Codes.** We present the rendering results obtained by interpolating between two latent codes.



Choose Best Interpolation between pivot-A and pivot-B.

<CRITERIA>

1. Smoothness: How natural or seamless the transition between images feels. if the transition feels choppy or sudden, it wouldn't be smooth.)

2. Quality (Photo-realism): How realistic the images look.

3. Identity Preservation: Region except hair is preserved during interpolation or changed. (Good identity preservation means less change of remaining area.)

Choose the best interpolation (smooth, natural, realistic) *

○ Img-0

○ Img-1

Figure 22. **User Study.** We show user study screenshot.

Table 8. Table of notations.

Symbol	Description
	Index
i	Gaussian index $i \in \{1, \ldots, N\}$ in 3D Gaussian attributes
j	Category index $j \in \{1, \ldots, N_c\}$ of edited attributes in synthetic dataset.
	Learnable Parameters and Networks
MLP_c	Canonical MLP estimating attributes of 3D Gaussians
MLP_d	Deformation MLP estimating deformation attributes
MLP_{pose}	Pose-conditioned deformation MLP estimating change of Gaussian attributes
MLP_z	Latent mapping MLP from CLIP feature f_I, f_T to subject-specific latent z
$P^{gc} = \{x_i^{gc}\}_{i=\{1 \cdots N\}}$	Learnable positions of 3D Gaussians
	Spaces of our Avatar Model
P^{gc}	Generic canonical space, single space shared on all subject
P^{sc}	Subject-specific canonical space, conditioned by subject latent $\boldsymbol{z}$
P^{fc}	FLAME-canonical space, deformed from subject-specific canonical space with blendshape
P^{d}	Deformed space, deforming P^{fc} with FLAME pose parameters
	Diffusion Related
T	Text-prompt queried into the diffusion model
$C(\cdot)$	2D key points and face landmarks estimator and renderer (OpenPose)
τ	Diffusion denoising time-step
ξ_0	Encoded latent of the queried RGB images of diffusion model
ξ_τ	Perturbed latent with noise time-step $\tau \in [0, 1]$
ϵ	Noise added to the latent
	Attributes of 3D Gaussians
$\boldsymbol{x}_i \in \mathbb{R}^3$	Center of i-th Gaussian, or point position in PEGASUS
$\boldsymbol{q}_i \in \mathbb{R}^4$	Covariance Matrix's Quaternion of i-th Gaussian
$\boldsymbol{s}_i \in \mathbb{R}^3$	Covariance Matrix's Scale Component of i-th Gaussian
$\boldsymbol{c}_i \in \mathbb{R}^3$	Color of i-th Gaussian
$\boldsymbol{o}_i \in \mathbb{R}$	Opacity of i-th Gaussian
	Off-the-Shelf Network
$\text{I2I}_\text{inpaint}$	Text-conditioned Image-to-Image inpainting pipeline, based on image diffusion
T2I	Text-to-Image diffusion model
I2V	Portrait animating Image-to-Video model, *portrait-Champ* or LivePortrait [9].
	FLAME Parameters of Avatar Deformation
$\boldsymbol{\theta} \in \mathbb{R}^{15}$	FLAME pose parameter
$\boldsymbol{\beta} \in \mathbb{R}^{100}$	FLAME shape parameters
$\boldsymbol{\psi} \in \mathbb{R}^{50}$	FLAME expression parameters
$\mathcal{E} \in \mathbb{R}^{50 \times 5023}$	FLAME expression blendshape parameters, estimated by MLP_d for each Gaussian
$\mathcal{P} \in \mathbb{R}^{100 \times 5023}$	FLAME shape blendshape parameters, estimated by MLP_d for each Gaussian
$\mathcal{W} \in \mathbb{R}^{15 \times 5023}$	FLAME Linear Blend Skinning (LBS) weight, estimated by MLP_d for each Gaussian
	Rendered and Observed Images
$\hat{\mathbf{I}}/\mathbf{I}$	Rendered / Ground Truth Image
$\mathbf{M}$	Mask of subpart region

References

[1] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 6

[2] H. Cha, B. Kim, and H. Joo. Pegasus: Personalized generative 3d avatars with composable attributes. In *CVPR*, 2024. 1, 4, 5

[3] Y. Chen, L. Wang, Q. Li, H. Xiao, S. Zhang, H. Yao, and Y. Liu. Monogaussianavatar: Monocular gaussian point-based head avatar. In *SIGGRAPH*, 2024. 1, 2

[4] CloudResearch. Connect cloud research. URL `https://connect.cloudresearch.com/researcher/`. 5

[5] A. Creative. Flux.1-dev-controlnet-inpainting-alpha. `https://huggingface.co/alimama-creative/FLUX.1-dev-Controlnet-Inpainting-Alpha`, 2024. 3

[6] R. Daněček, M. J. Black, and T. Bolkart. Emoca: Emotion driven monocular face capture and animation. In *CVPR*, 2022. 3, 4

[7] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019. 5

[8] Freepik. Portrait search results. `https://www.freepik.com/search?ai=excluded&query=Portrait`. Accessed: 2024-11-22. 6

[9] J. Guo, D. Zhang, X. Liu, Z. Zhong, Y. Zhang, P. Wan, and D. Zhang. Liveportrait: Efficient portrait animation with stitching and retargeting control. *arXiv preprint arXiv:2407.03168*, 2024. 3, 4, 16

[10] X. He, X. Li, D. Kang, J. Ye, C. Zhang, L. Chen, X. Gao, H. Zhang, Z. Wu, and H. Zhuang. Magicman: Generative novel view synthesis of humans with 3d-aware diffusion and iterative refinement. *arXiv preprint arXiv:2408.14211*, 2024. 3

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 2, 4

[12] L. Hu. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. In *CVPR*, 2024. 5, 6

[13] Z. Huang, Y. He, J. Yu, F. Zhang, C. Si, Y. Jiang, Y. Zhang, T. Wu, Q. Jin, N. Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024. 5

[14] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 5, 6

[15] Z. Ke, J. Sun, K. Li, Q. Yan, and R. W. Lau. Modnet: Real-time trimap-free portrait matting via objective decomposition. In *AAAI*, 2022. 5

[16] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 2023. 2

[17] R. Khirodkar, T. Bagautdinov, J. Martinez, S. Zhaoen, A. James, P. Selednik, S. Anderson, and S. Saito. Sapiens: Foundation for human vision models. In *ECCV*, 2025. 4

[18] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4d scans. *ACM Trans. Graph.*, 2017. 1, 2, 3

[19] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *SIGGRAPH ASIA*, 2015. 3

[20] MooreThreads. Moore-animateanyone. `https://github.com/MooreThreads/Moore-AnimateAnyone`, 2023. 5, 6

[21] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 6

[22] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 3

[23] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 4

[24] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 3

[25] Z. Shao, Z. Wang, Z. Li, D. Wang, X. Lin, Y. Zhang, M. Fan, and Z. Wang. Splattingavatar: Realistic real-time human avatars with mesh-embedded gaussian splatting. In *CVPR*, 2024. 4, 5

[26] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH*, 1985. 2

[27] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *Proc. ICLR*, 2020. 2

[28] S. Yang, H. Li, J. Wu, M. Jing, L. Li, R. Ji, J. Liang, and H. Fan. Megactor: Harness the power of raw video for vivid portrait animation. *arXiv preprint arXiv:2405.20851*, 2024. 5

[29] S. Yang, H. Li, J. Wu, M. Jing, L. Li, R. Ji, J. Liang, H. Fan, and J. Wang. Megactor-σ: Unlocking flexible mixed-modal control in portrait animation with diffusion transformer. *arXiv preprint arXiv:2408.14975*, 2024. 5

[30] Z. Yang, A. Zeng, C. Yuan, and Y. Li. Effective whole-body pose estimation with two-stages distillation. In *ICCV*, 2023. 3, 4

[31] J. Yu, H. Zhu, L. Jiang, C. C. Loy, W. Cai, and W. Wu. Celebv-text: A large-scale facial text-video dataset. In *CVPR*, 2023. 3, 6

[32] K. Zhang, Y. Zhou, X. Xu, B. Dai, and X. Pan. Diffmorpher: Unleashing the capability of diffusion models for image morphing. In *CVPR*, 2024. 2, 3

[33] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 3

[34] Y. Zhang, J. Gu, L.-W. Wang, H. Wang, J. Cheng, Y. Zhu, and F. Zou. Mimicmotion: High-quality human motion video generation with confidence-aware pose guidance. *arXiv preprint arXiv:2406.19680*, 2024. 5, 6

[35] Y. Zheng, W. Yifan, G. Wetzstein, M. J. Black, and O. Hilliges. Pointavatar: Deformable point-based head avatars from videos. In *CVPR*, 2023. 1, 2

[36] S. Zhu, J. L. Chen, Z. Dai, Y. Xu, X. Cao, Y. Yao, H. Zhu, and S. Zhu. Champ: Controllable and consistent human image animation with 3d parametric guidance. *arXiv preprint*

arXiv:2403.14781, 2024. 3