Supplemental Material
PEGASUS: Personalized Generative 3D Avatars with Composable Attributes

Hyunsoo Cha Byungjun Kim Hanbyul Joo
Seoul National University
243stephen@snu.ac.kr byungjun.kim@snu.ac.kr hbjoo@snu.ac.kr
https://snuvclab.github.io/pegasus/

A. Synthetic DB Generation

In this section, we provide further details of our synthetic database (DB) generation via part swapping, introduced in Sec. 4.2 of our main manuscript.

Hair. We empirically find that removing the hair of the target subject is necessary before swapping the hair from the attribute DB. To create a bald head representation of the target individual, we utilize the Stable Diffusion [13], employing auto-generated mask images for this purpose. To generate the hair mask, we utilize an off-the-shelf face parsing network [17, 21]. We dilate the mask image using a kernel of size 20 from OpenCV [1]. Then, to generate an image of the target person with a bald head, we employ Stable Diffusion in conjunction with ControlNet [18]. The prompt to generate the bald head is "bald, clean skin, smooth bald, small head, albedo." The negative prompt is "hair, wrinkles, shadow, light reflection, tattoo, sideburns, facial hair, cartoonish, abstract interpretations, hat, head coverings." The examples are shown in Fig. 1.

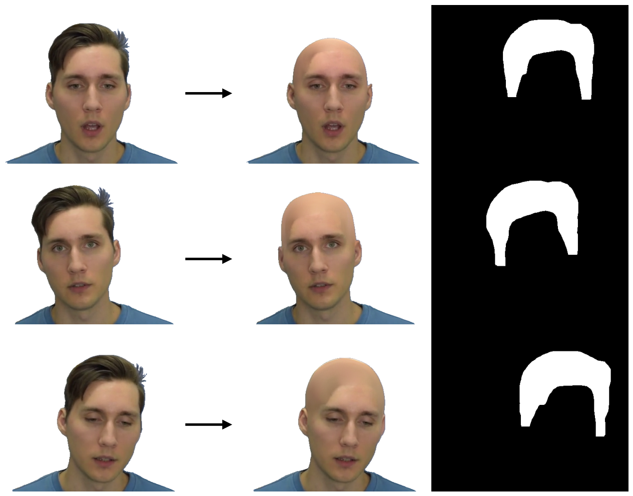


Figure 1. **Stable Diffusion Inpainting.** We leverage Stable Diffusion [13] and ControlNet [18] to remove the target's hair and make it bald, in order to synthesize different hair. The automatically generated mask images represent the area designated for inpainting.

Other Attributes. Our goal is to synthesize the shape and appearance of the facial attribute from the attribute DB into the target individual as seamlessly as possible. To achieve this, we first render the avatar from an attribute DB into the same view, shape, and facial expressions as the target frame of the target individual's video, as described in Sec. 4.2 in our main manuscript. Subsequently, we acquire the mask of the rendered facial attribute by employing a face parsing network [17, 21] and then slightly enlarge it by applying the dilate function in OpenCV. We also perform the segmentation for the target individual's image to acquire the mask of the target facial attribute by utilizing the face parsing network [17, 21], where the target facial part is subsequently "removed" via inpainting by employing the Fast Marching Method [15]. This process can be considered as a similar process of "bald head synthesis" before integrating the desired facial part from the attribute source. Finally, we seamlessly integrate the facial attribute from the attribute avatar into the target individual using Poisson blending [11]. Examples of nose and mouth synthesis employing this technique are illustrated in Fig. 2.

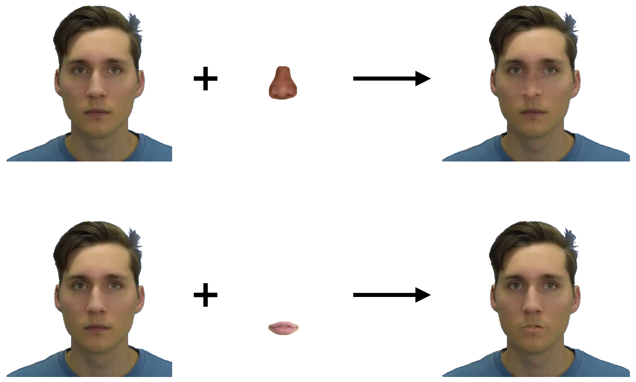


Figure 2. **Poisson Blending Inpainting.** We use Poisson blending [11] to synthesize the facial attribute and the target's face.

Tracking and Masking. To extract FLAME parameters from images, along with their corresponding camera parameters, we utilize the DECA model [3]. When FLAME parameters are directly extracted using the DECA model, we notice that the head pose estimation is noisy and jittery, particularly in the frames where the eyes in the original images are blinking. To improve the FLAME parameter estimation quality, following the similar process of PointAvatar [20], we apply an optimization procedure to align the 2D projection of FLAME's facial landmarks with the detection outputs of an off-the-shelf 2D facial landmark detector [2]. This optimization process is based on the assumption that the quality of the 2D landmark detection is more precise. We minimize the point-wise distance between the landmark obtained from FLAME and the 2D facial landmark to optimize the shape, pose, and camera parameters. Different from PointAvatar's approach, instead of using a singular translation vector for each video, we employ a unique vector for every image frame in scenarios involving in-the-wild video tracking.

To create the foreground mask image, we leverage an off-the-shelf background matting network [5] to obtain the portrait mask images from the videos. We use the face parsing network [17, 21] to obtain part segmentations of the faces and leverage SegmentAnything Model [7, 9] for segmenting head accessories.

B. Postprocessing of Zero-Shot Transfer

We provide further details of the Eq. (13) in our main manuscript, which is the process of combining the subsets of point clouds from both avatars. In short, the zero-shot process is performed via three steps: (1) naive composition after segmentation by introducing additional point clouds for the missing region; (2) optimization by aligning facial landmarks for better alignment; and (3) color blending for the added points for seamless outputs.

Obtaining the Additional Part from the Source Human. We use the estimated segmentation masks of the face attribute χ_ϕ and χ_{th} that can be controlled via latent code $\mathbf{z}$ to select the *target human*'s point cloud except for the facial attribute $\chi_{th} = 0$ and *source human*'s point cloud that includes the facial attribute $\chi_\phi = 1$:

$$P_{\text{naive}} = \{\mathbf{x}_\phi^{d,i}\}_{\chi_\phi^i=1} \cup \{\mathbf{x}_{th}^{d,i}\}_{\chi_{th}^i=0} \tag{1}$$

When we remove the facial attribute from the *target human* and bring in the facial attribute from the *source human*, it creates an empty space between the two point clouds. To fill this missing region, as shown in Fig. 4a, we bring in additional parts from the *source human*. Formally, this can be represented as follows:

$$P_{\text{naive w/ add}} = P_{\text{naive}} \cup \{\mathbf{x}_\phi^{d,i}\}_{\chi_{\phi,\text{add}}^i=1} \tag{2}$$

To create the additional segmentation mask $\chi_{\phi,\text{ add}}^i$, we borrow the knowledge from the FLAME [8] by leveraging

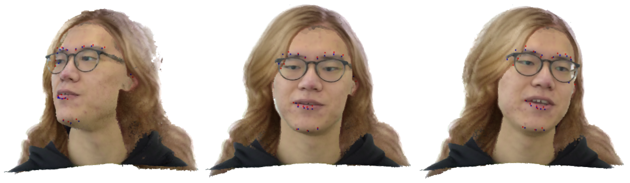
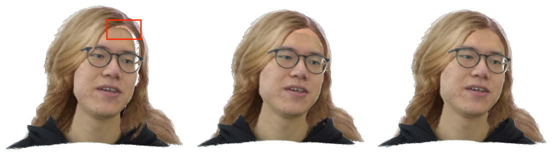


Figure 3. **Zero-Shot Landmarks for Optimization.** The red dot represents our personalized generative model's k-nearest neighbor of 3D Landmarks from FLAME keypoints, and the blue dot represents the target's k-nearest neighbor of 3D Landmarks from FLAME keypoints.



(a) Before Optimize. (b) After Optimize. (c) Color Blending.

Figure 4. **Zero-Shot Optimization Steps.** The red box represents the additional part to fill the empty space. Through zero-shot modeling, we generate an avatar with a high-quality and reasonable appearance in three stages of post-processing in Sec. B.

k-nearest neighbor $\mathcal{N}$. $\mathcal{N}_k(P_1, P_2)$ denotes the k-nearest neighbors in P_2 for each point in P_1. $\arg\min \mathcal{N}_k(P_1, P_2)$ represents the indices of the k-nearest neighbors from points in P_1 to points in P_2 [12]. We omit the subscript k when $k = 1$.

Note that $\{\mathbf{x}_\phi^{d,i}\}_{\chi_{\phi,\text{add}}^i=1}$ denotes the additional point clouds from the *source human* to fill the gaps between the *source human* and *target human* because of the exception of *target human*'s attribute, as shown in the red box of Fig. 4a. To create $\chi_{\phi,\text{add}}^i$, We exclude the vertices from the FLAME vertices $\mathbf{x}_{th}^{\text{FLAME}}$ that are not associated with the additional part by using χ_{th} and the back of the head part of the FLAME that we designate. We denote the mask cue for obtaining FLAME corresponding to the additional part as $\chi_{th,\text{add}}^{\text{FLAME}}$.

$$\mathbf{x}_{th,\text{add}}^{\text{FLAME}} = \{\mathbf{x}_{th}^{\text{FLAME}}\}_{\chi_{th,\text{add}}^{\text{FLAME}}} \tag{3}$$

We apply $\mathcal{N}$ to $\mathbf{x}_\phi^d$ and $\mathbf{x}_{th,\text{add}}^{\text{FLAME}}$ to obtain the nearest neighbor of *source human*. To create the additional part only, we use $(1 - \chi_\phi)$ except for *source human*'s attribute.

$$\chi_{\phi,\text{add}} = (1 - \chi_\phi) \circ \arg\min \mathcal{N}_k(\mathbf{x}_{th,\text{add}}^{\text{FLAME}}, \mathbf{x}_\phi^d), \tag{4}$$

where $\circ$ represents the Hadamard product. We use $k = 2000$ to generate the additional point clouds as described in Fig. 4a.

Optimization Step. After the naive composition, there is still a gap between the *source human*'s face attribute and *target human*'s other parts because of the misalignment of the subject-specific FLAME canonical space, as shown in Fig. 4a. To solve this issue, we apply the optimization process to minimize the distance between the *source human* and *target human*. To obtain the landmark points, we apply the k-nearest neighbor function between the landmarks of deformed FLAME vertices [3] and $\mathbf{x}^d$ as follows:

$$\mathbf{x}^{\text{landmarks}} = \mathcal{N}(\mathbf{x}^{\text{FLAME landmarks}}, \mathbf{x}^d) \tag{5}$$

We leverage the distance between the *source human*'s 3D landmark points and the *target human*'s 3D landmark points as shown in Fig. 3.

$$\mathbf{d}_1 = \|\mathbf{x}_{th}^{\text{landmarks}} - \mathbf{x}_{\phi}^{\text{landmarks}}\|_2^2 \tag{6}$$

Furthermore, we calculate the squared distances between points in the additional *source human* part, denoted as $\{\mathbf{x}_\phi^d\}_{\chi_{\phi,\text{add}}=1}$, and points in the *target human*, represented by $\{\mathbf{x}_{th}^d\}_{\chi_{th}=0}$, from the k-nearest neighbors. For simplicity, the superscript i is omitted.

$$\mathbf{d}_2 = \|\mathcal{N}(\{\mathbf{x}_\phi^d\}_{\chi_{\phi,\text{add}}=1}, \{\mathbf{x}_{th}^d\}_{\chi_{th}=0})\|_2^2 \tag{7}$$

We optimize the learnable angle-axis rotation vector $R \in \mathbb{R}^3$ and translation vector $t \in \mathbb{R}^3$ to minimize the distance $\mathbf{d} = \mathbf{d}_1 + \mathbf{d}_2$ by Adam optimizer [6]. Note that we apply the rotation and translation vector at the subject-specific FLAME-canonical space.

$$\mathbf{x}_{\phi,\text{moved}}^{fc} = R \cdot \{\mathbf{x}_\phi^{fc}\} + t \tag{8}$$

We obtain the moved *source human*'s point cloud $\mathbf{x}_{\phi,\text{moved}}^d$ from $\mathbf{x}_{\phi,\text{moved}}^{fc}$ by Eq. (8). As a consequence, the optimized point cloud is represented as follows:

$$P_{\text{optim}} = \{\mathbf{x}_{\phi,\text{moved}}^d\}_{\chi_\phi \circ \chi_{\phi,\text{add}}=1} \cup \{\mathbf{x}_{th}^d\}_{\chi_{th}=0} \tag{9}$$

The optimized rendering result is shown in Fig. 4b.

Blending Step. To generate a natural rendering of the additional part, denoted as $\{\mathbf{x}_\phi^d\}_{\chi_{\phi,\text{add}}=1}$, we leverage the feature information from the *target human* using the k-nearest neighbor.

$$\mathbf{c}_{\text{add}}^d = \arg\min \mathcal{N}(\mathbf{x}_{th}, \{\mathbf{x}_\phi^d\}_{\chi_{\phi,\text{add}}=1}) \circ \mathbf{c}_{th}^d \tag{10}$$

$$\mathbf{n}_{\text{add}}^d = \arg\min \mathcal{N}(\mathbf{x}_{th}, \{\mathbf{x}_\phi^d\}_{\chi_{\phi,\text{add}}=1}) \circ \mathbf{n}_{th}^d \tag{11}$$

The RGB and normal of the additional part come from the target human, so we obtain the naturally blended avatar through the zero-shot model. The natural blended results are shown in Fig. 4c.

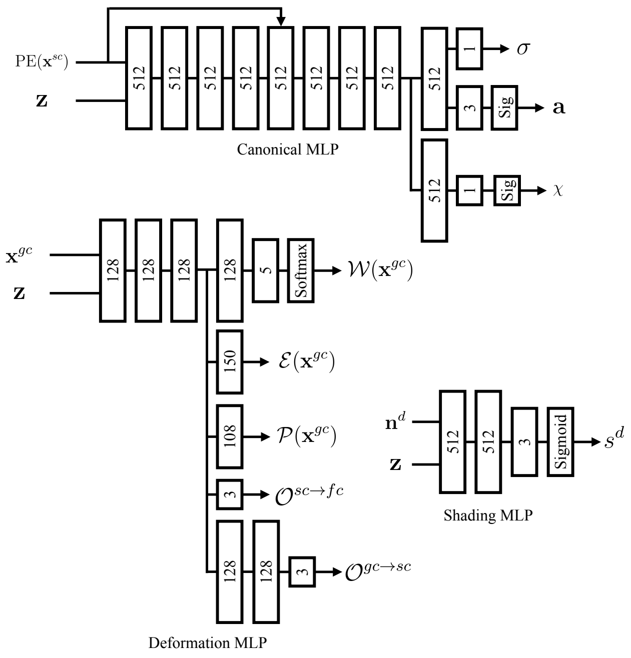


Figure 5. **Network Architecture of PEGASUS.** In PEGASUS, the latent code $\mathbf{z}$ serves as a condition for all the MLPs.

C. Implementation Details

Network Architecture In Fig. 5, we show the network architecture of PEGASUS. Following PointAvatar [20], we leverage ReLU activation function [10] for shading MLP, and Softplus activation function for canonical and deformation MLP for every layer. *Sig* denotes the sigmoid function in Fig. 5. Different from PointAvatar, we use an additional layer to output segmentation cues χ in canonical MLP. Also, we use two layers of MLP to create subject-specific canonical offset $\mathcal{O}^{gc\to sc}$

Loss Functions. The total loss for PEGASUS is defined as follows:

$$\begin{aligned}\mathcal{L} = \lambda_{\text{rgb}}\mathcal{L}_{\text{rgb}} + \lambda_{\text{mask}}\mathcal{L}_{\text{mask}} + \lambda_{\text{FLAME}}\mathcal{L}_{\text{FLAME}} + \lambda_{\text{vgg}}\mathcal{L}_{\text{vgg}} \\ + \lambda_{\text{normal}}\mathcal{L}_{\text{normal}} + \lambda_{\text{seg}}\mathcal{L}_{\text{seg}} + \lambda_{\mathbf{z}\ \text{reg}}\mathcal{L}_{\mathbf{z}\ \text{reg}}\end{aligned} \tag{12}$$

We leverage the loss functions from the facial implicit representations from monocular inputs [19, 20] as follows:

$$\mathcal{L}_{\text{rgb}} = \|c - c^{\text{GT}}\| \tag{13}$$

$$\mathcal{L}_{\text{mask}} = \|M - M^{\text{GT}}\| \tag{14}$$

$$\mathcal{L}_{\text{vgg}} = \|F_{\text{vgg}}(c) - F_{\text{vgg}}(c^{\text{GT}})\| \tag{15}$$

$$\begin{aligned}\mathcal{L}_{\text{FLAME}} = \frac{1}{N}\sum_{i=1}^{N}(&\lambda_e\|\mathcal{E}_i - \hat{\mathcal{E}}_i\|_2 \\ &+ \lambda_p\|\mathcal{P}_i - \hat{\mathcal{P}}_i\|_2 \\ &+ \lambda_w\|\mathcal{W}_i - \hat{\mathcal{W}}_i\|_2)\end{aligned} \tag{16}$$

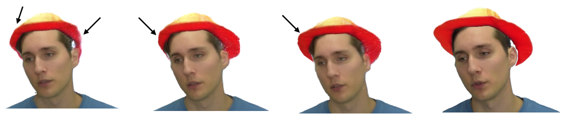


Figure 6. **Ablation: reconstruction by offsets $\mathcal{O}$.** Our three-stage canonical space framework creates reasonable and accurate reconstruction.

Following PointAvatar, c and c^{GT} denote the color of the rendering images from PEGASUS and ground-truth color. M denotes the mask from PEGASUS obtained by $\mathbf{m}_{\text{pix}} = \sum_i \alpha_i \mathbf{T}_i$. $F_{\text{vgg}}(\cdot)$ represent the features of pretrained VGG network [4, 14]. $\mathcal{E}, \mathcal{P},$ and $\mathcal{W}$ are the pseudo ground truth of the k-nearest neighbor vertices of the FLAME [8]. Note that our method, PEGASUS, does not predict the shape blendshapes basis $\mathcal{S}$, directly using the k-nearest neighbor vertices of the FLAME.

Given ground-truth object mask M_{seg}^{GT} and the predicted segmentation cues χ^d, the rendered color of the segmented point cloud represents $\mathcal{R}(\chi^d \circ \mathbf{x}^d)$. The segmentation loss is defined as:

$$\mathcal{L}_{\text{seg}} = \text{BCE}(\mathcal{R}(\chi^d \circ \mathbf{x}^d), M_{\text{seg}}^{GT}) \tag{17}$$

BCE represent the Binary Cross-Entropy loss. $\mathcal{R}$ is the alpha composition rendering function. We leverage the alpha composition function of PyTorch3D [12] to render the predicted segmentation cues.

We adopt the normal loss to encourage high-fidelity geometry and texture as follows:

$$\mathcal{L}_{\text{normal}} = \|\mathbf{n} - \mathbf{n}^d\| \tag{18}$$

We generate the pseudo ground truth normal $\mathbf{n}$ from the V^{tp} and the avatar trained with a single identity of each V^{db}. We apply the regularization of latent code to be close to zero.

$$\mathcal{L}_{\mathbf{z}\ \text{reg}} = \|\mathbf{z}\| \tag{19}$$

Training Strategy We train PEGASUS in two stages. In the first stage, we only use the target individual from V^{tp} for training. In this way, the initial point cloud is deformed from a sphere to have a reasonable face shape. In the second stage, we continue training using all part-swapped videos from $\hat{V}_i^{tp}$. We have empirically find that this two-stage training shows more reliable training. In all of our experiments, we start the second stage from the 10th epoch, using 1600 point clouds.

D. More Ablation Study

Multi-stage Canonical Spaces. In Fig. 6 and Tab. 3, our multi-stage canonical space and point deformation method outperforms the best metrics and quality compared to other approaches. Notably, as an example of Fig. 6, the closest high-quality reconstruction to the Ground Truth (GT) is achieved by the three-stage approach.

Additionally, we randomly sample the nose latent codes to utilize two and three-stage baselines in Fig. 7. Fig. 7 shows that the original architecture of PointAvatar fails to disentangle the target attribute. Fig. 7 demonstrates that the generic canonical space is necessary to generate disentangled facial attributes from random sampling while preserving individual identity.

Normal Loss. We show the advantage of our normal loss, which is used for training the avatar model. In Fig. 8, the result shows that the normal loss improves the RGB and normal qualities, resulting in more realistic appearances.

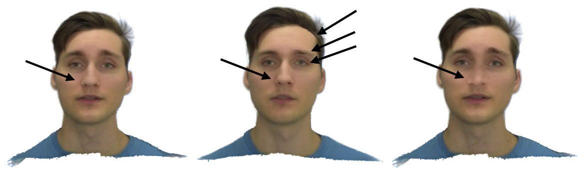


Figure 7. **Ablation: random sampling by offsets $\mathcal{O}$.** Multi-stage canonical spaces enhance disentangled nose generation.

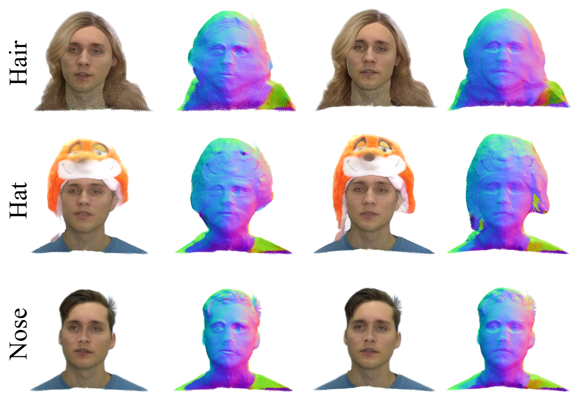


Figure 8. **Ablation: normal loss.** The normal loss improves the RGB and geometry quality of our model.

E. Additional Experiments

Temporal Consistency. We employ the temporal consistent metrics [16] to evaluate the preservation of identity across generated image sequences. TL-ID denotes the temporally local identity preservation, which evaluates the consistency of image sequences locally, focusing on the pairs of adjacent frames. TG-ID represents the temporally global identity preservation metric to measure the similarity across all possible pairs of video frames, including those that are not ad-

Method	TL-ID↑	TG-ID↑	L_2↓	LPIPS↓
CD + PA	**0.9968**	0.9095	114.73	0.1428
E4S + PA	0.9960	0.8677	110.66	0.1234
DELTA	0.9404	0.8368	135.03	0.1663
Ours$_{\text{swap}}$+PA	0.9940	**0.9558**	**97.170**	**0.1115**
Ours$_{\text{person-gen}}$	**0.9910**	**0.9254**	**105.65**	**0.1224**
Ours$_{\text{zero shot}}$	0.9908	0.9178	114.38	0.1328

Table 1. **Temporal metric (TL-ID, TG-ID) [16] and transfer accuracy (L_2, LPIPS).** We evaluate the SOTA baselines the same as Tab. 1 with two types of metrics: temporal consistency and preservation of transferred attributes.

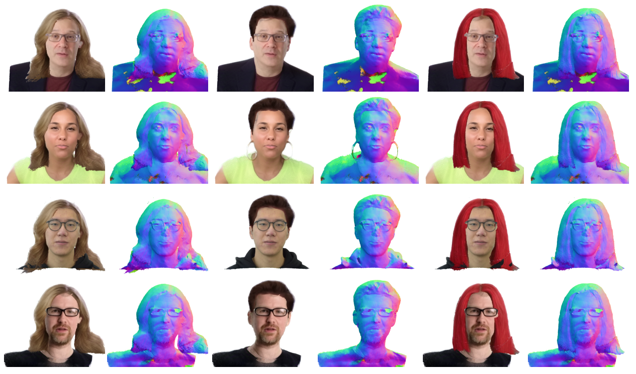


Figure 9. **Additional Results of Zero-Shot Transfer.** PEGASUS robustly transfers facial attributes to any target human without the need for additional training.

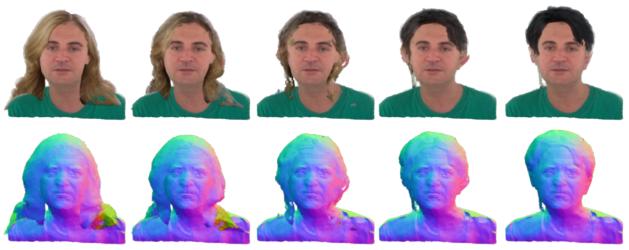


Figure 10. **Zero-Shot Interpolation.** With the help of interpolation-capable segmentation cues by the segmentation network of canonical MLP, we create interpolation in a zero-shot model.

jacent. We evaluate the same baselines in Tab. 1. As shown in Tab. 1, our synthesis method, Ours$_{\text{swap}}$+PA, represents the best quality of the TG-ID. TL-ID does not show significant differences and performs well across all baselines, as it evaluates consistency only for adjacent frames.

Attributes Transfer. In Tab. 1, we further quantify the preservation of the transferred attributes. We evaluate via the metrics by masking the region except the target attribute. We conduct a comparison between the images generated and the attributes of novel head poses. Our method, Ours$_{\text{swap}}$+PA, outperforms the SOTA baselines.

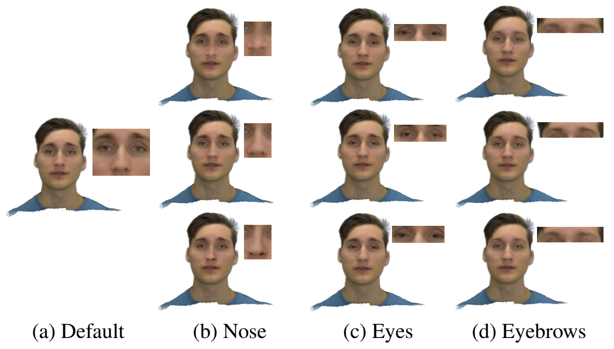


Figure 11. **Random sampling of PEGASUS.** We randomly sample the latent codes of the PEGASUS.

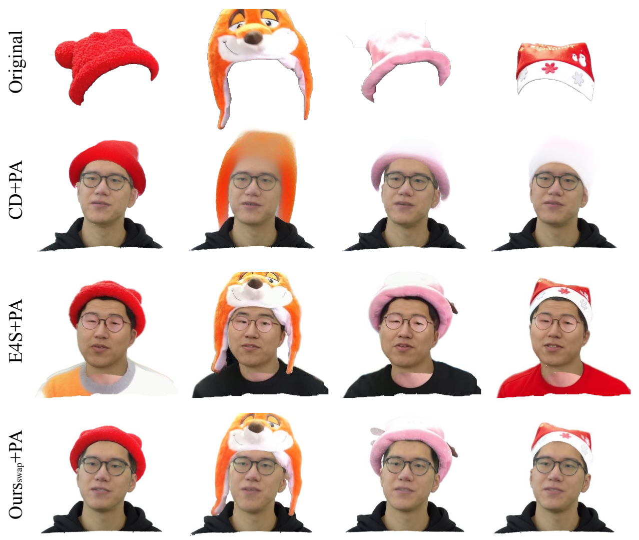


Figure 12. **Single Part-Swapped Avatar on Hat.** Our synthesis method creates high-quality and properly wearing avatars.

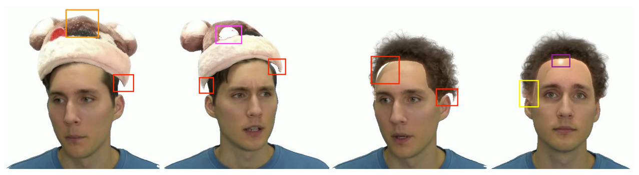


Figure 13. **Limitations of the Synthetic DB.** In the synthetic DB generation process, we illustrate the instances of failure cases through color-coded annotations. Orange box represents the artifacts occurring during the generation of attribute image. Red box indicates instances of facial attributes that are physically inconsistent, revealing inaccuracies in the appearance of the face. Magenta box marks the failure case of the segmentation. Purple box identifies instances where the diffusion model fails to generate bald faces inconsistently. Yellow box signifies the failure cases in post-processing.

F. More Results

Zero-Shot Transfer. Fig. 9 presents additional results of zero-shot transfer. PEGASUS robustly and naturally transfers facial attributes to any target human in the wild. Fig. 10 demonstrates facial attribute interpolation in zero-shot modeling, aided by latent code $\mathbf{z}$ interpolation. This shows that segmentation cues are capable of interpolation by the canonical MLP.

Random Sampling. We present PEGASUS' latent random sampling result to support the additional generative aspect of our approach in Fig. 11. We sample each latent code from the Gaussian distribution with the mean and variance of latent codes of each category. As depicted in Fig. 11, our method successfully generates random samples exhibiting distinguishable facial attributes.

Additional Categories. In Fig. 12, our synthesis method maintains the identity better than other baselines and also shows the hat similar to the original while being appropriately worn by the avatar.

G. Limitations

As a limitation, the quality of our personalized avatar still does not reach the photo-realistic quality, showing noticeable artifacts. Also, due to the reliance on non-physical-based methods for generating the synthetic DB, our approach exhibits limitations in achieving physical accuracy. We describe the failure cases and limitations of the synthetic DB generation in Fig. 13.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). 2017.

[3] Yao Feng, Haiwen Feng, Michael J Black, and Timo Bolkart. Learning an animatable detailed 3d face model from in-the-wild images. *ACM Trans. Gr.*, 40(4):1–13, 2021.

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016.

[5] Zhanghan Ke, Jiayu Sun, Kaican Li, Qiong Yan, and Rynson WH Lau. Modnet: Real-time trimap-free portrait matting via objective decomposition. 2022.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[8] Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4d scans. *ACM Trans. Graph.*, 36(6):194–1, 2017.

[9] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[11] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023.

[12] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. 2022.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004.

[16] Rotem Tzaban, Ron Mokady, Rinon Gal, Amit Bermano, and Daniel Cohen-Or. Stitch it in time: Gan-based facial editing of real videos. 2022.

[17] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. 2018.

[18] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[19] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C Bühler, Xu Chen, Michael J Black, and Otmar Hilliges. Im avatar: Implicit morphable head avatars from videos. 2022.

[20] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J Black, and Otmar Hilliges. Pointavatar: Deformable point-based head avatars from videos. 2023.

[21] zllrunning. face-parsing.pytorch. `https://github.com/zllrunning/face-parsing.PyTorch`, 2019.