Expressive Whole-Body 3D Gaussian Avatar

Gyeongsik Moon[1,2], Takaaki Shiratori[2], and Shunsuke Saito[2]

[1]DGIST
mks0601@dgist.ac.kr

[2]Codec Avatars Lab, Meta
{tshiratori,shunsukesaito}@meta.com

https://mks0601.github.io/ExAvatar

In this supplementary material, we provide more experiments, discussions, and other details that could not be included in the main text due to the lack of pages. The contents are summarized below:

- Sec. A: Details of co-registration of SMPL-X
- Sec. B: Detailed architecture of MLPs
- Sec. C: Regularizers
- Sec. D: Running time comparison
- Sec. E: 3D geometry comparison
- Sec. F: Comparison to generative AIs
- Sec. G: Implementation details
- Sec. H: Failure cases

A Details of co-registration of SMPL-X

This section describes details of the co-registration of SMPL-X of Sec. 3.1 of the main manuscript.

FLAME registration. Given DECA [2]'s output FLAME parameters (*i.e.*, face shape parameter, facial expression code, and jaw pose), we further fit them to 2D keypoints of face, where the 2D keypoints are from an off-the-shelf regressor [1]. For the fitting, we minimize the below loss functions.

$$L_{\text{FLAME}} = L_{\text{kpt}} + 0.1 L_{\text{init}}, \tag{1}$$

where L_{kpt} and L_{init} represent 1) $L1$ distance between projected and target 2D keypoints and 2) $L1$ distance between optimizing FLAME parameters and the initial regressed ones, respectively. In this way, we can make FLAME's meshes pixel-aligned with the image while preventing them from being too far from DEAC's regressed ones.

SMPLX registration. Given Hybrid-X [9]'s output SMPL-X parameters, we further fit them to 2D keypoints of the whole body, where the 2D keypoints are from an off-the-shelf regressor [1]. We replace the facial expression code of Hybrid-X with that of DECA as DECA's output is more accurate. As FLAME and SMPL-X share the same facial expression space, we share the facial expression code of FLAME and SMPL-X during the registration. Before performing the LBS inside of the SMPL-X layer, we 1) add the joint offset $\Delta\mathbf{J}$ to the T-pose joint locations of SMPL-X and 2) add the face offset $\Delta\mathbf{V}_{\text{face}}$ to the face region of the T-pose template mesh. The joint offset allows us to obtain a more personalized 3D skeleton and surface than only using the SMPL-X shape parameter.

Please note that the joint offset affects both skeleton and surface as the added joint locations are used for following forward kinematics and LBS.

We minimize the below loss functions.

$$L_{\text{SMPLX}} = L_{\text{kpt}} + 0.1L_{\text{init}} + L_{\text{face}} + L_{\text{reg}}, \tag{2}$$

$$L_{\text{face}} = 10L_{\text{vertex}} + 10000L_{\text{lap}} + L_{\text{edge}}, \tag{3}$$

$$L_{\text{reg}} = 0.01L_{\text{shape}} + 100L_{\text{jo}} + L_{\text{sym}}, \tag{4}$$

where L_{kpt} and L_{init} represent 1) $L1$ distance between projected and target 2D keypoints and 2) $L1$ distance between optimizing SMPL-X parameters and the initial regressed ones, respectively. We compute the L_{kpt} with and without the face offset $\Delta\mathbf{V}_{\text{face}}$ to prevent the face offset from representing any pose-dependent and expression-dependent things. L_{face} is to optimize the face offset $\Delta\mathbf{V}_{\text{face}}$. L_{vertex}, L_{lap}, and L_{edge} represent 1) $L1$ distance between SMPL-X face vertices and FLAME vertices, 2) $L2$ distance between Laplacian of SMPL-X face mesh and FLAME mesh, and 3) $L1$ distance between edge length of SMPL-X face mesh and FLAME mesh, respectively. We compute L_{vertex} with and without the pose and facial expression to prevent the face offset from representing any pose-dependent and expression-dependent things. L_{lap} and L_{edge} are computed only for meshes without pose and facial expressions. We compute L_{face} only when the face is visible. To check the face visibility, we compute two vectors. First, a vector from the face center to the middle of the eyes in the xz space of the camera-centered coordinate system. Second, a vector from the camera position (origin) to the face center in xz space of the camera-centered coordinate system. We decide the face is visible if the dot product between the two vectors is smaller than $\cos(135°)$. Finally, L_{reg} is to regularize the SMPL-X shape parameter, joint offset, and face offset. L_{shape} is a squared $L2$ norm of the SMPL-X shape parameter. L_{jo} is a squared $L2$ norm of the joint offset, which prevents an extreme joint offset, and L_{sym} is for symmetricity of the joint offset and face offset, similar to Feng *et al.* [2].

B Architecture of MLPs

This section describes the detailed architecture of MLPs of Sec. 3.2 of the main manuscript. The MLPs are briefly described in Fig. 4 of the main manuscript.

B.1 MLPs without pose conditioning

The MLPs without the pose conditioning (the red MLPs of Fig. 4 of the main manuscript) consist of two types of MLPs. The first MLP takes the triplane feature $\mathbf{F}$ and outputs the geometry of 3D Gaussian points (*i.e.*, $\Delta\mathbf{V}_{\text{tri}}$ and $\mathbf{S}_{\text{tri}}$). It consists of four fully connected layers with a hidden size of 128. We use

Table A: Frames per second comparisons of 3D human avatars.

NeuMan [7]	Vid2Avatar [5]	X-Avatar [12]	**ExAvatar (Ours)**	**ExAvatar (Ours wo. pose cond.)**
0.02	0.04	0.12	26	47

the group normalization [13] and ReLU activation function between each fully connected layer. The second MLP takes the triplane feature $\mathbf{F}$ and outputs RGB colors of 3D Gaussian points $\mathbf{C}_{\text{tri}}$. It has the same architecture as that of the first MLP except for the output dimension.

B.2 MLPs with pose conditioning

The MLPs with the pose conditioning (the blue MLPs of Fig. 4 of the main manuscript) consist of two types of MLPs. The MLPs have the same architecture as that of the MLPs of Sec. B.1 except for the input and output dimensions. The first MLP takes the triplane feature $\mathbf{F}$ and 3D pose θ of SMPL-X without the root pose and outputs geometry offsets of 3D Gaussian points (*i.e.*, $\Delta\mathbf{V}_{\text{pose}}$ and $\Delta\mathbf{S}_{\text{pose}}$). The second MLP takes the 1) triplane feature $\mathbf{F}$, 2) 3D pose θ of SMPL-X without the root, and 3) normal of each 3D Gaussian point and outputs RGB offset $\Delta\mathbf{C}_{\text{pose}}$.

C Regularizers

This section describes regularizers, not introduced in Sec. 3.4 of the main manuscript. First, to prevent 3D Gaussian points from being too far from underlying canonical mesh $\bar{\mathbf{V}}$, we penalize squared $L2$ norm of 3D Gaussian offsets $\Delta\mathbf{V}_{\text{tri}}$ and $\Delta\mathbf{V}_{\text{pose}}$. Second, to prevent 3D Gaussian from being too big, we penalize the squared $L2$ norm of 3D Gaussian scales. Third, as hands often suffer from noisy colors due to their small scales and complicated articulations, we minimize the squared $L2$ distance between RGB colors of hand 3D Gaussian points and the average colors of hand 3D Gaussian points. Finally, we use the same L_{jo} of Sec. A.

D Running time comparison

Tab. A shows that ours achieves real-time animation and rendering speed while existing works [5, 7] fail to. Without the pose conditioning (the blue MLP of Fig. 4 of the main manuscript), ours achieves even faster speed. For all methods, we measure the running time only for the human animation and rendering without the background rendering. A single NVIDIA V100 GPU is used, and the rendering resolution is 1024×1024.

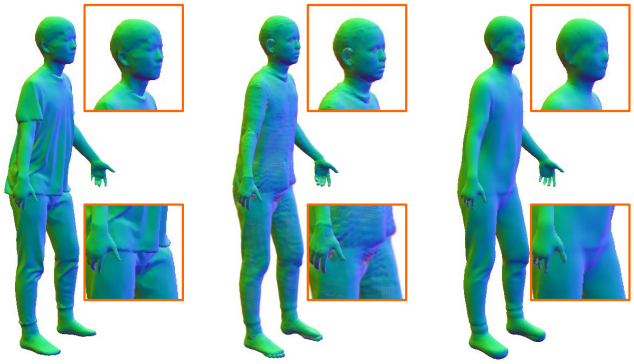

Fig. A: Qualitative comparison of rendered normals from (a) our ExAvatar and (b) X-Avatar [12] on the test set of X-Humans [12].

Table B: 3D geometry comparison between our ExAvatar and X-Avatar [12] on the subset of the test set of X-Humans [12].

Methods	Mask IoU ↑	Depthmap (mm) ↓	Normal consistency ↑
X-Avatar	94.13	11.24	0.823
ExAvatar (Ours)	**96.11**	**10.97**	**0.868**

E Geometry comparison

Fig. A and Tab. B show that our ExAvatar produces better 3D geometry than previous state-of-the-art 3D whole-body avatar [12], especially around the face and neck. It is not straightforward to extract actual 3D geometry from 3D Gaussians. Instead, we render masks/depth maps/normal maps of Gaussians to multiple viewpoints and comparing them with ground truth. However, when rendering depth/normals, even recent SOTA methods [3] simply take the depth/normal from the center of Gaussians and alpha-blend it in the screen space. This may result in incorrect depth/normal maps as elongated Gaussians can deviate from the depth/normal values of the Gaussian centers. Nevertheless, we report 3D geometry comparisons in the figure and table for reference. We leave actual 3D geometry extraction from ExAvatar as a future work.

F Comparison to generative AIs

Motivated by the powerful modeling capability of the diffusion models [6, 11], several human animation methods [14] have been introduced. They can animate humans only from a single image using DensePose [4] or 2D pose as the driving signals. Despite their photorealistic and plausible outputs, they lack authenticity, as shown in Fig. B. The person of the (b) generated image from MagicAnimate [14] has definitely a different face identity and clothes compared to the (a) captured image, although they look similar. The result of MagicAnimate [14] is

Fig. B: Qualitative comparison of (b) our ExAvatar and (c) state-of-the-art generative method [14] for the human animation.

Fig. C: Failure case of our ExAvatar due to (b) the wrong regressed facial expression code ψ from the off-the-shelf regressor [2].

obtained using their official released code. This is because of the hallucination, one of the biggest challenges for modeling generative AIs. On the other hand, ours might lack plausibility, especially for unobserved human parts such as inside of mouth and cloth dynamics; however, ours has more authenticity. We think combining the plausibility of generative approaches and the authenticity of our approach should be an interesting and future direction, as discussed in Sec. 5 of the main manuscript.

G Implementation details

PyTorch [10] is used for implementation. For the training, we use Adam optimizer [8]. The initial learning rate is set to 10^{-3} and reduced by a factor of 10 at the 75 % and 90 % of total number of iterations. We use a single V100 NVIDIA GPU for experiments, and the training takes 1.5 hours to 5 hours depending on the duration of the videos. All other details will be available in our code.

H Failure cases

Fig. C shows that our rendered avatar does not have a smiling facial expression despite the person in the image is smiling. This is because of the wrong regressed facial expression code ψ from the off-the-shelf regressor [2] as (b) rendered SMPL-X mesh also has the wrong facial expression. As our ExAvatar is animated with

Fig. D: Failure case of our ExAvatar due to the baked facial appearance of a subject.

the regressed facial expression code, ours also has wrong facial expression. We think advanced facial models and high-fidelity regressors should be investigated to address this failure case.

Fig. D additionally shows our failure case. Although the person in (a) the image is frowning, our rendered avatar is slightly smiling. This is because the subject of our avatar in the NeuMan dataset [7] is always smiling during the short video capture. Hence, the smiling facial appearance is baked into our avatar. Canonicalizing facial appearance from a short monocular is greatly challenging. In particular, when the face takes a few pixels in the video like our case, 2D keypoints are often noisy, which makes the lip geometry of SMPL-X/FLAME are not fit perfectly in the co-registration stage (Sec. 3.1 of the main manuscript). Such a misalignment error is propagated to our ExAvatar learning stage, which causes the failure case. We think using priors of the canonical facial appearances using generative methods can be one way to address this failure case.

References

1. Contributors, M.: Openmmlab pose estimation toolbox and benchmark. `https://github.com/open-mmlab/mmpose` (2020)
2. Feng, Y., Feng, H., Black, M.J., Bolkart, T.: Learning an animatable detailed 3D face model from in-the-wild images. ACM TOG (2021)
3. Guédon, A., Lepetit, V.: SuGaR: Surface-aligned gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In: CVPR (2024)
4. Güler, R.A., Neverova, N., Kokkinos, I.: DensePose: Dense human pose estimation in the wild. In: CVPR (2018)
5. Guo, C., Jiang, T., Chen, X., Song, J., Hilliges, O.: Vid2Avatar: 3D avatar reconstruction from videos in the wild via self-supervised scene decomposition. In: CVPR (2023)
6. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: NeurIPS (2020)
7. Jiang, W., Yi, K.M., Samei, G., Tuzel, O., Ranjan, A.: NeuMan: Neural human radiance field from a single video. In: ECCV (2022)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2014)
9. Li, J., Bian, S., Xu, C., Chen, Z., Yang, L., Lu, C.: HybrIK-X: Hybrid analytical-neural inverse kinematics for whole-body mesh recovery. arXiv preprint arXiv:2304.05690 (2023)
10. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
11. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022)
12. Shen, K., Guo, C., Kaufmann, M., Zarate, J.J., Valentin, J., Song, J., Hilliges, O.: X-Avatar: Expressive human avatars. In: CVPR (2023)
13. Wu, Y., He, K.: Group normalization. In: ECCV (2018)
14. Xu, Z., Zhang, J., Liew, J.H., Yan, H., Liu, J.W., Zhang, C., Feng, J., Shou, M.Z.: MagicAnimate: Temporally consistent human image animation using diffusion model. arXiv preprint arXiv:2311.16498 (2023)