Supplementary Materials for GANHead: Towards Generative Animatable Neural Head Avatars

Sijing Wu[1] Yichao Yan[2*] Yunhao Li[1] Yuhao Cheng[2]
Wenhan Zhu[2] Ke Gao[3] Xiaobo Li[3] Guangtao Zhai[1,2*]
[1]Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
[2]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[3]Alibaba Group
{wusijing, yanyichao, lyhsjtu, chengyuhao, zhuwenhan823, zhaiguangtao}@sjtu.edu.cn
{gaoke.gao, xiaobo.lixb}@alibaba-inc.com

In the supplementary document, we first provide some implementation details in Section 1, then show additional qualitative and quantitative results in Section 2. Finally, the limitations and broader impact are discussed in Section 3. Please also refer to the accompanying video for more intuitive results.

1. Implementation Details

1.1. Network Architecture

Our model is primarily constructed by MLPs, and we use PyTorch [8] to implement our model. Our model consists of a canonical generation module that generates head avatars in canonical space, and a deformation module that deforms the generated avatars to target poses and expressions.

The canonical generation module consists of three networks: geometry network $\mathcal{G}$, normal network $\mathcal{N}$ and texture network $\mathcal{T}$. $\mathcal{G}$ is built by a 3D feature generator followed by an MLP conditioned by the generated feature similar to [1]. $\mathcal{N}$ and $\mathcal{T}$ have roughly the same structure, as shown in Fig. 1.

The deformation module is composed of the shape removing network $\mathcal{D}$ followed by a MLP $\mathcal{C}$ that predicts the pose bases, expression bases and LBS weights similar to [13], as illustrated in Fig. 2

1.2. Data Processing

We use the textured scans in FaceVerse-Dataset [10] to train our model. At the beginning, we flip the scans to face forward and shift them to the origin, so that the scans are right inside the $[-1, 1]^3$ cube.

The training of our model requires accurate estimation of FLAME [6] parameters for each scan. To this end, we initialize the FLAME parameters by inferencing the pretrained DECA model [3], and then further optimize these parameters based on mesh-to-scan distance and landmark loss. To inference DECA and obtain 3D landmarks, we render the textured scans from the front view using PyTorch3D [9]. The rendered RGB image is then used to detect 2D landmarks using Dlib [4]. Then, we use the Pyrender library to render depth map and then project the 2D landmarks to 3D scans according to the depth to obtain 3D landmarks.

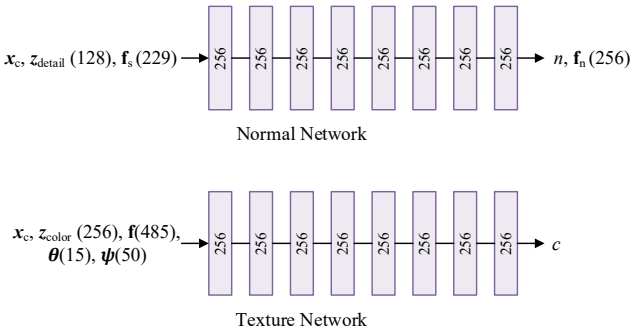

Figure 1. **The architecture of $\mathcal{N}$ and $\mathcal{T}$.** Each block represents a linear layer whose output dimension is specified inside the block. The number in brackets indicate the length of the tensor.

As described in the main paper, we train our model in two stages. In the second stage, rendered multi-view images are required. As for the ground truth scans, we render them from 9 views $(0°, 40°, 80°, \cdots)$ to 256×256 images using PyTorch3D [9]. As for the coarse geometry obtained in the first stage, we first extract meshes from the predicted occupancy using MISE [7], and then calculate the 3D to 2D correspondence using the depth map rendered by Pyrender. We will release our code for research purposes, and for more details please refer to our source code.

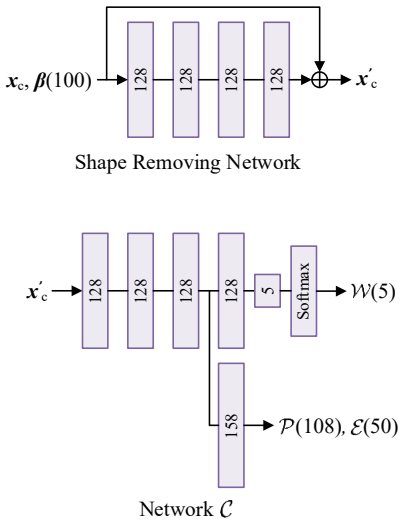

Figure 2. **The architecture of $\mathcal{D}$ and $\mathcal{C}$.** Each block represents a linear layer whose output dimension is specified inside the block. The number in brackets indicates the length of the tensor.

1.3. Training

Losses: We train our model in two stages. In the first stage, we add two auxiliary losses $\mathcal{L}_{bone}$ and $\mathcal{L}_{weight}$ during the first training epoch inspired by [1,2]. Thus, at the first epoch of the stage one, the training loss is:

$$\begin{aligned}\mathcal{L}_{stage1} =& \mathcal{L}_{occ} + \lambda_d \mathcal{L}_{deshape} + \lambda_l \mathcal{L}_{lbs} + \lambda_r \mathcal{L}_{reg} + \\ & \lambda_b \mathcal{L}_{bone} + \lambda_w \mathcal{L}_{weight},\end{aligned} \tag{1}$$

with

$$\begin{aligned}\mathcal{L}_{bone} &= BCE(\mathcal{G}(x_b, \boldsymbol{z}_{shape}), 1) \\ \mathcal{L}_{weight} &= \|\mathcal{W}_{joint} - \mathcal{W}_0\|_2^2,\end{aligned} \tag{2}$$

where $\mathcal{L}_{bone}$ ensures the 20 sampled points x_b along the 4 bones (a virtual concept that is similar to human body model) to be inside the surface, and $BCE(\cdot)$ denotes the binary cross entropy loss. $\mathcal{L}_{weight}$ enforces the predicted LBS weights of the joints $\mathcal{W}_{joint}$ only relevant to their two neighboring bones. (*i.e.*, $\mathcal{W}_0$ is a vector that takes 1 for the neighboring bones and 0 elsewhere). We take $\lambda_d = 10$, $\lambda_l = 0.2$, $\lambda_r = 10^{-3}$, $\lambda_b = 1$, $\lambda_w = 10$ and $\lambda_p = \lambda_e = 500$.

In the second stage, the weight of each loss function is set as: $\lambda_c = \lambda_n = \lambda = 1$ and $\lambda_r = 10^{-3}$.

Optimization: We initialize all three latent codes to zero, and train our model using Adam optimizer [5] with learning rate $\eta = 10^{-3}$ for the first training stage and $\eta = 2 \times 10^{-3}$ for the second stage, and $\beta = (0.9, 0.999)$.

2. Additional Experimental Results

2.1. Head Avatar Generation Quality Comparison

As illustrated in the main paper, our model can generate diverse animatable head avatars with complete geometry and realistic texture. Before our method, only i3DMM [12] can achieve similar functions. In this section, we compare our method to the SOTA method i3DMM in terms of the quality of the generated head avatars qualitatively and quantitatively.

We randomly sample the latent codes of our model and i3DMM respectively to generate diverse head avatars. The geometry and texture of some generated head avatars are shown in Fig. 4. Our model can generate head avatars with more diverse hairstyles and better geometry and more realistic texture. Furthermore, our model can learn more more complex geometry like collars.

To quantitatively evaluate the quality of the generated avatars, we conduct a user study. We ask 10 volunteers to assess the geometry and texture quality of the generated head avatars respectively. We randomly select 30 samples of each method (*i.e.* our method and i3DMM [12]), and each sample is rendered to image in resolution 256×256 from three views. Geometry images and texture images are rendered separately. Thus, each volunteer will score 120 images in all, which takes about 15 to 20 minutes. Before the formal scoring, we show each volunteer five additional images for training. In the formal scoring, the volunteers are asked to score the results of three methods from 0 to 5 (0 is the worst and 5 is the best, specifically, $0 \sim 1$: 'very bad', $1 \sim 2$: 'bad', $2 \sim 3$: 'ordinary', $3 \sim 4$: 'good', $4 \sim 5$: 'very good'). As illustrated in Fig. 3, our method outperforms the SOTA method in both aspects.

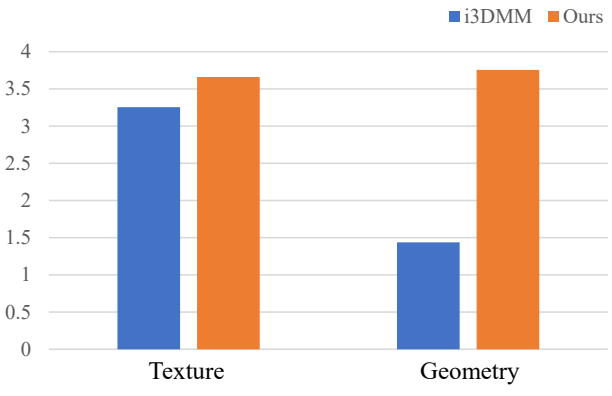

Figure 3. **User study results.** The average geometry and texture scores of our method are higher than those of the SOTA method.

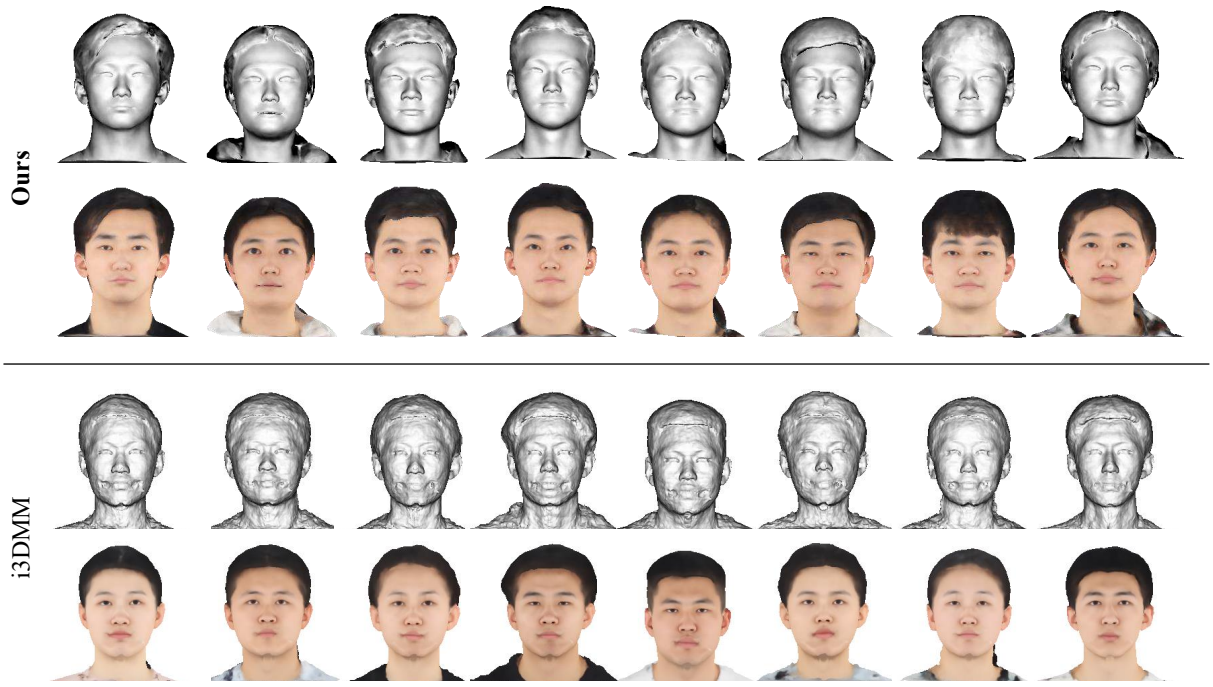

Figure 4. **Head avatar generation comparison with SOTA method.** We randomly sample the latent codes of our model and of i3DMM, and exhibit some generated head avatars.

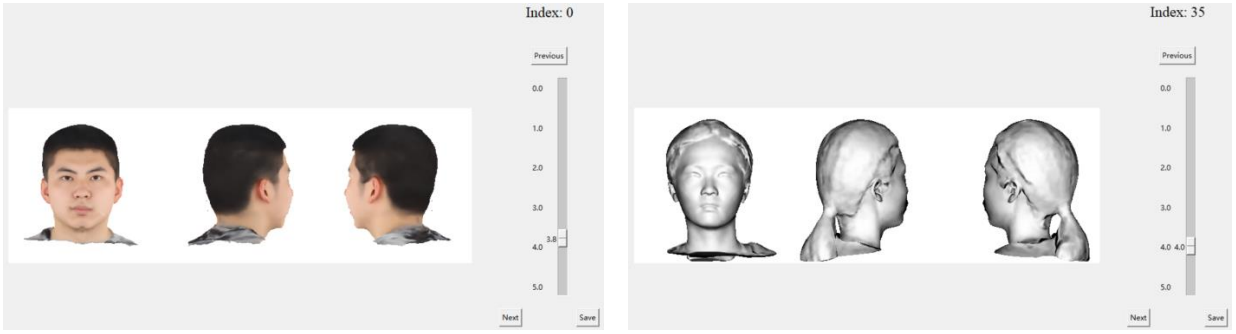

Figure 5. **User study examples.**

2.2. Additional Generation and Animation Results

We generate more neural head avatars, and then deform them to the target expressions, as illustrated in Fig. 8.

2.3. Qualitative results on Multiface Dataset

Since there are only Asians in the FaceVerse-Dataset that we used in the main paper, we also train our model on a subset of Multiface dataset [11] (252 scans from 10 subjects) which contains samples of other races. The shape code and detail code sampling results are shown in Fig.6. Due to the lack of geometry details of the meshes in the Multiface dataset, the sampling results only have rough shape on the hair region. Furthermore, We randomly sample the shape, detail and color latent codes to generate head avatars, and then deform them to the target poses and expressions controlled by the given FLAME parameters, as shown in Fig. 7. The results demonstrate the generalization ability of our method on different datasets and races.

3. Limitations and Broader Impacts

Limitations: While our method contributes towards building generative animatable head avatars with complete geometry and realistic texture, some challenges still remain. First, the data distribution and quality generated by our method depends on that of the training dataset. Since there are only Asians in the FaceVerse-Dataset that we used to

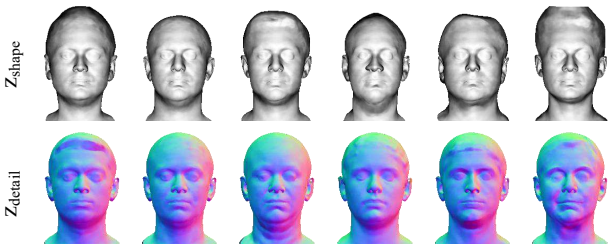

Figure 6. **Shape and detail latent codes sampling results on Multiface dataset.** When sampling the shape code, the detail code is set to the average value, and vice versa.

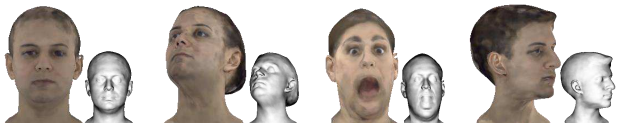

Figure 7. **Head avatars generation and animation on Multiface dataset.**

train our model, our model can only generate Asians. When trained on the Multiface dataset, our model can only generate avatars with only rough shape on the hair region due to the lack of details in the training dataset. Future work could address this by collecting a larger high-quality human head scan dataset with a rich population. Second, although our model can generate head avatars with realistic texture, it cannot produce facial details like freckles and wrinkles. More elaborate design of the 2D loss function in the second training stage may help our model learn more details. Finally, how to model the internal structure of the mouth such as teeth and tongue is still challenging.

Broader Social Impact: Our GANHead model can generate diverse novel realistic 3D head avatars, which is meaningful for the coming meta verse and AAA games. For instance, GANHead can help users create novel animatable digital humans quickly and produce diverse digital humans in the meta verse. Our method can also bring negative impact. In the future, GANHead can be used to make computer generated 3D human heads visually realistic and can be misused to create fake face animation using Deepfake-like technology. We heartily encourage future research to study more in detecting computer generated head avatar motions.

References

[1] Xu Chen, Tianjian Jiang, Jie Song, Jinlong Yang, Michael J Black, Andreas Geiger, and Otmar Hilliges. gdna: Towards generative detailed neural avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20427–20437, 2022. 1, 2

[2] Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11594–11604, 2021. 2

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

[4] Davis E King. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758, 2009. 1

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[6] 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. 1

[7] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 1

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1

[9] 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. 1

[10] Lizhen Wang, Zhiyuan Chen, Tao Yu, Chenguang Ma, Liang Li, and Yebin Liu. Faceverse: a fine-grained and detail-controllable 3d face morphable model from a hybrid dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20333–20342, 2022. 1

[11] Cheng-hsin Wuu, Ningyuan Zheng, Scott Ardisson, Rohan Bali, Danielle Belko, Eric Brockmeyer, Lucas Evans, Timothy Godisart, Hyowon Ha, Alexander Hypes, Taylor Koska, Steven Krenn, Stephen Lombardi, Xiaomin Luo, Kevyn McPhail, Laura Millerschoen, Michal Perdoch, Mark Pitts, Alexander Richard, Jason Saragih, Junko Saragih, Takaaki Shiratori, Tomas Simon, Matt Stewart, Autumn Trimble, Xinshuo Weng, David Whitewolf, Chenglei Wu, Shoou-I Yu, and Yaser Sheikh. Multiface: A dataset for neural face rendering. In *arXiv*, 2022. 3

[12] Tarun Yenamandra, Ayush Tewari, Florian Bernard, Hans-Peter Seidel, Mohamed Elgharib, Daniel Cremers, and Christian Theobalt. i3dmm: Deep implicit 3d morphable model of human heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12803–12813, 2021. 2

[13] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C Bühler, Xu Chen, Michael J Black, and Otmar Hilliges. Im avatar: Implicit morphable head avatars from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13545–13555, 2022. 1

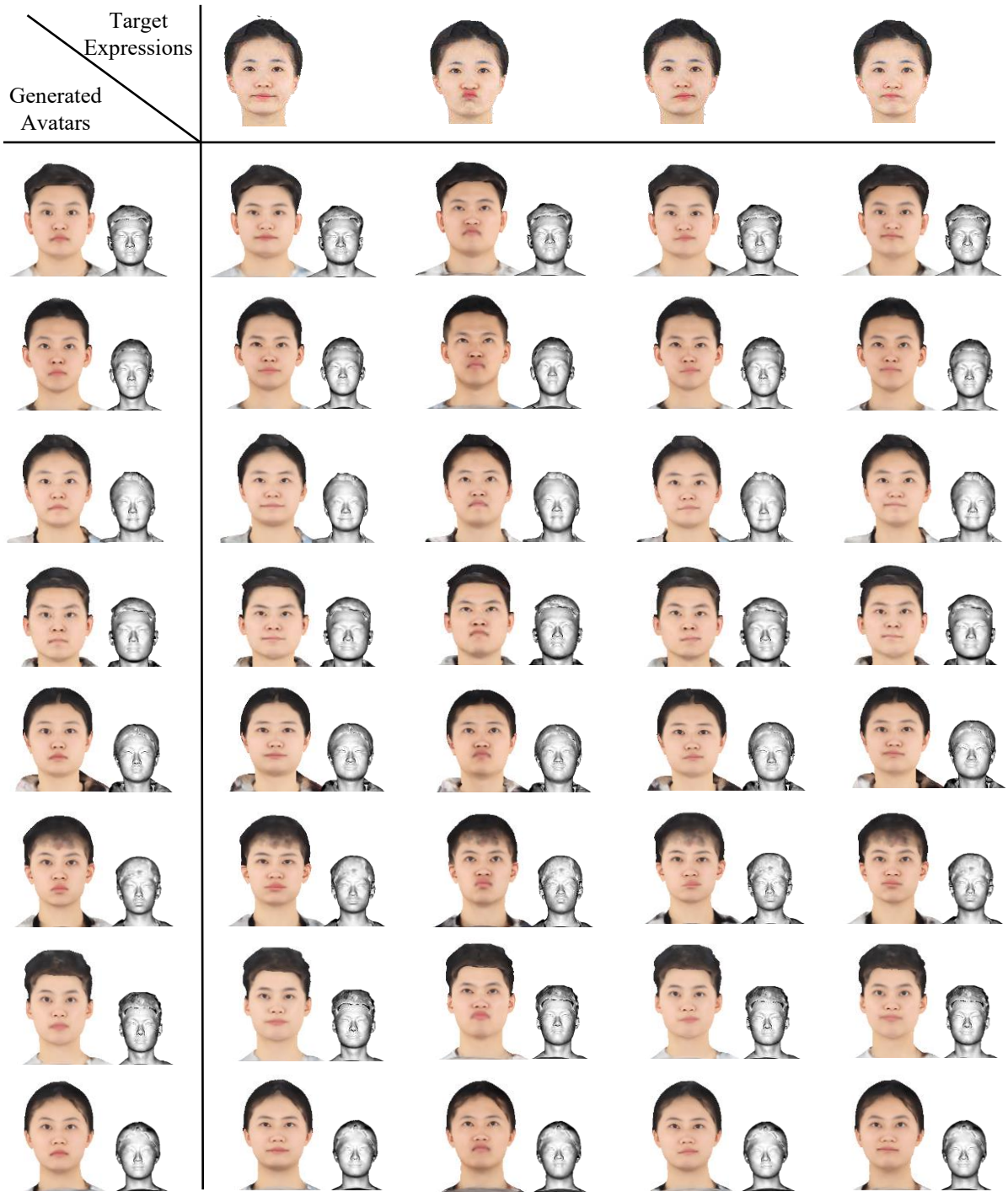

Figure 8. **Head Avatar Generation and Animation.** We randomly generate some neural head avatars, and then deform them to the target expressions. We show texture and geometry of each sample, and we encourage readers to zoom-in for details.