Efficient 3D Implicit Head Avatar with Mesh-anchored Hash Table Blendshapes Supplementary Material

Ziqian Bai[1,2*] Feitong Tan[1] Sean Fanello[1] Rohit Pandey[1]
Mingsong Dou[1] Shichen Liu[1] Ping Tan[3] Yinda Zhang[1]
[1] Google [2] Simon Fraser University [3] The Hong Kong University of Science and Technology

In this supplementary material, we provide additional method details and more results, including hash encoding details (Sec. A), warping fields details (Sec. B), network architectures (Sec. C), training and testing details (Sec. D), data statistics (Sec. E), additional experiments (Sec. F and the supplementary webpage), as well as discussions on limitations (Sec. G).

A. Hash Encoding Details

As described in Sec.3.1 and 3.2, we attach local hash table blendshapes on each 3DMM vertex, which are linearly blended with expression-dependent weights predicted via the U-Net running in UV space as the merged hash table for each vertex. The hyper-parameters of hash tables are shown in Tab. A.

Parameters	Values
Number of levels	2
Hash table size	2^8
Number of feature channels	4
Coarsest resolution	32
Finest resolution	64
Number of blendshapes per-vertex	5

Table A. Hyper-parameters for mesh-anchored hash table blendshapes.

Instead of attaching hash tables to all 3DMM vertices (*i.e.* FLAME [5] in our work), we select a subset of vertices to reduce the computation and model size, as well as ensure a more uniform vertex distribution on the 3DMM surface. More specifically, we subsample the vertices using poisson-disk sampling from meshlab [2] on a template mesh without eyeballs, and manually add 10 iris vertices, resulting in 1772 vertices in total.

*Work was conducted while Ziqian Bai was an intern at Google.

B. Warping Field Details

As described at the end of Sec.3.3, following prior works on deformable NeRF [1, 4, 9], we overfit 3D warping fields on training frames to alleviate the negative influence of misalignments between tracked 3DMM meshes and images due to the noise in 3DMM tracking and unmodeled per-frame contents such as hair movements. These warping fields are discarded during testing as in [4] and [1] since they are overfit to training frames.

More specifically, we first assign a learnable latent code $\mathbf{e}_i$ for each training frame i. Given a query point $\mathbf{q}$, we apply positional encoding on its coordinates and concatenate with the latent code $\mathbf{e}_i$, then feed them into an MLP $\mathcal{F}_{\mathcal{E}}(\cdot)$ to obtain a rigid transformation consists of 3 components: a 3D rotation $\mathbf{R} \in SO(3)$, a rotation center $\mathbf{c}^{rot}$, and a 3D translation $\mathbf{t}$. We then compute the warped query point $\mathbf{q}'$ by applying the rigid transformation to the original query point as

$$\mathbf{R}, \mathbf{c}^{rot}, \mathbf{t} = \mathcal{F}_{\mathcal{E}}\left(\mathbf{q}, \mathbf{e}_i\right) \tag{1}$$

$$\mathbf{q}' = \mathbf{R}\left(\mathbf{q} + \mathbf{c}^{rot}\right) - \mathbf{c}^{rot} + \mathbf{t}. \tag{2}$$

In practice, we represent $\mathbf{R}$ with a pure log-quaternion and directly regress it with the MLP $\mathcal{F}_{\mathcal{E}}(\cdot)$. As described in Sec.3.3, we denote this full warping procedure as $\mathbf{q}' = \mathcal{T}_i(\mathbf{q}) = \mathcal{F}_{\mathcal{E}}(\mathbf{q}, \mathbf{e}_i)$. The warped query point $\mathbf{q}'$ is then used to compute the density and color for volumetric rendering.

C. Network Architectures

Here we introduce the detailed network architectures of three main components described in Sec.3.2 and 3.3: the U-Net running in UV space, the MLP to predict densities and colors, and a warp field MLP predictor. Please refer to the main paper for details on how these components come together to form our avatar model.

C.1. U-Net running in UV space

As described in Sec.3.2, the U-Net running in UV space takes the 3DMM vertex displacements as the input and outputs expression-dependent weights (to weighted sum hash tables) and per-vertex features. For the encoder side, we use downsampling residual blocks to extract a feature pyramid with the number of channels for each level as $\{8, 16, 32, 64, 128, 256\}$, with 128 as the input resolution and downsample to 64, 32, 16, 8, 4, 2. In the decoder side, we use upsampling residual blocks (*i.e.* with transposed convolutions) and set the number of output channels for each level to 128, 64, 64, 64, 64, 64. Finally, we use a 1×1 convolution layer to get the weights map and the feature map. The leaky ReLU is applied after each convolutional layer with a slope 0.2. The input vertex displacement map has a resolution of 128×128. The output expression-dependent weights map has a size of $128 \times 128 \times 5$ (4 channels predicted by network, 1 channel set to a constant one) and the output feature map has a size of $128 \times 128 \times 24$. The weights map and the feature map are then sampled back to 3DMM vertices as described in Sec.3.2.

C.2. MLP for Densities and Colors

As described in Sec.3.3, for each query point, we use the tiny MLP to decode densities and colors from the summed hash table embedding $\overline{\mathbf{h}}_i$, the nearest per-vertex feature $\mathbf{f}_{ik^*}$, the query point tangent coordinates $\mathbf{q}_{ik^*}$ of the nearest vertex applied with positional encoding, and the camera view direction with positional encoding. The tiny MLP consists of two hidden layers, where each hidden layer contains a Fully Connected layer with ReLU activation and 64 neurons. Please refer to Sec.3.3 and Fig.3 for more pipeline details. For the positional encoding, we use 8 frequency bands on the query point tangent coordinates, and 4 frequency bands on the camera view direction.

C.3. Warp Field MLP

The warp field MLP $\mathcal{F}_{\mathcal{E}}$ described in Sec.3.3 and Sec. B consists of a backbone and three output branches. The backbone contains 5 hidden layers, where each layer has 128 neurons. Then, we append three branches, each is a 2-layer MLP with 128 neurons width, for regressing the three outputs described in Sec. B: pure log-quaternion of the 3D rotation (*i.e.*, SO(3)) $\mathbf{R}$, rotation center $\mathbf{c}^{rot}$, and 3D translation $\mathbf{t}$. ReLU activation is used in all layers except the output layers. We adapt a coarse-to-fine positional encoding strategy as used in Nerfies [8] on the query point coordinates before feeding into the MLP $\mathcal{F}_{\mathcal{E}}$ for better training stability. We start with 0 frequency bands and increase to 6 linearly after 80k training iterations.

D. Training and Testing Details

To obtain a consistent 3D world space, we normalize the 3DMM meshes with their neck poses to align the head in 3D space. During training, we use a hierarchical sampling strategy as in [6], where we use 32 coarse and 32 fine sample points per ray. During testing, we obtain a union occupancy grid for all training expressions, and run ray marching on those valid voxels to achieve efficient rendering. To ensure stable training, we enable 3D warping fields after 5k iterations. During training, we use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$. In each mini-batch, we random sample 256 rays from 8 images (*i.e.* 2048 rays in total) and set the learning rates to: (1) 10^{-4} for the warping field MLP and exponentially decay to 10^{-5} after 400k. (2) $5 * 10^{-4}$ for other neural networks and exponentially decay to $5 * 10^{-5}$ after 400k. We train the model with 400k iterations for each subject.

E. Data Settings

In Tab. B, we show more details on our data statistics over 10 subjects.

	Number of Train Frames	Number of Test Frames	Resolution
subject0	1560	434	(512, 402)
subject1	1480	740	(512, 422)
subject2	1440	603	(512, 380)
subject3	1360	564	(512, 368)
subject4	1450	304	(512, 398)
subject5	2655	595	(512, 372)
subject6	1818	696	(512, 452)
subject7	3912	817	(512, 512)
subject8	2656	898	(512, 344)
subject9	2049	351	(512, 512)

Table B. Data statistics over 10 subjects.

F. Additional Experiments

F.1. Additional Results

In this section, we provide additional experimental results and comparisons with prior state-of-the-art methods. Please see Fig. A and Fig. B for qualitative results and the accompanying **supplementary webpage** for video results.

In Fig. A and Fig. B, we show more image results comparing with prior state-of-the-art approaches. PointAvatar [12] and INSTA [13] give overall inferior renderings than ours due to their limited model capacities in capturing static (*e.g.*, glasses, hairs) and dynamic (*e.g.*, expression-dependent deformations and wrinkles) avatar details. NeRFBlendshape [3] produces less stable results, leading to severe artifacts around mouth and obvious floaters on avatar boundaries. MonoAvatar [1] stably generates high quality renderings and animations, but is much slower than our method (*i.e.*, 0.5 FPS vs. 35.9 FPS) and slightly smoother on some details, for example, hairs, teeth and wrinkles. Our method overall achieves one of the best rendering quality, . Please refer to Sec. G for more discussions on our limitations.

In the accompanying **supplementary webpage**, we include video results on various subjects with side-by-side comparisons to prior state-of-the-art methods, including PointAvatar [12], INSTA [13], NeRFBlendshape [3], and MonoAvatar [1]. The videos show that our method is able to produce high-quality renderings while maintaining real-time speed.

F.2. Comparisons to More Works

Here, we provide comparisons to more state-of-the-art methods that deliver relatively fast solutions for (partially) volumetric head avatar, including AvatarMAV [10] and LatentAvatar [11]. AvatarMAV [10] represents the head avatar by feature grid blendshapes to achieve fast training. LatentAvatar [11] learns a neural expression latent space instead of using 3DMM expression codes, and generate triplanes from this expression latent space. The triplane is rendered into a low resolution feature map, which is then used to synthesis the output RGB images via a 2D CNN.

As shown in Tab. C, our method is able to achieve comparable rendering quality with these SOTA approaches, while supporting real-time rendering simultaneously. Note that LatentAvatar [11] uses heavy CNNs to directly synthesis output images, which leads to good sharpness (shown by LPIPS) but temporally 3D inconsistent high-frequency details. Also, directly synthesis image with CNN is an orthogonal direction to our method, which can also be appended to our pipeline.

	LPIPS	SSIM	PSNR	Mean FPS
AvatarMAV [10]	0.128	0.792	23.51	2
LatentAvatar [11]	0.092	0.763	21.94	16
Ours	0.100	0.795	22.77	35.9

Table C. Quantitative comparisons with more state-of-the-art approaches. Our method achieves comparable rendering quality, while supports real-time rendering with a 512×512 resolution.

F.3. Ablation on Discarding Hash Tables

Here, we investigate a new ablation setting *No Hash + UV CNN*, where we discard all hash tables while keeping other parts unchanged. In this way, our model decodes the neural radiance field thoroughly from the vertex-attached features as in MonoAvatar [1] but with a much smaller MLP for fast rendering. This gives the following results: 0.164 / 0.759 / 21.57 for LPIPS / SSIM / PSNR, which are largely inferior than our full model *Ours*. This indicates that the local hash tables are important for boosting the model capacity to achieve photorealism rendering quality.

F.4. Geometry Visualization and Analysis

For the purpose of a comprehensive system analysis, we visualize the resulting geometry (as normal maps) of our avatars and compare with prior state-of-the-art approaches. Fig. C shows the normal map visualization. PointAvatar [12] gives smooth geometry estimation thank to their relighting formulation. But their renderings are overall blurrier than other methods. INSTA [13] generates geometries closing to the 3DMM meshes since they regularize the NeRF depth to the rasterized 3DMM depth on face region, which also leads to incorrect shapes for beard. Moreover, their rendered images are also suffered from unsatisfying quality. Despite NeRFBlendshape [3] gives relatively good renderings, their estimated geometry is very noisy, presumably because that they do not leverage the 3DMM mesh as a shape prior. MonoAvatar [1] gives both decent geometries and renderings, but is one order of magnitudes slower than real-time speed. Our method gives reasonable geometries that are slightly noisier than MonoAvatar, but supports real-time rendering speed while maintaining decent image quality.

G. Limitations

Comparing with the state-of-the-art high quality MonoAvatar [1], our method is facing some quality trade-offs. On the positive side, our method captures more high frequency details such as hairs, teeth, and wrinkles thank to the high flexibility of hash table embeddings. However, this also introduces slightly more floaters than MonoAvatar [1] (Fig. D), which is a common issue for methods based on instant NGPs [7]. Presumably due to the same reason as well as poor tracking, we also observe a slightly less stale performance around the mouth interior regions and thin structures such as glasses frames. Further improving the robustness and stability without hurting quality and speed is an interesting future direction to explore.

References

[1] Ziqian Bai, Feitong Tan, Zeng Huang, Kripasindhu Sarkar, Danhang Tang, Di Qiu, Abhimitra Meka, Ruofei Du, Mingsong Dou, Sergio Orts-Escolano, et al. Learning personalized high quality volumetric head avatars from monocular rgb videos. In *CVPR*, pages 16890–16900, 2023. 1, 2, 3, 4, 5, 6

[2] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 1

[3] Xuan Gao, Chenglai Zhong, Jun Xiang, Yang Hong, Yudong Guo, and Juyong Zhang. Reconstructing personalized semantic facial nerf models from monocular video.

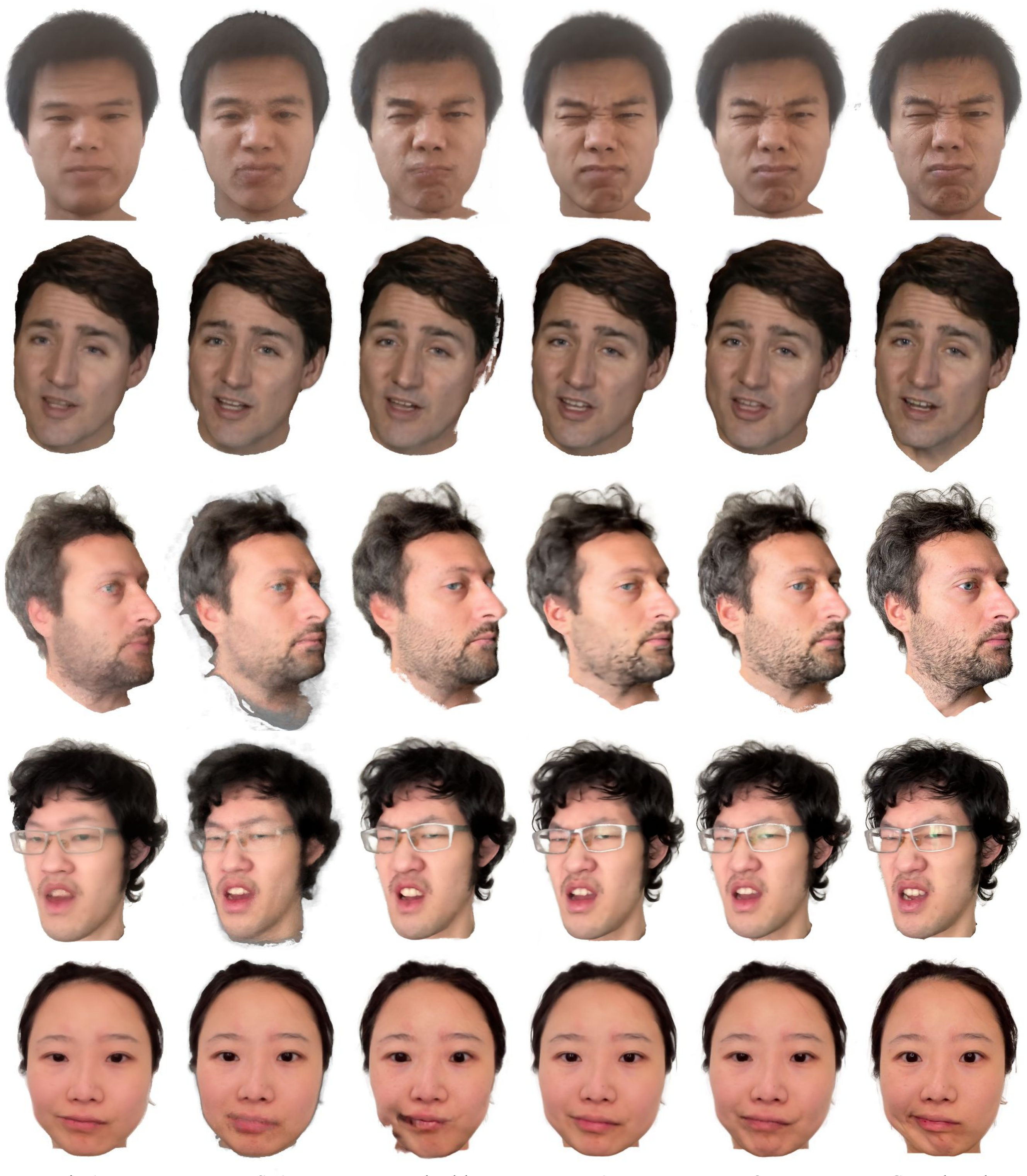

Figure A. Comparisons on the rendering quality to previous state-of-the-art methods. From left to right, each column contains the images of: 1) PointAvatar [12], 2) INSTA [13], 3) NeRFBlendshape [3], 4) MonoAvatar [1], 5) Ours, 6) Ground Truth. Our method faithfully reconstructs the personalized expressions and high-frequency details, achieving one of the best rendering quality with real-time rendering speed.

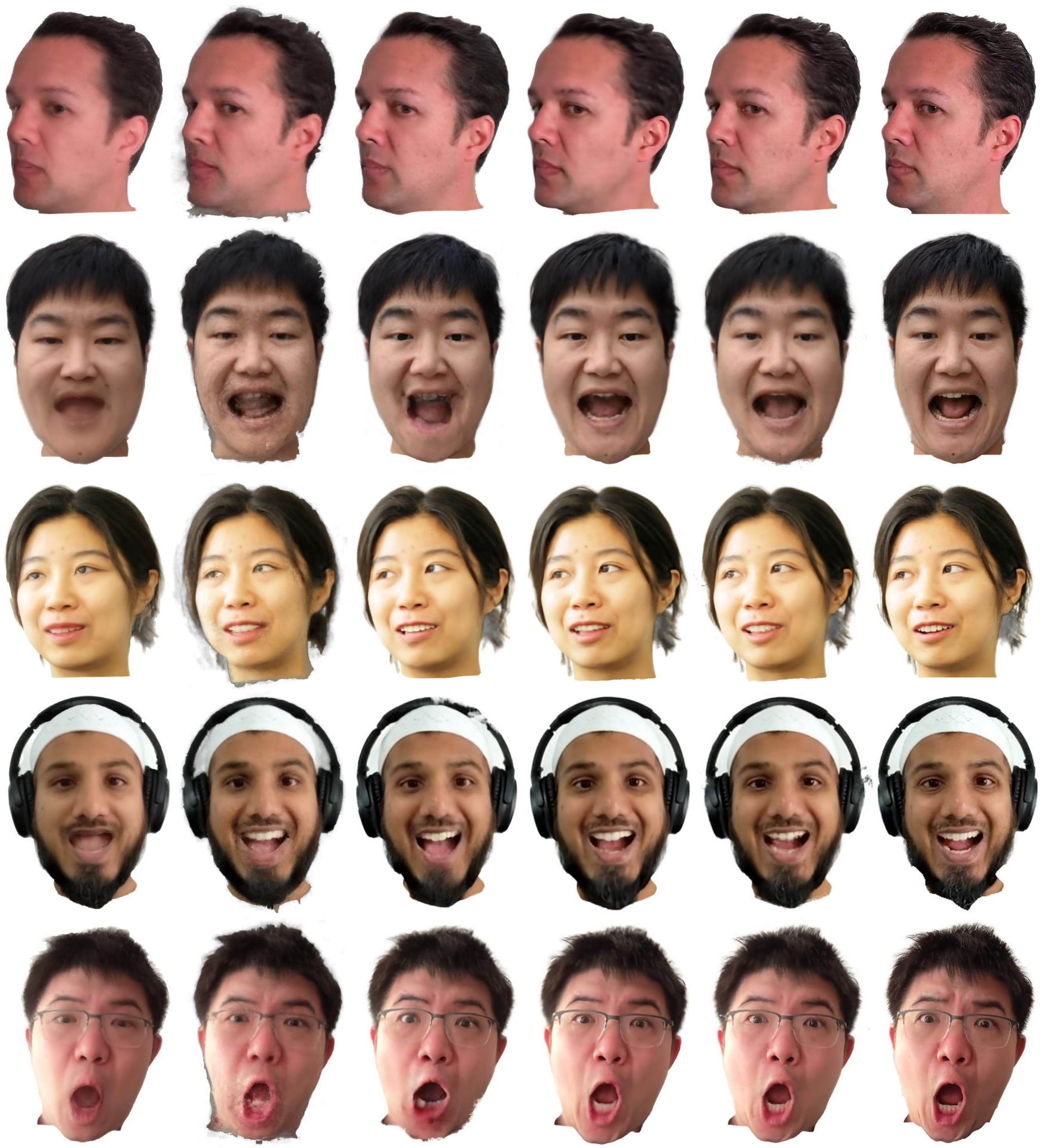

Figure B. Comparisons on the rendering quality to previous state-of-the-art methods. From left to right, each column contains the images of: 1) PointAvatar [12], 2) INSTA [13], 3) NeRFBlendshape [3], 4) MonoAvatar [1], 5) Ours, 6) Ground Truth. Our method faithfully reconstructs the personalized expressions and high-frequency details, achieving one of the best rendering quality with real-time rendering speed.

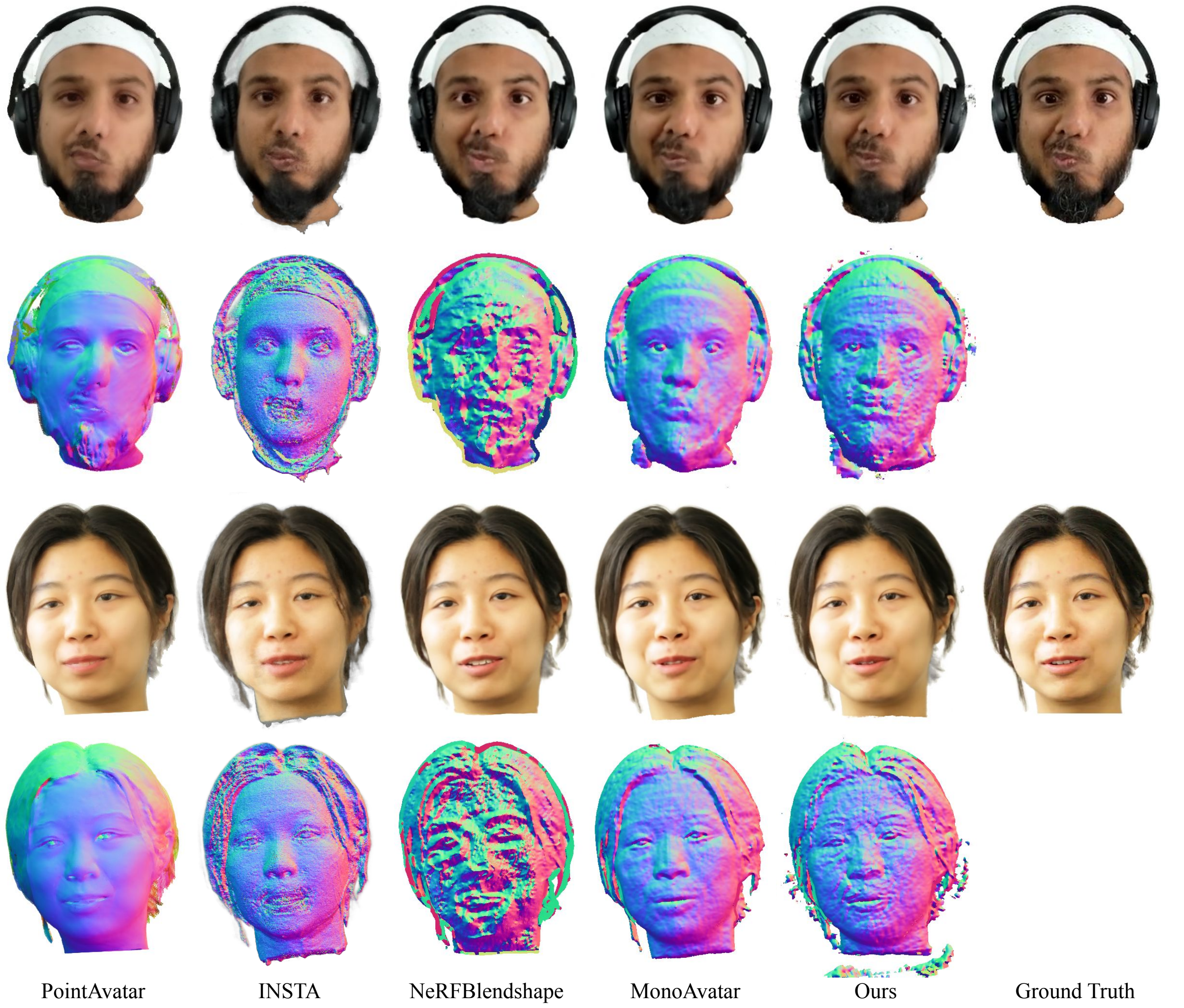

Figure C. Visualization of normal maps and compare to previous state-of-the-art methods. From left to right, each column contains the images of: 1) PointAvatar [12], 2) INSTA [13], 3) NeRFBlendshape [3], 4) MonoAvatar [1], 5) Ours, 6) Ground Truth. Our method estimates reasonable geometries, while achieving one of the best rendering quality with real-time rendering speed.

ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia), 41(6), 2022. 2, 3, 4, 5, 6

[4] Wei Jiang, Kwang Moo Yi, Golnoosh Samei, Oncel Tuzel, and Anurag Ranjan. NeuMan: Neural Human Radiance Field From a Single Video. 2022. 1

[5] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a Model of Facial Shape and Expression From 4D Scans. *ACM TOG*, 36(6):194:1–194:17, 2017. 1

[6] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes As Neural Radiance Fields for View Synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2

[7] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 3

[8] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable Neural Radiance Fields. pages 5865–5874, 2021. 2

[9] Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Hu-

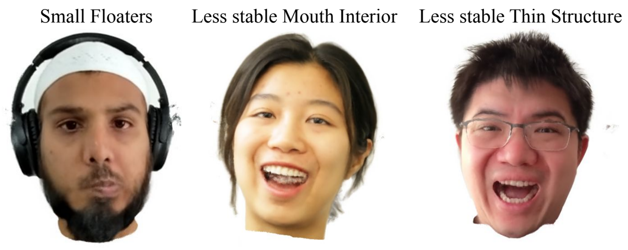

Figure D. Due to the high flexibility of hash table embeddings and poor tracking on mouth interiors, we observe minor artifacts of our method, including small floaters, less stable mouth interiors and thin structures.

mannerf: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16210–16220, 2022. 1

[10] Yuelang Xu, Lizhen Wang, Xiaochen Zhao, Hongwen Zhang, and Yebin Liu. Avatarmav: Fast 3d head avatar reconstruction using motion-aware neural voxels. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023. 3

[11] Yuelang Xu, Hongwen Zhang, Lizhen Wang, Xiaochen Zhao, Han Huang, Guojun Qi, and Yebin Liu. Latentavatar: Learning latent expression code for expressive neural head avatar. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023. 3

[12] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J Black, and Otmar Hilliges. Pointavatar: Deformable point-based head avatars from videos. In *CVPR*, pages 21057–21067, 2023. 2, 3, 4, 5, 6

[13] Wojciech Zielonka, Timo Bolkart, and Justus Thies. Instant volumetric head avatars. In *CVPR*, pages 4574–4584, 2023. 2, 3, 4, 5, 6