Supplementary for HyperGaussians: High-Dimensional Gaussian Splatting for High-Fidelity Animatable Face Avatars

Gent Serifi Marcel C. Buehler

ETH Zurich, Switzerland

https://gserifi.github.io/HyperGaussians

This supplement contains more details and derivations in Secs. 2 and 3, supplementary results in Sec. 4, and discusses the potential societal impact of this work in Sec. 5.

1. Positioning with respect to Related Works

We provide an overview of the main differences between HyperGaussians and the most closely related works on Gaussian avatars in Tab. 1. We partition the methods into 3 categories: optimization-based, feed-forward, and pseudo ground-truth. Since the latter two categories require substantial computational resources for pre-training, we mainly focus on optimization-based techniques in our experiments.

2. HyperGaussian Details

This section provides more details about the formulation of HyperGaussians.

2.1. Parameterization

Each HyperGaussian consists of a high-dimensional mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ (or precision matrix $\boldsymbol{\Lambda}$ when applying the *inverse covariance trick*, Sec. 2.3) with optimizable parameters. We parameterize the mean $\boldsymbol{\mu}$ directly and decompose the covariance $\boldsymbol{\Sigma}$ into its Cholesky factor $\boldsymbol{L}$ [3], such that $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$, where $\boldsymbol{L}$ is a lower triangular matrix with positive diagonal entries. To ensure the uniqueness of the factorization, we apply an exponential activation function, $\boldsymbol{L}_{i,i} \leftarrow \exp \boldsymbol{L}_{i,i}$, to the diagonal entries of the parameter matrix. We further show in Sec. 2.3 that it is not necessary to parameterize the entire Cholesky factor $\boldsymbol{L}$ when working with the inverse covariance trick (Eqs. (5) and (6)).

2.2. Splatting

This paragraph explains how we construct the 3D covariance matrix for splatting the conditioned Gaussians. We build on top of the same decomposition of $\boldsymbol{\Sigma}_{\text{3D}}$ into rotation and scaling as in vanilla 3DGS, but replace the static terms $\boldsymbol{R}$ and $\boldsymbol{S}$ by their conditional analogues. More formally,

$$\boldsymbol{\Sigma}_{\text{3D}} = \boldsymbol{R}(\bar{\boldsymbol{q}})\bar{\boldsymbol{S}}\bar{\boldsymbol{S}}^\top \boldsymbol{R}(\bar{\boldsymbol{q}})^\top, \tag{1}$$

where $\boldsymbol{R}(\cdot)$ denotes quaternion to rotation matrix conversion, $\bar{\boldsymbol{q}} = \mathbb{E}[\mathcal{A}_{\boldsymbol{q}}|\boldsymbol{z}]$, and $\bar{\boldsymbol{S}} = \text{diag}\big(\mathbb{E}[\mathcal{A}_{\boldsymbol{s}}|\boldsymbol{z}]\big)$. Moreover, $\boldsymbol{z}$ is the latent code and $\mathcal{A}$ stands for *attribute* (see Eq. (4) in the main paper).

This is an important distinction to NDGS [3], which directly splats using the conditional covariance matrix $\boldsymbol{\Sigma}_{\text{3D}} = \text{Cov}[\mathcal{A}_{\boldsymbol{\mu}_{\text{3D}}}|\boldsymbol{z}]$. NDGS has no degrees of freedom for the conditional orientation of the 3D Gaussians, *i.e.*, the 3D Gaussians cannot conditionally rotate.

To illustrate this, note that conditioning can be interpreted geometrically as taking an m-dimensional slice through the multivariate Gaussian with $(m + n)$ total dimensions. Fig. 1 shows two examples for $m = n = 1$. Notice that the conditional covariance matrix has the same shape regardless of where the slice is taken (Eq. (3) in the main paper).

We instead use $\text{Cov}[\mathcal{A}_{\boldsymbol{\mu}_{\text{3D}}}|\boldsymbol{z}]$, $\text{Cov}[\mathcal{A}_{\boldsymbol{q}}|\boldsymbol{z}]$, and $\text{Cov}[\mathcal{A}_{\boldsymbol{s}}|\boldsymbol{z}]$ for visualizing uncertainties (see Fig. 2).

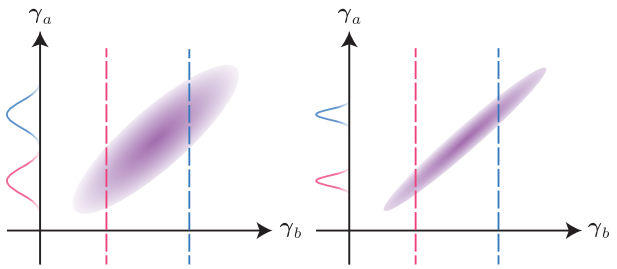

Figure 1. **Geometric interpretation of Gaussian conditioning** on two examples with large (left) and small (right) uncertainty at different realizations of $\boldsymbol{\gamma}_b$. As a result, the conditional mean shifts, while the conditional covariance is the same for both slices.

Method	Representation	Local context	Dynamic context	Resource requirements
GaussianAvatars [14]	3DGS	×	×	Low
SurFHead [9]	2DGS	×	×	Low
SplattingAvatar [15]	3DGS	✓	×	Low
MonoGaussianAvatar [1]	3DGS	×	✓	Moderate
FlashAvatar (FA) [20]	3DGS	×	✓	Low
GaussianHeadAvatar (GHA) [21]	3DGS	×	✓	Moderate
GAGAvatar [2]	FF 3DGS	✓	✓	High
LAM [5]	FF 3DGS	✓	×	High
FastAvatar [11]	FF 3DGS	✓	×	High
Avat3r [17]	FF 3DGS	✓	✓	High
GAF [17]	MVDiffusion 3DGS	×	×	Very High
CAP4D [18]	MVDiffusion 3DGS	×	×	Very High
Ours (FA)	HGS	✓	✓	Low
Ours (GHA)	HGS	✓	✓	Moderate

Table 1. **Differences to the most closely related works.** GaussianAvatars [14] and SuRFHead [9] deform 3D Gaussians based on an underlying FLAME mesh [10] without local embeddings or dynamic inputs like facial expressions. SplattingAvatar [15] optimizes local embeddings, but the Gaussian properties are not dependent on expressions or pose. MonoGaussianAvatar [1], FlashAvatar [20], and GaussianHeadAvatar [21] predict expression-dependent offsets to the Gaussian properties, but their lack of local context leads to blurry or distorted results, see comparison in the main paper. Our proposed representation (HGS) attaches high-dimensional Gaussians to the mesh and optimizes learnable local embeddings for modulating the Gaussian properties based on expressions. A more recent line of work deploys large-scale generative models to either directly regress 3D Gaussian parameters or augment the training signal during optimization. GAGAvatar [2] and LAM [5] derive 3D Gaussians from DINOv2 [12] features. FastAvatar [11] and Avat3r [8] train an encoder-decoder model from scratch, where Avat3r additionally uses position maps from DUSt3R [19] and feature maps from Sapiens [6]. GAF [17] and CAP4D [18] train multi-view diffusion models [4, 13, 16] to generate pseudo ground-truth images for 3DGS optimization. These feed-forward and pseudo GT methods show promising results in sparse-view settings, their primary focus. However, they lose the ability to leverage all information when more views are available. The typical range is 1 to 4 images per forward pass. CAP4D proposes a sophisticated inference scheme to address this issue. However, it still takes 12 GPU hours to produce a single avatar. Traditional optimization-based techniques still achieve superior performance at significantly lower computational effort in such cases.

2.3. Derivation of the Inverse Covariance Trick

We explain in the main paper that a naïve implementation of the conditioning is very inefficient for large latent codes $\boldsymbol{\gamma}_b$. Here, we provide a more detailed derivation of the inverse covariance trick.

We reformulate our HyperGaussians in terms of their precision matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ such that $\boldsymbol{\gamma} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$. We consider the following block matrix view:

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{bmatrix} \tag{2}$$

with $\boldsymbol{\Lambda}_{ba} = \boldsymbol{\Lambda}_{ab}^\top$.

As the inverse of $\boldsymbol{\Sigma}$, $\boldsymbol{\Lambda}$ inherits symmetry, Eq. (3), as well as positive definiteness since its eigenvalues are the reciprocal of the eigenvalues of $\boldsymbol{\Sigma}$, and therefore all positive:

$$\boldsymbol{\Lambda}^\top = \left(\boldsymbol{\Sigma}^{-1}\right)^\top = \left(\boldsymbol{\Sigma}^\top\right)^{-1} = \boldsymbol{\Sigma}^{-1} = \boldsymbol{\Lambda}. \tag{3}$$

This is important, as it allows us to reuse the same parameterization that we described in Sec. 2.1 to represent $\boldsymbol{\Lambda} = \boldsymbol{L}\boldsymbol{L}^\top$. Conveniently, we also get the Cholesky factor $\boldsymbol{L}_{11}$ of $\boldsymbol{\Lambda}_{aa}$ as a side product:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{L}_{11} & \boldsymbol{0} \\ \boldsymbol{L}_{21} & \boldsymbol{L}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{L}_{11}^\top & \boldsymbol{L}_{21}^\top \\ \boldsymbol{0} & \boldsymbol{L}_{22}^\top \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{L}_{11}\boldsymbol{L}_{11}^\top & \boldsymbol{L}_{11}\boldsymbol{L}_{21}^\top \\ \boldsymbol{L}_{21}\boldsymbol{L}_{11}^\top & \boldsymbol{L}_{21}\boldsymbol{L}_{21}^\top + \boldsymbol{L}_{22}\boldsymbol{L}_{22}^\top \end{bmatrix}. \end{aligned} \tag{4}$$

With this new formulation, the conditional mean and covariance matrix can be expressed as

$$\begin{aligned} \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}\left(\boldsymbol{\gamma}_b - \boldsymbol{\mu}_b\right) \\ \boldsymbol{\Sigma}_{a|b} &= \boldsymbol{\Lambda}_{aa}^{-1}, \end{aligned} \tag{5}$$

where the term $\boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}$ can be further broken down into

$$\boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab} = \left(\boldsymbol{L}_{11}\boldsymbol{L}_{11}^\top\right)^{-1}\boldsymbol{L}_{11}\boldsymbol{L}_{21}^\top = \boldsymbol{L}_{11}^{-\top}\boldsymbol{L}_{21}^\top. \tag{6}$$

Note that products with $\boldsymbol{L}_{11}^{-\top}$ can be evaluated in an efficient and numerically stable manner since $\boldsymbol{L}_{11}$ is a triangular matrix of small size, which is independent of the latent dimension. These equations also show that it is sufficient only to parameterize $\boldsymbol{L}_{11}$ and $\boldsymbol{L}_{21}$ for modeling the conditional distribution. This reduces the number of parameters

from $\mathcal{O}\big((m+n)^2\big)$ to $\mathcal{O}\big(m^2+mn\big)$ compared to the full parameterization. Please see the main paper for a benchmark comparison between the naïve implementation and the one applying the inverse covariance trick.

2.4. Derivation of Uncertainty

We observe an interesting property about HyperGaussians, which arises naturally from their Bayesian interpretation. HyperGaussians are at their core multivariate Gaussian distributions. Their conditional covariance matrices indicate the variance of each Gaussian across the different expressions of the training subject and can be intuitively interpreted as uncertainty.

More formally defined, we have

$$\begin{aligned}\sigma &:= \log\det \boldsymbol{\Sigma}_{a|b} \\ &= -\log\det \boldsymbol{\Lambda}_{aa} \quad (1) \\ &= -2\log\det \boldsymbol{L}_{11} \quad (2) \\ &= -2\,\mathrm{tr}\log \boldsymbol{L}_{11}, \quad (3)\end{aligned} \tag{7}$$

where we used $\det \boldsymbol{\Sigma}_{a|b} = \det \boldsymbol{\Lambda}_{aa}^{-1} = \big(\det \boldsymbol{\Lambda}_{aa}\big)^{-1}$ in step (1), $\det \boldsymbol{\Lambda}_{aa} = \det \boldsymbol{L}_{11}\boldsymbol{L}_{11}^\top = \big(\det \boldsymbol{L}_{11}\big)^2$ in step (2), and $\det \boldsymbol{L}_{11} = \prod_{i=1}^{m}(\boldsymbol{L}_{11})_{i,i}$ in step (3). The log in step (3) is applied element-wise.

Again, the inverse covariance trick (Sec. 2.3) is key to efficiently compute this quantity. These values are summed up across the conditional distributions for all Gaussian attributes. To render these uncertainties, we further apply a sigmoid function and map the values to colors. This agreement between the uncertainty estimates and what would intuitively be considered difficult regions emerges without explicit supervision. We demonstrate an example of this effect in Fig. 2. Note that the foreground mask in the bottom-left neck area was unstable throughout the video, leading to high uncertainty despite being rigid. Moreover, the uncertainty correctly captures the variability of the lips and specular reflections on glasses. We further observe that, while rigid, the glass frames exhibit significant displacements due to underlying mesh deformations.

3. GaussianHeadAvatar Details

GaussianHeadAvatar (GHA) [21] represents a scene with N Gaussians, which have position X, multi-channel color C, rotation Q, scale S and opacity A. The way GHA models expression-dependent effects is by maintaining a canonical set of Gaussians $\{X_0, F_0, Q_0, S_0, A_0\}$, where F_0 is a per-point feature vector, and training an MLP-based expression conditioning dynamic generator Φ, which predicts dynamic changes with respect to the canonical model. More specifically, given expression θ and head pose β, they compute

$$\{X, C, Q, S, A\} = \Phi(X_0, F_0, Q_0, S_0, A_0; \theta, \beta) \tag{8}$$

Figure 2. **Uncertainty quantification** on one of the training subjects. Green denotes low uncertainty, while Red denotes high uncertainty. Note that the semantic structure arises purely from the probabilistic formulation without additional supervision.

in order to obtain a deformed set of Gaussians.

For our HyperGaussians integration, we modify the pipeline such that the MLP outputs per-Gaussian latent codes for each attribute, which are then used to condition the HyperGaussians. Formally,

$$\{\boldsymbol{z}_X, \boldsymbol{z}_C, \boldsymbol{z}_Q, \boldsymbol{z}_S, \boldsymbol{z}_A\} = \Phi(X_0, F_0, Q_0, S_0, A_0; \theta, \beta) \tag{9}$$

are the expression-dependent latents, which are then used for computing the conditional means $\mathbb{E}[\mathcal{A}|\boldsymbol{z}]$ that are fed to the remainder of the pipeline, just like the MLP offsets in the original method, analogous to our FlashAvatar integration.

4. Supplementary Experiments

NeRSemble Details For our multi-view setting, we used 10 subjects from the NeRSemble [7] dataset, with the following IDs: 017, 018, 024, 031, 033, 036, 037, 129, 141, 144. We select all sequences labelled with `EMO`, `EXP` (excluding tongue sequences), `SEN` for training, and the `FREE` sequences for testing.

4.1. Qualitative Results

We show video results for self- and cross-reenactment on the supplementary HTML page. In addition, Figs. 3 and 4 show additional results against NDGS [3] after integration into the baseline methods, similar to HyperGaussians. Moreover, Fig. 5 provides qualitative results for varying latent dimensionalities.

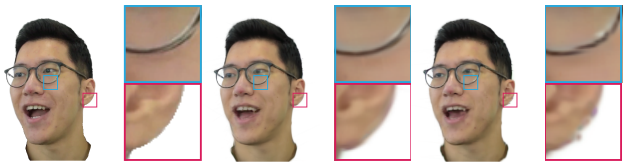

Figure 3. **Comparison with NDGS** integrated into FlashAvatar. The limited degrees of freedom in NDGS lead to misalignments of thin structures and edges. The numbers show PSNR, SSIM (10^{-1}), and LPIPS (10^{-2}).

4.2. Comparison with NDGS

Our approach differs from NDGS [3] in its mathematical formulation and capabilities. While NDGS uses the conditional covariance matrix, which is *independent* of the latent code, to represent the size and shape directly, our HyperGaussians apply multivariate Gaussians on each attribute where the *conditional means dynamically adapt the location, scale, and orientation* of the derived 3D Gaussians *in response to* the latent code. This crucial difference gives HyperGaussians the necessary degrees of freedom to model thin geometry and complex deformations with higher accuracy. Fig. 3 shows our method eliminates the artifacts visible in NDGS (particularly on glass frames) and produces substantially more precise geometry at curved boundaries. We further demonstrate the benefits of HyperGaussians over NDSG by integrating them into GaussianHeadAvatar [21] in a similar fashion. Again, we can observe in Fig. 4 that NDGS produces inferior reconstructions compared to HyperGaussians. More specifically, it lacks detail for specular reflections on the glasses and teeth (row 3), and it fails to accurately position the glasses at the correct vertical location (a downward shift in row 4 due to surrounding skin deformations). Moreover, the bottom-right wrinkles in row 4 disappear when the expression latent deviates too far from the mean, since NDGS uses the joint probability density to modulate opacity. These observations can also be made quantitatively in Tab. 2.

Method	PSNR ↑	SSIM(10^{-1}) ↑	LPIPS(10^{-2}) ↓
GaussianHeadAvatar (GHA) [21]	24.10	8.819	20.273
NDGS (GHA) [3]	24.22	**8.820**	20.235
Ours (GHA)	**24.38**	8.819	**19.768**

Table 2. **Quantitative comparison** with GHA and NDGS in the multi-view setting. Notice that NDGS is unable to match the improvements of HyperGaussians in terms of PSNR and LPIPS, and performs only marginally better on SSIM. This highlights the limited capabilities of NDGS due to its reduced degrees of freedom.

4.3. Ablation Study

We complement the ablation study from the main paper with supplementary results for different MLP configurations in Tab. 3 and Fig. 5. The default FlashAvatar MLP [20] has 6 layers with 256 neurons, totaling 375K parameters. Replacing vanilla 3D Gaussians with HyperGaussians adds optimizable parameters (see Sec. 2.1). One might assume that simply increasing the parameter count for the FlashAvatar MLP would improve the results. However, this is not the case. We ablate different MLP configurations in Tab. 3. Adding more parameters to the MLP does not perform as well as adding HyperGaussian. In fact, it performs the same, or worse than the baseline, while significantly slowing down rendering speed. FlashAvatar with vanilla 3DGS runs at 347 FPS. With a large MLP, this number drops to 158 (for 256×40, 2.6M parameters) and 178 (for 512×11, 2.7M parameters). With HyperGaussians ($n = 8$ and 2.6M parameters), the original MLP (256×6) outperforms the other MLP variants for all metrics while maintaining a rendering speed of 300 FPS. All metrics and rendering times were computed on a single NVIDIA GeForce RTX 2080 Ti for images with resolution 512×512. In summary, the HyperGaussians' performance improvement cannot be matched by increasing the complexity of the MLP. HyperGaussians boost the performance while maintaining fast rendering speed. Moreover, while FlashAvatar with vanilla 3DGS achieves higher FPS, we observe faster convergence with HyperGaussians. Fig. 6 visualizes the training progress over time and shows that HyperGaussians achieve a higher quality for a given time budget.

A fundamental advantage of HyperGaussians is their ability to distill highly local context, enabling independent deformations between spatially proximate but semantically distinct regions. For instance, our method can independently model glass frames near the upper cheek or the upper teeth adjacent to the jaw. In contrast, FlashAvatar suffers from stronger coupling between neighboring Gaussians due to its shared MLP architecture and direct offset approach. This coupling creates an optimization challenge where improvements in one region often degrade quality in others. Our approach allows each region to optimize independently, preserving detailed geometry and appearance across semantically different but spatially adjacent facial features.

We ablate the effect of different latent dimensionalities in Tab. 3. We find that HyperGaussian are robust towards different latent dimensions. A latent dimension of 8 performs very well, but we already observe an improvement for a single latent dimension ($n = 1$) over the vanilla 3DGS variant. The top row corresponds to the FlashAvatar baseline [20], which does not use any HyperGaussians.

MLP (width×depth)	HGS-Dim.	# Param.	FPS	PSNR ↑	SSIM (10^{-1}) ↑	LPIPS (10^{-2}) ↓
256×6	-	375K	347	29.43	9.466	5.107
256×40	-	2.6M	158	28.10	9.380	5.720
512×11	-	2.7M	178	29.50	9.472	5.122
256×6	1D	1.2M	291	29.73	9.492	5.066
256×6	2D	1.4M	304	29.92	9.503	5.000
256×6	4D	1.8M	293	29.89	9.507	4.994
256×6	8D	2.6M	300	29.99	9.510	4.978
256×6	16D	4.1M	298	29.92	9.511	4.978
256×6	32D	7.2M	281	29.89	9.511	4.976
256×6	64D	13.4M	273	29.91	9.512	5.020

Table 3. **MLP and Latent Dimensionality Ablations.** Simply increasing the parameter count for the FlashAvatar MLP does not improve the metrics. Our HyperGaussians, however, improve the performance of the original MLP out-of-the-box. As an additional benefit, HyperGaussians render at around 300 FPS while the deeper MLPs, with comparable parameter counts, drop to 158 FPS (256×40) and 178 FPS (512×11), respectively. Note that the drop of 10 FPS for 1D is likely due to memory bottlenecks caused by poor cache and vector load/store locality. Green denotes the best and Yellow the second best.

5. Societal Impact

It is important to be aware that photorealistic, high-quality face avatars from monocular videos can have societal implications. While our novel HyperGaussian representation offers exciting possibilities for entertainment, communication, and virtual experiences, it could potentially be misused to spread misinformation and deception. Realistic face avatars could be exploited to produce convincing deepfakes, potentially undermining trust in visual media and influencing societies and politics. We strongly condemn any form of abuse or malicious use of our research and advocate for responsible development and application of face avatar technology, always in strict accordance with local laws and regulations.

References

[1] Yufan Chen, Lizhen Wang, Qijing Li, Hongjiang Xiao, Shengping Zhang, Hongxun Yao, and Yebin Liu. Monogaussianavatar: Monocular gaussian point-based head avatar. In *ACM SIGGRAPH*, pages 1–9, 2024. 2

[2] Xuangeng Chu and Tatsuya Harada. Generalizable and animatable gaussian head avatar. *Advances in Neural Information Processing Systems*, 37:57642–57670, 2024. 2

[3] Stavros Diolatzis, Tobias Zirr, Alexander Kuznetsov, Georgios Kopanas, and Anton Kaplanyan. N-dimensional gaussians for fitting of high dimensional functions. In *ACM SIGGRAPH*, pages 1–11, 2024. 1, 3, 4

[4] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv preprint arXiv:2405.10314*, 2024. 2

[5] Yisheng He, Xiaodong Gu, Xiaodan Ye, Chao Xu, Zhengyi Zhao, Yuan Dong, Weihao Yuan, Zilong Dong, and Liefeng Bo. Lam: Large avatar model for one-shot animatable gaussian head. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–13, 2025. 2

[6] Rawal Khirodkar, Timur Bagautdinov, Julieta Martinez, Su Zhaoen, Austin James, Peter Selednik, Stuart Anderson, and Shunsuke Saito. Sapiens: Foundation for human vision models. In *European Conference on Computer Vision*, pages 206–228. Springer, 2024. 2

[7] Tobias Kirschstein, Shenhan Qian, Simon Giebenhain, Tim Walter, and Matthias Nießner. Nersemble: Multi-view radiance field reconstruction of human heads. *ACM Trans. Graph.*, 42(4), 2023. 3, 7

[8] Tobias Kirschstein, Javier Romero, Artem Sevastopolsky, Matthias Nießner, and Shunsuke Saito. Avat3r: Large animatable gaussian reconstruction model for high-fidelity 3d head avatars, 2025. 2

[9] Jaeseong Lee, Taewoong Kang, Marcel Buehler, Min-Jung Kim, Sungwon Hwang, Junha Hyung, Hyojin Jang, and Jaegul Choo. Surfhead: Affine rig blending for geometrically accurate 2d gaussian surfel head avatars. In *The Thirteenth International Conference on Learning Representations*, 2025. 2

[10] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017. 2

[11] Hao Liang, Zhixuan Ge, Ashish Tiwari, Soumendu Majee, GM Godaliyadda, Ashok Veeraraghavan, and Guha Balakrishnan. Fastavatar: Instant 3d gaussian splatting for faces from single unconstrained poses. *arXiv preprint arXiv:2508.18389*, 2025. 2

[12] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 2

[13] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 2

[14] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In *Conference on Computer Vision and Pattern Recognition*, pages 20299–20309, 2024. 2

[15] Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. SplattingAvatar: Realistic Real-Time Human Avatars with Mesh-Embedded Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[16] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 2

[17] Jiapeng Tang, Davide Davoli, Tobias Kirschstein, Liam Schoneveld, and Matthias Niessner. Gaf: Gaussian avatar reconstruction from monocular videos via multi-view diffusion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5546–5558, 2025. 2

[18] Felix Taubner, Ruihang Zhang, Mathieu Tuli, and David B Lindell. Cap4d: Creating animatable 4d portrait avatars with morphable multi-view diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5318–5330. IEEE Computer Society, 2025. 2

[19] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024. 2

[20] Jun Xiang, Xuan Gao, Yudong Guo, and Juyong Zhang. Flashavatar: High-fidelity head avatar with efficient gaussian embedding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 4, 9

[21] Yuelang Xu, Benwang Chen, Zhe Li, Hongwen Zhang, Lizhen Wang, Zerong Zheng, and Yebin Liu. Gaussian head avatar: Ultra high-fidelity head avatar via dynamic gaussians. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1931–1941, 2024. 2, 3, 4

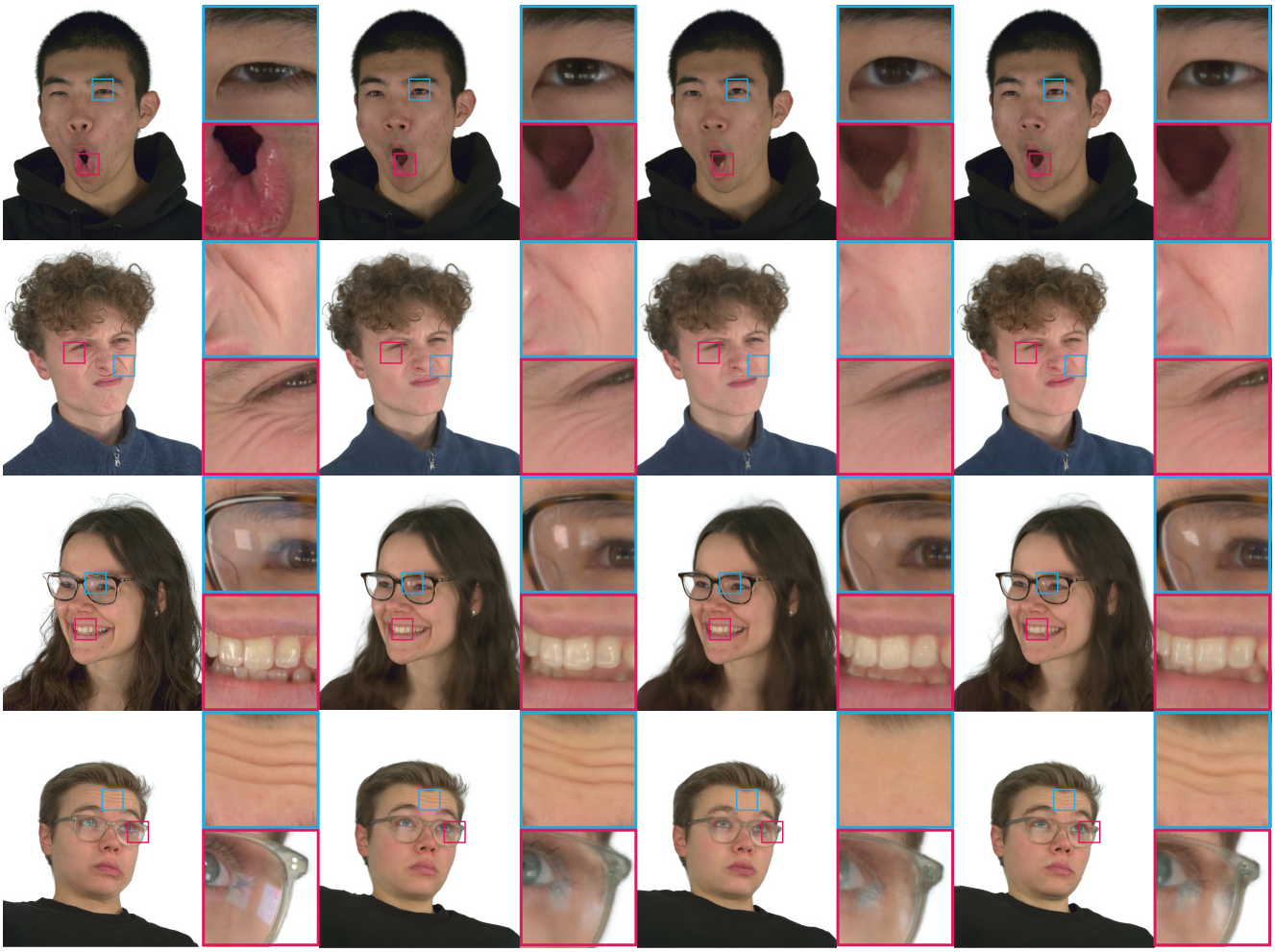

Figure 4. **Qualitative Comparison** with GHA and NDGS on subjects from the NeRSemble [7] dataset. While NDGS helps improve skin deformations, the resulting wrinkles are less pronounced (rows 2 and 4 show blurrier, washed-out results). Moreover, NDGS lacks visual fidelity for specular highlights (row 3) and struggles with accurate geometric alignment (teeth artifacts in row 1 and downward displacement of the glass frames in row 4).

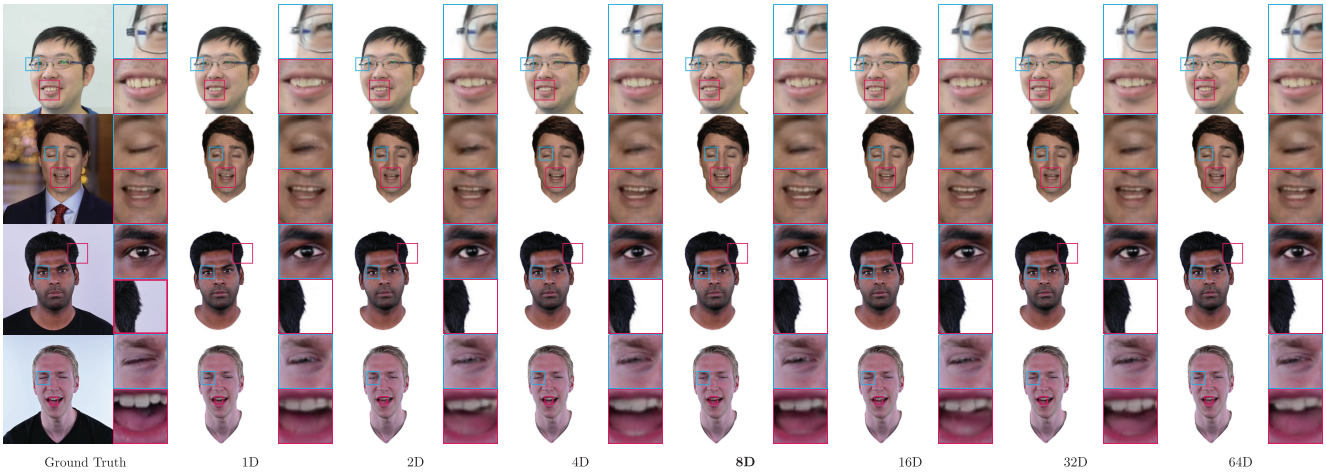

Figure 5. **Qualitative comparison** for varying latent dimensionalities. We find that HyperGaussians are robust towards different latent dimensions. A latent dimension of 8 performs best, but we already observe an improvement for a single latent dimension ($n = 1$) over the vanilla 3DGS variant.

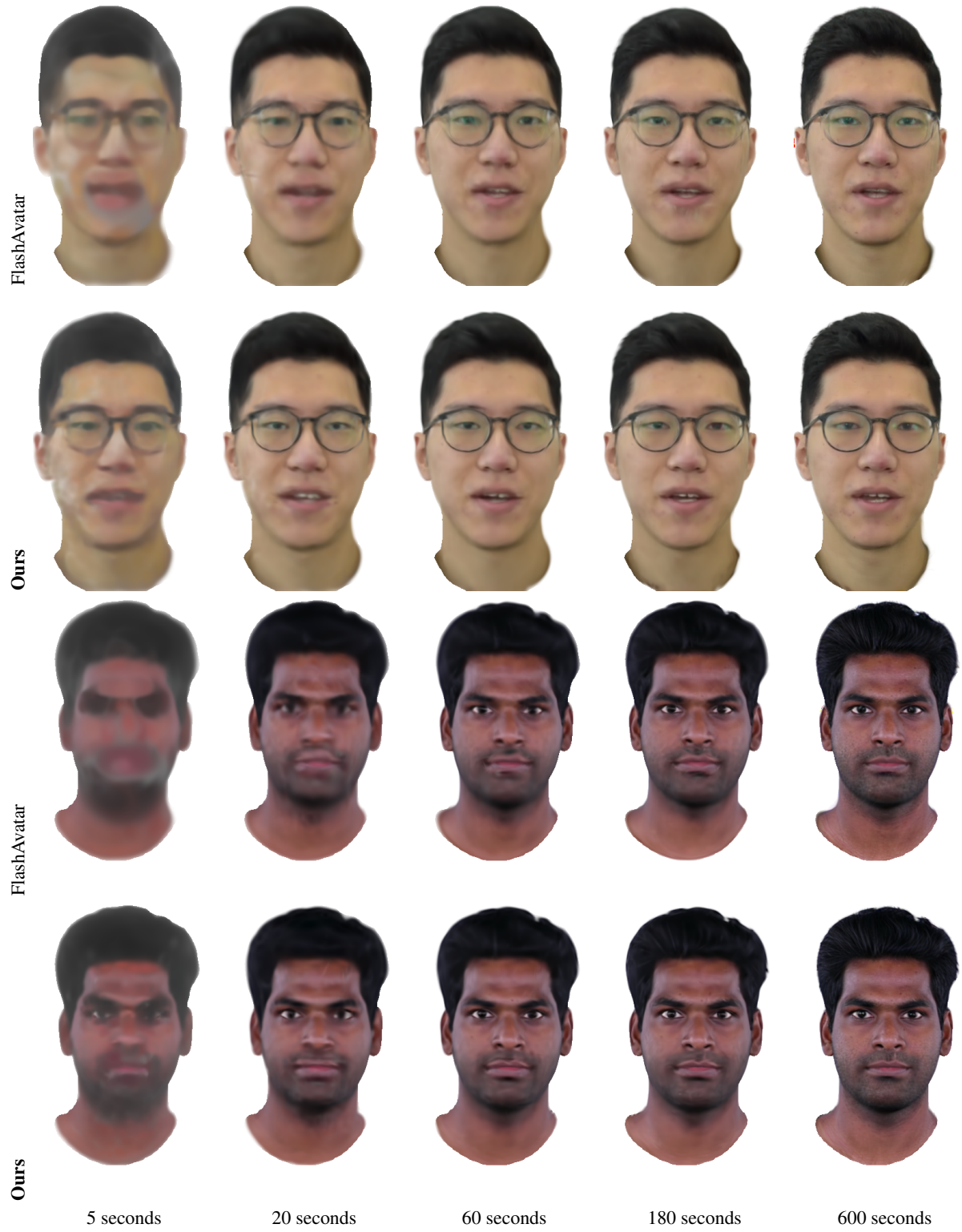

Figure 6. We compare the convergence speed of FlashAvatar [20] vs. **Ours**. The *only difference* between FlashAvatar and **Ours** is the substitution of 3D Gaussians (top) with HyperGaussians (bottom), as described in the case study in the main paper. From the beginning, HyperGaussians display sharper results.