GAUSSIAN3DIFF: 3D Gaussian Diffusion for 3D Full Head Synthesis and Editing

Supplementary Material

In this supplementary material, we provide additional details regarding the implementations and additional results. We also discuss the limitations of our model, and provide a supplementary HTML page showing animated results.

1. Implementation Details

1.1. Auto-decoder Loss Functions

Reconstruction Loss. We first briefly introduce the supervision we adopt in image reconstructions in both training stages 1.3. Note that all those supervisions are imposed on both the raw-3D low-res predictions and super-resolution module high-res outputs. First, we utilize the pixel-wise $\mathcal{L}_1$ loss,

$$\mathcal{L}_1(\mathbf{I}) = ||\mathbf{I} - \hat{\mathbf{I}}||_1 \ . \tag{1}$$

In addition, to learn perceptual similarities, we use the LPIPS [26] loss, which has been shown to better preserve image quality compared to the more standard perceptual loss:

$$\mathcal{L}_{\text{LPIPS}}(\mathbf{I}) = ||F(\mathbf{I}) - F(\hat{\mathbf{I}})||_2 \ , \tag{2}$$

where $F(\cdot)$ denotes the perceptual feature extractor.

Moreover, we render the multi-view depths from Panohead [2] samples for supervision, as in DepthNeRF [5]:

$$\mathcal{L}_{\text{depth}}(\mathbf{I}) = ||D(\mathbf{I}) - D(\hat{\mathbf{I}})||_2 \ , \tag{3}$$

where $D(\cdot)$ denotes the depth corresponding to the input image, which are computed in volumetric rendering.

Finally, a common challenge when handling the specific task of encoding facial images is the preservation of the input identity. To tackle this, we incorporate a dedicated recognition loss measuring the cosine similarity between the output image and its source,

$$\mathcal{L}_{\text{Id}}(\mathbf{I}) = 1 - \left\langle R(\mathbf{I}), R(\hat{\mathbf{I}}) \right\rangle \ , \tag{4}$$

where R is the pretrained ArcFace [4] network.

In summary, the overall reconstruction loss is the weighted summation of the loss functions described above:

$$\mathcal{L}_{\text{rgb}} = \mathcal{L}_1 + \mathcal{L}_{\text{LPIPS}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{Id}}\mathcal{L}_{\text{Id}} \ . \tag{5}$$

where we set $\lambda_{\text{depth}} = 0.1$ and $\lambda_{\text{Id}} = 0.1$. In the first training stage, the auto-decoder is optimized without the SR module, we set $\lambda_{\text{Id}} = 0$.

Gaussian Regularization Loss. Inspired by existing studies on 3D local representations [7, 13, 27], we impose the following regularization terms to stabilize the Gaussian training and better mirror the structure of the scene. We first impose $\mathcal{L}_{\text{cov}}$, the coverage loss which penalizes the L_1 norm of the Gaussian influence values (RBF) at the sample locations and encourages sparsity where possible:

$$\mathcal{L}_{\text{cov}}(\mathcal{G}, \boldsymbol{x}) = \lambda_{\text{cov}} \frac{1}{K} \sum_{k}^{K} g_i(\mathbf{x}) \ , \tag{6}$$

where g is the RBF value at query point $\mathbf{x}$ influenced by Gaussian $\mathcal{G}$ and $K = 3$ specifies the number of nearest Gaussians that contribute to $\mathbf{x}$.

To prevent the optimized ellipsoid from growing too large, we penalize the ellipsoid volume size by:

$$\mathcal{L}_{\text{vol}}(\mathcal{G}) = \lambda_{\text{vol}} \frac{4\pi}{3} \prod_{j}^{3} (\Sigma^{-1})_{j,j} \ , \tag{7}$$

where $(\Sigma^{-1})_{j,j}$ parameterizes the 3 axis-aligned radiis, which are the reciprocal of the variance along each principal axis.

Besides, we calculate the silhouette of a given image to encourage a cleaner background:

$$\mathcal{L}_{\text{silhouette}}(\mathbf{I}) = \lambda_{\text{silhouette}} ||M(\mathbf{I}) - M(\hat{\mathbf{I}})||_2 \ , \tag{8}$$

where $M(\cdot)$ denotes the silhouette mask of the given image.

We also impose a Total-Variance smoothness term over the Gaussians to encourage nearby Gaussians on the UV should change smoothly:

$$\mathcal{L}_{\text{TV}}(\mathcal{G}) = \lambda_{\text{TV}} \frac{1}{N} \sum_{h,w}^{N} |\mathcal{G}_{h+1,w} - \mathcal{G}_{h,w}| + |\mathcal{G}_{h,w+1} - \mathcal{G}_{h,w}| \ , \tag{9}$$

where N is the UV map size and h, w is the spatial UV map index.

Furthermore, we ensure that the optimized Gaussian centers remain aligned with the initial 3DMM registration by employing a regularization term that penalizes deviations from the initialized 3DMM mesh vertices $\mathbf{v}$, as expressed in the following loss function:

$$\mathcal{L}_{\text{mesh}}(\mathcal{G}) = \lambda_{\text{mesh}} \frac{1}{N} \sum_{h,w}^{N} \left|\left| \mathbf{v}_{h,w} - \boldsymbol{\mu}_{h,w} \right|\right|_2^2 \ ,$$

as in the approach of [12].

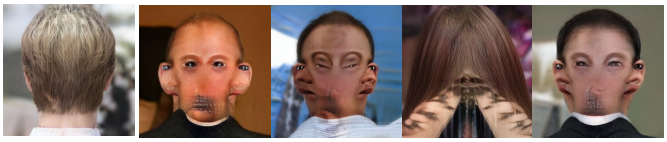

Figure 1. **Visualization of positive and negative samples from Panohead.** The display includes one positive sample and four negative samples here. The positive sample process a "human front face" score of $2.16e-04$, while all the negative samples process scores ≥ 0.5.

The final Gaussian regularization loss reads as the weighted summation of all the terms listed above:

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{cov}} + \mathcal{L}_{\text{vol}} + \mathcal{L}_{\text{silhouette}} + \mathcal{L}_{\text{TV}} + \mathcal{L}_{\text{mesh}} \ , \quad (10)$$

where we set $\lambda_{\text{cov}} = 0.001$, $\lambda_{\text{tanh}} = 0.1$, $\lambda_{\text{silhouette}} = 1$, $\lambda_{\text{vol}} = 1$, $\lambda_{\text{TV}} = 0.1$, and $\lambda_{\text{mesh}} = 0.01$ as the defined loss weights.

Code Prior Loss. For optimizing auto-decoder D, following DeepSDF [14] we have

$$\mathcal{L}_{\text{code}} = \lambda_{\text{code}} \frac{1}{\sigma^2} \|\mathbf{z}\|_2^2 \ , \quad (11)$$

where in the latent shape-code space, we assume the prior over codes $p(\mathbf{z}_i)$ to be a zero-mean multivariate-Gaussian with a spherical covariance $\sigma^2 I$. We initialize $\mathbf{z}$ from $\mathcal{N}(0, 0.01^2)$ and set $\lambda_{\text{code}} = 1e-4$ following DeepSDF.

1.2. Data Preparation

Panohead Data Generation. Due to its ability to generate high-quality full-head 3D portraits, we adopt Panohead [2] as our 3D data source [28]. Though the tri-grid design greatly alleviates the Jenus problem [15] compared with tri-plane [3], in practice we still find around 30% among Panohead synthesized samples are likely to fail, as shown in Fig. 1. We observed that these samples even cannot be reconstructed well and greatly poison the auto-decoder's performance. Therefore, we filter those samples with CLIP [16], where we calculate the CLIP score of the input back head image with two text prompts "human front face" and "back of human head" and calculate the normalized softmax probability. We leave all samples with the "human front face" score < 0.007 for training.

Registration. Due to the scale differences between 3D Morphable Models (3DMMs) and 3D human heads synthesized from Panohead [2], direct fitting of a 3DMM to multi-view images with pre-defined camera poses used in data generation is not feasible [11]. To overcome this challenge, we employed a 3DMM fitting algorithm to simultaneously estimate 3DMM parameters and camera poses from multi-view images. Subsequently, a homogeneous transformation with scaling is computed to align the estimated camera poses with the pre-defined camera poses. This transformation is then applied to the fitted 3DMM mesh to transform it into the Panohead coordinates.

1.3. Training Details

Auto Decoder. We adopted AdamW optimizer with no weight decay for all the experiments, and set $lr = 0.05$ for optimizing $\mathbf{z}$, $lr = 0.0025$ for optimizing autodecoder D and $lr = 1e-5$ for optimizing Gaussians and $lr = 5e-2$ for optimizing a tiny 2-layer rendering MLP with $8 \times 32 \times 4$ shape. We decay all the learning rates at 100k steps by a factor of 0.5.

We first train the autodecoder using 64×64 raw renderings from Panohead with batch size 30 for 250K iterations and achieve PSNR around 28db. For scalability, for each data point, we randomly crop a 36×36 patch for supervision. We augment the background color to white to encourage better foreground-background disentanglement. In the next stage, we cascade a pre-trained ESRNet [22] after the autodecoder to upsample the 64×64 renderings, resulting in high-resolution synthesis at 256×256. $\mathcal{L}_{\text{Id}}$ is only imposed during the second stage. We set the learning rate for ESRNet with $1e-4$ and jointly train both models with batch size 4 for another 250k iterations. The overall training takes around 6 days on two RTX A6000 GPUs. The stored autodecoder checkpoint occupies a space of 66.96 MiB, while the ESRNet consumes 63.94 MiB.

Diffusion Models. We adhere to the diffusion model implementation as outlined in Imagen [18], and the specific details are enumerated in Tab. 1. Note that we normalize the Gaussians and payload features to $[-1, 1]$ for better diffusion learning. Besides, we set the expression of the reconstructed 3D Gaussians to the neutral state. This ensures that the generated outcomes can be directly animated by adding expression offsets, as per the methodology proposed in FLAME [10]. We train the diffusion model using Jax on 16 A100 GPUs for $500K$ iterations, which takes around 5 days.

2. More Analysis

2.1. More Ablation Studies

Selection of K Nearby Gaussians for Blending. We investigate the choice of K value in Gaussian blending and include the quantitative results in Tab. 2. Specifically, we reconstruct 24 instances using our method for $20k$ iterations using identical hyper-parameters except the K and employing $\mathcal{L}_{\text{L1}}$ loss. Note that we only compare the 64x64 outputs from the autodecoder, as the number of K only affects the volumetric rendering. As it can be observed, the reconstruction performance initially improves steadily with in-

Diffusion Model Details	
Learning Rate	$2e-5$
Batch Size	96
Optimizer	ADAM
$\mathcal{G}_0$ shape	(9+3×8)×256×256
U-Net base channels	256
U-Net channel multiplier	1, 1, 2, 2, 4, 4
U-Net res block	2
U-Net attention resolutions	32, 16, 8
U-Net attention head	4
U-Net attention head channels	64
U-Net norm layer type	GroupNorm
Diffusion steps	1000
Noise schedule	Cosine
Clip Input $[-1, 1]$	True
normalizing Gaussian centers	(x + 0.12) ∗ 2
normalizing Gaussian rotations	x / math.pi
normalizing Gaussian radii	(x.abs(). clip(0, 0.15) − 0.06) ∗ 10
normalizing Gaussian payload	tanh(x)

Table 1. Hyperparameters and architecture of diffusion model f_θ. Additionally, it is worth noting that we also normalize $\mathcal{G}_0$ to range $[-1, 1]$ to enhance diffusion training, employing the normalization function detailed above.

Table 2. **Quantitative Ablation of K in Gaussian Blending.** We evaluate the reconstruction results at $20K$ iterations. Increasing the value of K improves the quality of reconstruction, albeit with an associated increase in computational resources. Nevertheless, beyond K=3, a decline in performance is observed, indicative of potential undertraining of the model. BS denotes batch size.

K value	PSNR ↑	MAE ↓	VRAM(MiB)@BS=4 ↓	Latency(s)/step ↓
1	27.72 ± 2.53	0.0421 ± 0.013	6886	0.4
2	27.92 ± 2.40	0.0412 ± 0.012	11508	0.53
3	**28.39** ± 2.55	**0.0392** ± 0.012	16124	0.67
4	28.28 ± 2.37	0.0400 ± 0.012	20774	0.8
5	28.25 ± 2.33	0.0395 ± 0.011	25402	0.87

creasing K, as does the computational cost. This is because evaluating the RBF value of more Gaussians requires storing extra gradients and more computation. Surprisingly, we did not observe quantitative performance gain after $K = 3$. We further qualitatively compare the reconstruction differences in Fig. 2. Upon visual inspection, it is evident that smaller K values produce noisier reconstructions in both color and depth, while increasing K values gradually produce smoother and more complete results. To trade off the overall performance, we adopt $K = 3$ in our experiments. However, efficient training with larger K is still worth exploring in future work.

Auto-Decoder Design. Though existing 3D diffusion methods like Rodin [21] also adopt an auto-decoding design, it is noteworthy that only a shallow rendering MLP is reused across instances. This design requires the learning of a shared decoder across hundreds of instances, followed by fitting each 3D object with the shared frozen decoder. In our ablation study, we specifically investigate the use of a shared convolutional decoder D in Fig. 3. Specifically, we conduct texture transfer and expression editing experiments on overfitted samples where no shared auto-decoder is used. As illustrated, the interpolation and expression editing yield semantically inconsistent results when compared with Fig. 8, such as the absence of teeth. Moreover, adopting a convolutional decoder for fitting enhances scalability, allowing for optimizing $10,000$ instances using 2 GPUs.

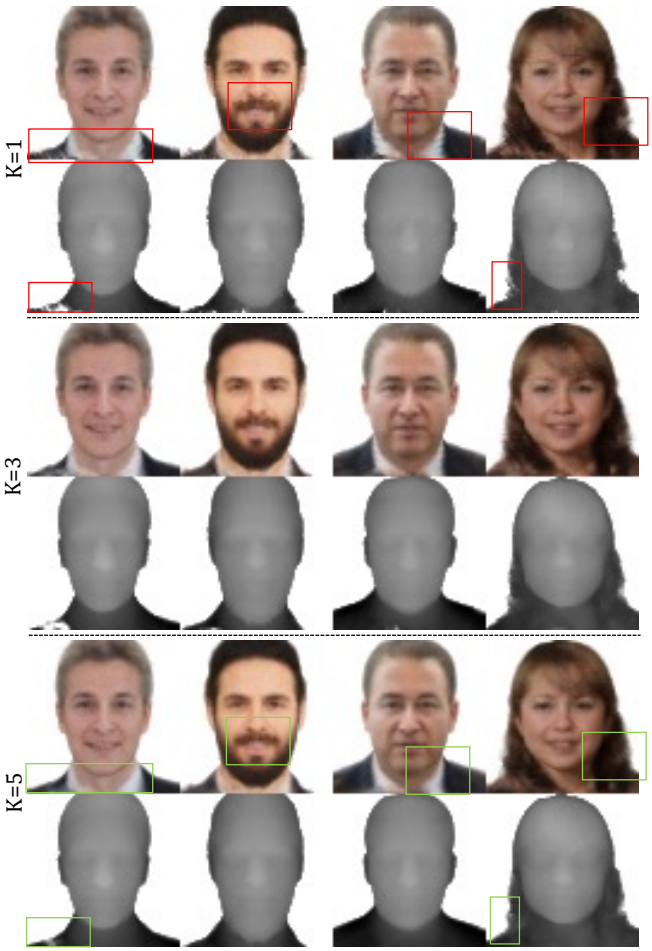

Figure 2. **Qualitative Ablation of K in Gaussian Blending.** Despite achieving comparable quantitative performance, a smaller K results in visually noisier reconstruction, as shown in the red box. Setting $K = 5$ enhances smoothness, as illustrated in the green box. We opt for $K = 3$ to strike a balance in overall performance. All results are obtained directly from the auto-decoder D, without employing super resolution.

2.2. Limitations and Failure Cases

Single-image Inversion. Though we use multi-view images for reconstruction, here we include single-view inversion results in Fig. 4. Specifically, we first register the

Figure 3. **Ablation Study on autodecoder.** We first reconstruct two instances individually without a shared convolutional decoder and show the texture transfer in row-1, and animation results in row-2. Visually inspected, the baseline design lacks the ability to learn unseen missing features from other instances, *e.g.*, teeth and diverse expression. Our design shows better generality and scalability.

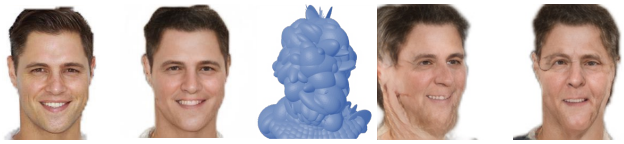

Figure 4. **Single-view inversion on the pre-trained auto-decoder** D We randomly sample one image from CelebA-HQ [9] and conduct test-time optimization to find the corresponding Gaussians. while the input view can be plausibly reconstructed, the results for novel views and expressions exhibit noticeable artifacts.

3DMM parameters following the procedures introduced in Sec. 1.2. Next, we adopt the SG2 [1] inversion paradigm, which optimizes the latent code $\mathbf{z}$ and Gaussians $\mathcal{G}$ via the reconstruction loss. As can be seen, though the input reconstruction is reasonable, the optimized Gaussians are noisy and the novel view synthesis and novel expressions are unsatisfactory. Therefore, how to conduct 3D Gaussian reconstruction from few-shot images is still open for future research. Existing efforts in 3D-GAN inversion may also provide possible solutions, *e.g.*, training a 3D-aware encoder [8] or introducing pseudo depth loss [25].

Dataset Bias. A well-recognized challenge in deep generative models is the presence of bias, whereby the model exhibits a propensity to generate one gender more frequently than the other, or certain attributes become closely intertwined. Given that our training data originates from Panohead, our model inherently inherits the bias issue present in the data. Specifically, we have observed a tendency for the generated females to exhibit a "smile" more frequently when compared to the generated male instances. This bias becomes apparent when examining Fig. 6 columns 4, 6, and 7, where the facial expressions of the source identities are predominantly neutral. Nevertheless, the female reenactment results still depict a subtle smiling expression, with visible teeth.

Jointly Training on Synthetic and Real-world Dataset. Real-world high-quality avatar dataset [24] suffers from quantity and diversity, while the synthetic 3D-GAN synthetic samples suffer from resolution and expressions, where the raw 3D-renderings are usually limited to 64×64 with one expression available. An open direction is to study how to marry the best of both data sources to establish a high-fidelity and diverse avatar system.

Avatar-specific Design. Presently, our training framework does not incorporate avatar-specific design elements, such as a mouth in-inpainting module in Next-3D [20], landmarks supervision and face region-specific GAN loss as utilized GFPGAN [23]. Moreover, each training instance is constrained to a single expression. Expanding our method to datasets and training designs tailored for avatar-specific characteristics hold the potential for further performance improvement.

Application on the General Scenes. Currently, we only showcase the 3D Gaussians-based generation ability within the realm of 3D portraits. However, given the widespread use of UV map in the industry, expanding GAUSSIAN3DIFF as a general 3D generative model is worth future investigation.

3. More Application Results

Shape-Texture Transfer. GAUSSIAN3DIFF naturally supports geometry-texture disentanglement, where Gaussians managing the geometry and attached local tri-planes determining the texture within a local region defined on the UV map. As illustrated in Fig. 8, we conduct both shape and texture transfer by swapping the corresponding areas on the UV map from the source identity to the target. Visually inspected, the texture-transferred identity inherits the texture characteristics from the source while maintaining the shape layout of the target, and the shape-transferred results maintain the layout of the source without affecting their textures.

3D Inpainting using Diffusion Model. Our trained diffusion model supports 3D in-painting given geometry and texture mask individually. We show both the geometry- and texture-based inpainting in Fig. 5. Both yield holistically reasonable results while keeping the corresponding inputs within the mask unchanged. Note that there are still noticeable color changes between the inpainting boundaries, which could be addressed by introducing inpainting-specific training design [17].

3DMM-based Editing. Thanks to the delicate design, GAUSSIAN3DIFF marries the best of both the model-based 3DMM and neural representations through the rasterized UV space, and naturally supports 3DMM-based editing, *e.g.*, avatar modeling by changing the shape and expression codes. We show this ability by reenacting the shape and

Figure 5. **Diffusion-based Inpainting given Geometry or Texture Mask.** We provide either the geometry part (first 9 channels of the $256 \times 256 \times 33$ tensor $\mathcal{G}_0$) or the texture part (last 24 channels of the $256 \times 256 \times 33$ tensor $\mathcal{G}_0$) of the *upper face* as hints, and take the diffusion model f_θ in-paints the remaining details. As illustrated in columns $2-5$, all the generated results exhibit an identical upper-face shape, including the layout of the eyes and forehead, but with different textures. Conversely, the diffusion-inpainted results with texture masks (columns $6-10$) showcase an identical upper-face texture, encompassing features such as hair and forehead color, while varying in shape.

Figure 6. **3DMM-based Editing**. We reenact 8 random sampled source inputs (row-1) onto the targets (row-$2, 3, 4$) by driving the target Gaussians through the adjustment of the inner 3DMM mesh using the shape and expression codes from the source. The reenactment results preserve their original texture while adapting to the shape and expression of the source inputs.

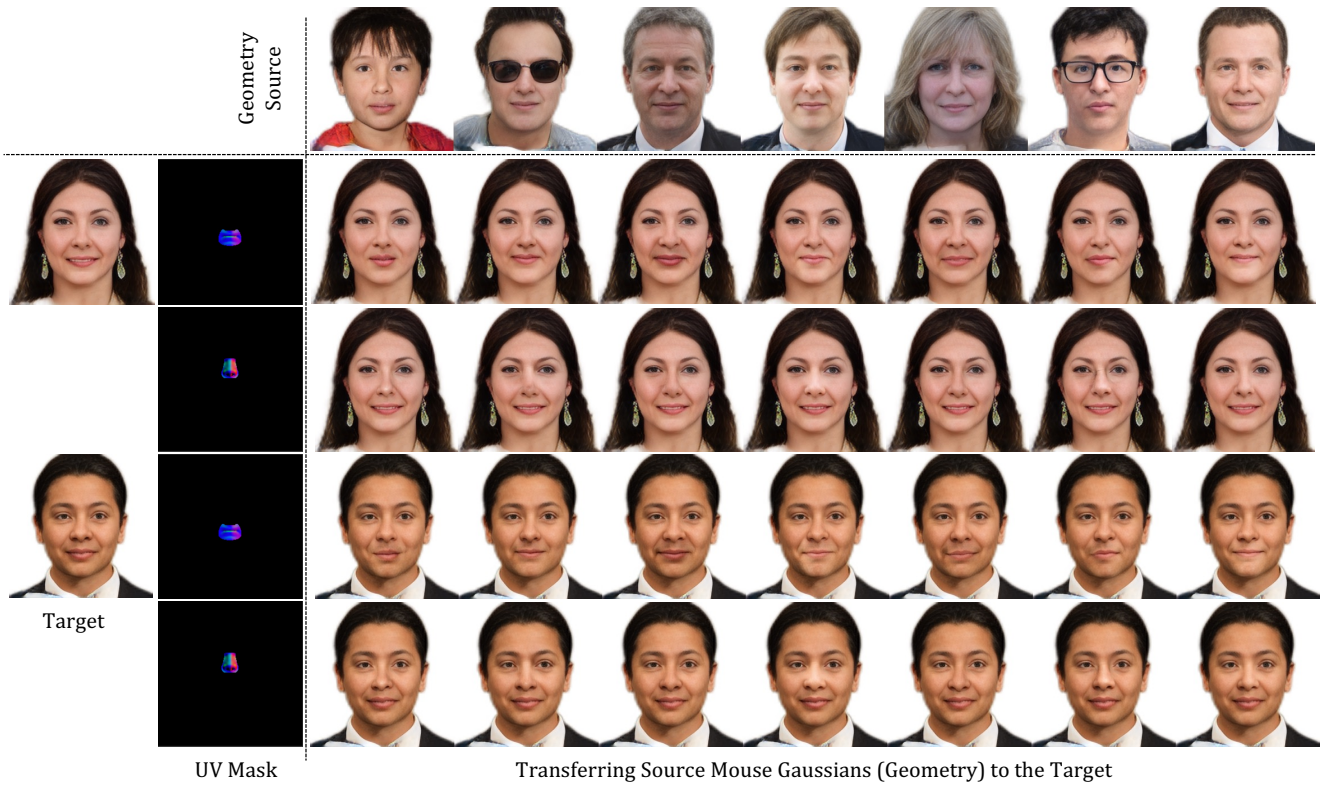

Figure 7. **Regional Editing with UV Mask**. The UV-space design in GAUSSIAN3DIFF enables the editing of specific semantic regions defined on the UV map. In this example, we transfer the geometry shapes of the "nose" and "mouth" from the source identities to the target while leaving the remaining areas unchanged. It is important to note that all the source and target identities presented here are diffusion-sampled.

expression codes of given source identities onto the target reconstructed instances. Specifically, we drive the learned Gaussians by adding the offsets between the positions of source identity vertices and initialized target identity vertices. We reuse the identities included in Fig.4 in the main paper for simplicity. As shown in Fig. 6, the reenactment results maintain their original texture details but accurately follow the shape and expression of the given source input. This application has the potential to facilitate avatar editing in game engines and media creation.

Regional-based Editing. In addition to global interpolation and transfer capabilities, GAUSSIAN3DIFF provides support for region-based editing, allowing modification exclusively within the semantic region defined by the UV mask. This functionality is illustrated in Fig. 7, where we showcase the transfer of corresponding source Gaussians (geometry) to the target, guided by the provided mask. The transferred results exhibit the same shape as the source within the defined semantic region, while the remaining areas remain unchanged. Benefiting from the thoughtful design of the UV space, regional editing consistently produces semantically consistent results when transferring between mouth/nose regions of varying sizes across different identities. Furthermore, this demonstration underscores that GAUSSIAN3DIFF can surpass 3DMM constraints, enhancing controllability and exhibiting significant potential for avatar personalization within game engines [6].

4. More comparisons with Baselines

We include the qualitative reenactment comparison of IDE-3D [19] to our method in Fig. 9. Both methods can accurately reflect the input expression in their reeactment results, although IDE-3D exhibits a deficiency to preserve the input identity, attributed to its segmentation-based animation paradigm. This difference accounts for their inferior PCK performance compared to ours in the main paper.

Moreover, IDE-3D and FE-NeRF lack support for region-based transfer between identities. This limitation arises from the fact that global-based representation such as tri-plane and MLP define all the 3D components in the 3D space. Attempting a direct copy-paste of one tri-plane region to another results in inconsistencies, especially when there are variations in their absolute sizes.

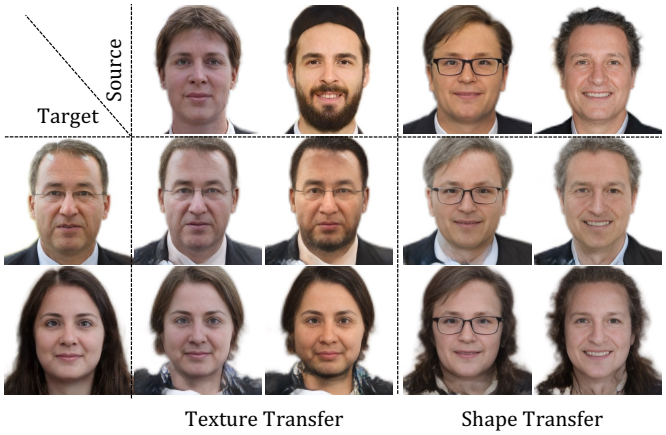

Figure 8. **Shape Texture Transfer.** Given two randomly sampled source identities, each serving as the source of texture (columns $1-2$) and shape (columns $3-4$), we transfer their corresponding information defined in the UV space to four targets. Importantly, the transferred textures semantically adhere to the target shape; for instance, the beard in column 2 varies according to the size of the Gaussians in that region. Additionally, the texture source identity of column 1 does not display teeth, while the transferred results do, illustrating that the autodecoder D learns a shared meaningful latent space. These results, as evidenced in Fig. 3, cannot be achieved via per-instance fitting.

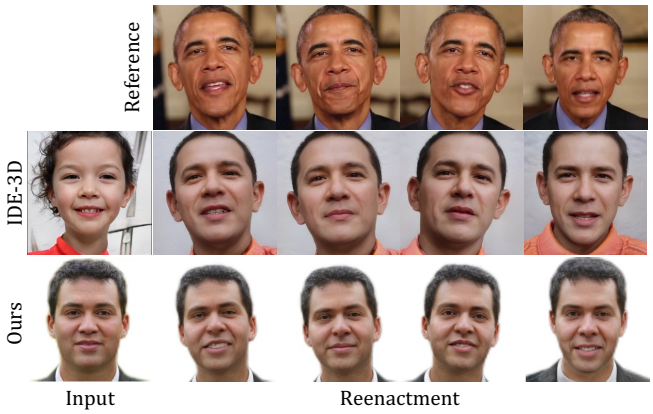

Figure 9. **Comparison with IDE-3D on Reenactment.** Employing four reference images (row-1), we animate the identities synthesized by IDE-3D and our method to the target expression. Given IDE-3D's segmentation-based nature, utilizing a reference identity with a different shape inevitably alters the facial layout. Nevertheless, GAUSSIAN3DIFF is build on a identity-and-expression disentangled modeling, facilitated by the underlying 3DMM, ensuring the preservation of the input identity through the reenactment process.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 4

[2] Sizhe An, Hongyi Xu, Yichun Shi, Guoxian Song, Umit Y. Ogras, and Linjie Luo. Panohead: Geometry-aware 3d full-head synthesis in 360deg. In *CVPR*, pages 20950–20959, 2023. 1, 2

[3] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, 2022. 2

[4] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. In *CVPR*, 2019. 1

[5] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, 2022. 1

[6] Unreal Engine. MetaHuman - Realistic Person Creator. 6

[7] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3D shape. In *CVPR*, 2020. 1

[8] Yushi Lan, Xuyi Meng, Shuai Yang, Chen Change Loy, and Bo Dai. E3dge: Self-supervised geometry-aware encoder for style-based 3d gan inversion. In *CVPR*, 2023. 4

[9] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. MaskGAN: Towards diverse and interactive facial image manipulation. In *CVPR*, 2020. 4

[10] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *TOG*, 36(6), 2017. 2

[11] 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. 2

[12] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Trans. Graph.*, 37(4), 2018. 1

[13] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Trans. Graph.*, 40(4), 2021. 1

[14] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019. 2

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

[16] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2

[17] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 4

[18] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *NIPS*, 35:36479–36494, 2022. 2

[19] Jingxiang Sun, Xuan Wang, Yichun Shi, Lizhen Wang, Jue Wang, and Yebin Liu. Ide-3d: Interactive disentangled editing for high-resolution 3d-aware portrait synthesis. *ACM Transactions on Graphics (TOG)*, 41(6):1–10, 2022. 6

[20] Jingxiang Sun, Xuan Wang, Lizhen Wang, Xiaoyu Li, Yong Zhang, Hongwen Zhang, and Yebin Liu. Next3d: Generative neural texture rasterization for 3d-aware head avatars. In *CVPR*, 2023. 4

[21] Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *CVPR*, pages 4563–4573, 2023. 3

[22] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *ECCVW*, 2018. 2

[23] Xintao Wang, Yu Li, Honglun Zhang, and Ying Shan. Towards real-world blind face restoration with generative facial prior. In *CVPR*, 2021. 4

[24] Cheng-hsin Wuu, Ningyuan Zheng, Scott Ardisson, Rohan Bali, Danielle Belko, Eric Brockmeyer, Lucas Evans, Timothy Godisart, Hyowon Ha, Xuhua Huang, 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. 4

[25] Jiaxin Xie, Hao Ouyang, Jingtan Piao, Chenyang Lei, and Qifeng Chen. High-fidelity 3d gan inversion by pseudo-multi-view optimization. In *CVPR*, 2023. 4

[26] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 1

[27] Xiaoshuai Zhang, Abhijit Kundu, Thomas Funkhouser, Leonidas Guibas, Hao Su, and Kyle Genova. Nerflets: Local radiance fields for efficient structure-aware 3d scene representation from 2d supervision. In *CVPR*, 2023. 1

[28] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. DatasetGAN: Efficient labeled data factory with minimal human effort. In *CVPR*, 2021. 2