# Instant Facial Gaussians Translator for Relightable and Interactable Facial Rendering

DAFEI QIN, The University of Hong Kong and Deemos Technology Co., Ltd., China
HONGYANG LIN, ShanghaiTech University and Deemos Technology Co., Ltd., China
QIXUAN ZHANG, ShanghaiTech University and Deemos Technology Co., Ltd., China
KAICHUN QIAO, ShanghaiTech University and Deemos Technology Co., Ltd., China
LONGWEN ZHANG, ShanghaiTech University and Deemos Technology Co., Ltd., China
ZIJUN ZHAO, ShanghaiTech University and Deemos Technology Co., Ltd., China
JUN SAITO, Adobe Research, USA
JINGYI YU, ShanghaiTech University, China
LAN XU, ShanghaiTech University, China
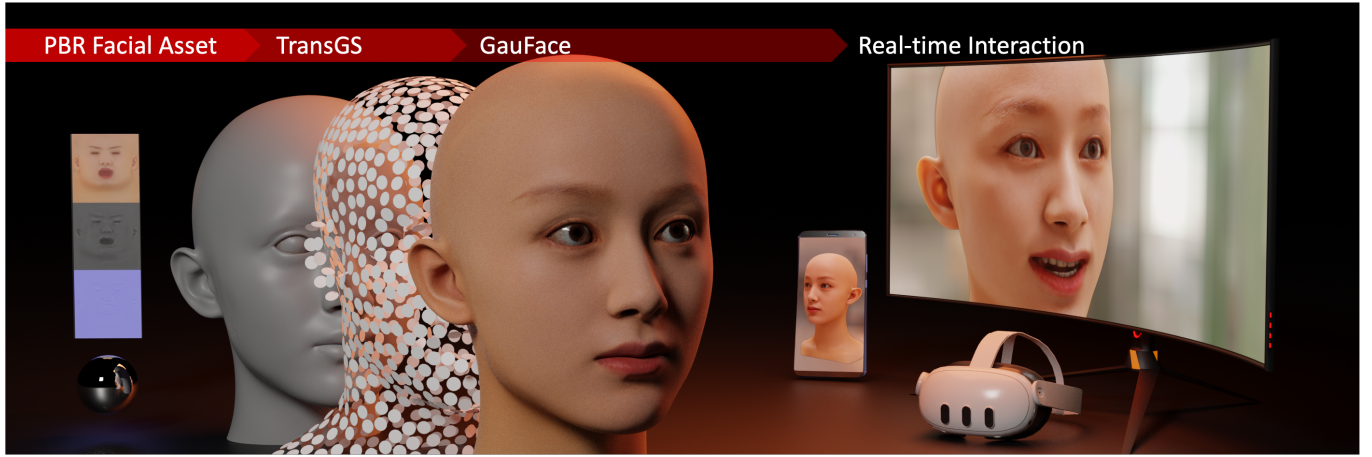TAKU KOMURA*, The University of Hong Kong, China

Fig. 1. Conditioning on a desired lighting, our proposed *TransGS* system transfers physically-based facial assets (left) to *GauFace* (middle), a novel Gaussian representation, in *seconds*. The resulting *GauFace* asset offers high-quality real-time rendering and animation across various platforms (right).

The advent of digital twins and mixed reality devices has increased the demand for high-quality and efficient 3D rendering, especially for facial avatars. Traditional and AI-driven modeling techniques enable high-fidelity 3D asset generation from scans, videos, or text prompts. However, editing and rendering these assets often involves a trade-off between offline quality and online speed. In this paper, we propose *GauFace*, a novel Gaussian Splatting representation, tailored for efficient animation and rendering of physically-based facial assets. Leveraging strong geometric priors and

Authors' addresses: Dafei Qin, qindafei@connect.hku.hk, The University of Hong Kong and Deemos Technology Co., Ltd., Shanghai, China; Hongyang Lin, ShanghaiTech University and Deemos Technology Co., Ltd., Shanghai, China, linhy@shanghaitech.edu.cn; Qixuan Zhang, ShanghaiTech University and Deemos Technology Co., Ltd., Shanghai, China, zhangqx1@shanghaitech.edu.cn; Kaichun Qiao, ShanghaiTech University and Deemos Technology Co., Ltd., Shanghai, China, qiaokch2022@shanghaitech.edu.cn; Longwen Zhang, ShanghaiTech University and Deemos Technology Co., Ltd., Shanghai, China, zhanglw2@shanghaitech.edu.cn; Zijun Zhao, ShanghaiTech University and Deemos Technology Co., Ltd., Shanghai, China, zhaozj2022@shanghaitech.edu.cn; Jun Saito, jsaito@adobe.com, Adobe Research, Seattle, USA; Jingyi Yu, ShanghaiTech University, Shanghai, China, yujingyi@shanghaitech.edu.cn; Lan Xu, ShanghaiTech University, Shanghai, China, xulan1@shanghaitech.edu.cn; Taku Komura, taku@cs.hku.hk, The University of Hong Kong, Hong Kong, China.

constrained optimization, *GauFace* ensures a neat and structured Gaussian representation suitable for efficient rendering and generative modeling. Then, we introduce *TransGS*, a diffusion transformer that instantly translates physically-based facial assets into the corresponding *GauFace* representations. Specifically, we adopt a patch-based pipeline to handle the vast number of Gaussians effectively. We also introduce a novel pixel-aligned sampling scheme with UV positional encoding to ensure the throughput and rendering quality of *GauFace* assets generated by our *TransGS*. Once trained, *TransGS* can instantly translate facial assets with lighting conditions to *GauFace* representation, delivering high fidelity and real-time facial interaction of 30fps@1440p on a *Snapdragon® 8 Gen 2* mobile platform. Notably, with the rich conditioning modalities, it also enables editing and animation capabilities reminiscent of traditional CG pipelines.

We conduct extensive evaluations and user studies, compared to traditional offline and online renderers, as well as recent neural rendering methods, which demonstrate the superior performance of our approach for facial asset rendering. We also showcase diverse immersive applications of facial assets using our *TransGS* approach and *GauFace* representation, across various platforms like PCs, phones and even VR headsets.

# 1 INTRODUCTION

The advent of digital twins and mixed reality (MR) devices is transforming expectations for 3D asset quality and rendering efficiency. Given their central role in human interaction, facial avatars, which convey emotions and intentions, must be rendered with precision. Our innate ability to perceive even the slightest inaccuracies in facial animations can lead to the uncanny valley effect, where almost but not entirely lifelike appearances cause viewer discomfort.

Production-level workflows for crafting lifelike facial avatars involve both modeling and rendering. The modeling phase aims to recover CG-friendly facial assets with nuanced facial idiosyncrasies. These include realistic geometry with structured UV unwrapping, physically-based appearance (e.g., diffuse albedo, specular intensity, roughness, normal map, and displacement), and the motion rig to empower expression manipulation and animation. The past decade has witnessed the rapid progress of such facial modeling. Early attempts [Alexander et al. 2010; Debevec et al. 2000] require immense artistic sculpting or expensive apparatus like Light Stage. Recent advances in AI-Generated-Content lower the burden of facial modeling. One can easily customize physically-based facial assets from more light-weight inputs, i.e., a single scan [Li et al. 2020b], a monocular video or image [Cao et al. 2022], or even text prompts [Zhang et al. 2023a]. Yet, the rendering phase of such facial assets has been left behind. Thus far, most rendering tools in existing CG software adopt the GPU-based rasterization pipeline, offline or online. Offline renderers like Arnold [Autodesk 2024] or Cycles [Blender Foundation 2024], are characterized by intricate ray tracing computations, frequently leading to prolonged rendering duration. Conversely, online ones like Unity3D Build-in Pipeline or OpenGL prioritize interactive responsiveness at the expense of rendering fidelity. Hence, this leads to the substantial distinction between facial avatars featured in offline cinematic productions and online gaming environments. Take the distinct characters of Keanu in the feature film "The Matrix" and the game "Cyberpunk 2077" as examples.

Recent neural advances [Mildenhall et al. 2021; Tewari et al. 2021] conduct volume rendering at photo-realism, bringing new potentials to narrow the gap between traditional offline and online rendering. Notably, 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] stands out for its exceptional rendering ability at speed and quality. Also, its explicit representation can be seamlessly integrated into the GPU-based rasterization pipeline. Various approaches extend 3DGS into dynamic scenes, i.e., human performance [Jiang et al. 2024; Wu et al. 2023] or facial animations [Chen et al. 2023; Qian et al. 2024; Saito et al. 2024]. Yet, they rely on per-scene training for modeling real-life scenes from video or image inputs, thus unable to directly apply to CG-friendly facial assets for instant improvement of rendering quality. One can apply offline render techniques, e.g. monte carlo path-tracing and subsurface-scattering on the facial assets to create high-quality images under diverse expressions and view angles and subsequently optimize the 3DGS for render acceleration. But such cumbersome data preparation and optimization make it impractical for interactive applications, let alone supporting lighting setup, online editing, or animation control as if using the original facial assets.

To tackle the above challenges, we propose *TransGS* , a novel translator to instantly translate CG-friendly facial assets into novel 3DGS-like representations. As shown in Fig. 1, our approach enables real-time and high-quality rendering of physically-based facial assets, even comparable with offline rendering techniques. It also can be seamlessly integrated into various platforms like PC, phone, or VR headset, and it is compatible with traditional facial assets for relighting, editing, and animation capabilities.

The key design in our approach is a Gaussian representation tailored for facial assets, dubbed *GauFace* . On the one hand, our *GauFace* bridges Gaussian splatting with the strong geometry and texture priors in the facial assets. On the other hand, careful optimization yields a neat and structured representation, paving the way for adopting an efficient generative paradigm in our translator. Specifically, in our *GauFace* , we attach and rig the Gaussian primitives onto the face mesh to naturally support the mesh-based animation of the original facial asset. To model nuanced appearance changes under animation, we adopt a novel shading vector to disentangle the deformation-dependent/agnostic shading effects. Then, for a facial asset with a desired lighting map, we tailor the optimization scheme from 3DGS to obtain the corresponding *GauFace* asset. We adopt a *Pixel Aligned Sampling* scheme to uniformly initialize Gaussians on the UV plane and defer the pruning operation during optimization until rendering. Such strategies not only constrain Gaussian attributes to have neat and tight distributions but also make their sampling positions known to provide rich conditioning priors for our generative translator.

Then, we develop our instant generative translator *TransGS* with a diffusion transformer (DiT) architecture to model the intricate one-to-many relationship between a 3D facial asset and its *GauFace* transcript. To train *TransGS* , we first collect 143 physically-based facial assets with 4K textures and geometry and render them under 134 HDR environment maps, resulting in a total of 1,023 combinations. We prepare 1,071 high-quality rendered images for each combination to optimize the corresponding *GauFace* asset. We then train *TransGS* on the optimized 1,023 *GauFace* assets. During training, we adopt a patch-based pipeline of DiT architecture to effectively handle the vast number of Gaussian points per *GauFace* asset. We also utilize UV positional encoding to guide the transformer's attention toward textures proximal to the Gaussian sampling positions. With rich conditioning inputs on the geometry, PBR textures of the facial asset, and the desired HDR environment lighting, we can swiftly generate the *GauFace* transcript of a facial asset in a matter of seconds. Thanks to such efficient generation and rich conditions, *TransGS* supports swiftly transferring new modifications of geometry and image textures from the original facial assets to the generated *GauFace* assets, thereby offering control and editing capabilities akin to traditional CG pipelines. We conduct extensive evaluations, qualitative and quantitative, to validate the effectiveness and the rendering quality of our approach. We demonstrate superior rendering results to traditional online rendering in common CG software, even close to the quality of offline ones. We also showcase the capability of *TransGS* on a series of immersive applications on various platforms, i.e., multi-source conditioned generation, cross-platform interactions, and rapid editing.

## 2 RELATED WORKS

### 2.1 Face Rendering

*Traditional Rendering Techniques.* A shading model is necessary for rendering photorealistic images. Phong shading[Phong 1998] interpolates surface normals for smooth highlights, followed by Physically Based Rendering (PBR)[Pharr et al. 2016] which models realistic light interactions, and later, Bidirectional Surface Scattering Reflectance Distribution Function (BSSRDF)[Jensen et al. 2001] extends this by considering light that penetrates and scatters within surfaces. Some following works[Borshukov and Lewis 2005; Habel et al. 2013; Hanrahan and Krueger 2023] focus on facial rendering, which significantly influence the color and realism of facial assets in digital imagery. d'Eon and Luebke [2007] enhances the realism of specular reflections on the skin, adapting them to different lighting conditions and angles. Donner et al. [2008] introduces inter-layer absorption to get higher skin rendering accuracy. Hery et al. [2016] introduces exponential and non-exponential path tracing to simulate more accurate photon behavior within human skin, enhancing realism and detail in rendered images.

*Volume Rendering.* Several studies have explored the application of Neural Radiance Fields (NeRF) [Mildenhall et al. 2021] for facial reconstruction and rendering. Gafni et al. [2021] utilizes deformable 3D Morphable Models (3DMMs) [Blanz and Vetter 2023] to capture dynamic facial movements. Athar et al. [2022]; Gao et al. [2022] enhance the modeling capabilities by incorporating additional Multilayer Perceptrons (MLPs) to account for deviations from the 3DMM-defined space. Grassal et al. [2022]; Zheng et al. [2022] introduce view and expression-dependent textures to achieve detailed textural fidelity. Furthermore, Lombardi et al. [2021] boosts rendering speeds by substituting traditional NeRFs with volumetric primitives, while Ma et al. [2021] utilizes a rendering-adaptive per-pixel decoder for enhanced rendering efficiency and Zielonka et al. [2023] significantly reduces optimization times by leveraging Instant-NGP [Müller et al. 2022]. However, while these techniques excel in reconstructing real-life scenes, their applicability to the translation of CG assets is constrained. Besides, as an implicit representation, NeRF and its variants are not directly compatible with existing graphics pipelines, preventing their board applications.

*3DGS Variants.* Recently, 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] proposed an explicit volume rendering pipeline, achieving both high-quality rendering and real-time performance. Research has explored various techniques to enhance 3DGS in avatar reconstruction and animation. Serval works [Ma et al. 2024; Qian et al. 2024; Rivero et al. 2024; Shao et al. 2024] use geometry-based parameterization to rig 3D Gaussians. Implicit representations like neural networks are introduced [Dhamo et al. 2023; Saito et al. 2024; Xiang et al. 2024; Xu et al. 2024] to enlarge the representation capacity of 3DGS. SplatFace [Luo et al. 2024] introduces a constrained splat-to-surface method for better facial animation. PSAvatar [Zhao et al. 2024] uses points-based geometry representation to fit the face deformation better. GaussianHead [Wang et al. 2024] embeds the Gaussian features to a multi-scale tri-plane structure to increase the representation power. However, these methods either disregard deformation-dependent color information [Qian et al. 2024;

Shao et al. 2024] or incorporate additional neural networks for fitting [Dhamo et al. 2023; Saito et al. 2024; Xu et al. 2024], posing challenges in balancing rendering quality and efficiency.

### 2.2 Face Generation

*PBR Facial Asset Generation.* The development of 3D face generation was notably advanced by AvatarMe [Lattas et al. 2020] and other improved versions [Lattas et al. 2023; Luo et al. 2021], which offer photorealistic 3D avatars that integrate well with conventional computer graphics pipelines. Li et al. [2020b] introduces an end-to-end framework to automate the generation of high-quality facial assets and rigs using mesh and UV maps. Cao et al. [2022] employs a universal avatar prior, trained on a vast dataset, to facilitate high-quality facial asset creation from simple phone scans. Zhang et al. [2023a] generate PBR facial assets conditioning on images or text prompts.

*GAN / NeRF Based Generation.* Several works use GAN [Hou et al. 2022, 2021; Isola et al. 2017; Karras et al. 2019, 2020; Papantoniou et al. 2023] and diffusion-based methods [Ponglertnapakorn et al. 2023; Zhang et al. 2024] to generate different views and relightable face images. To improve 3D consistency, NeRF has been widely adopted [Chan et al. 2021; Gu et al. 2022; Schwarz et al. 2020], with parametric human head models [Hong et al. 2022; Zhuang et al. 2022] to disentangle the rendering pose, identity, expression, and appearance. Implicit geometry structures like Signed Distance Function (SDF) [Or-El et al. 2022] and tri-plane [Chan et al. 2022] are leveraged for better 3D consistency. Subsequently, Wang et al. [2023] employs a diffusion model to generate tri-plane-based 3D models. While the transition from GAN-based image generators to tri-plane-based models has substantially increased 3D coherence, it still falls short of explicit 3D representations like meshes and voxel grids.

*GS-Based Generation.* 3DGS has been adapted to generate general 3D models [Liu et al. 2024; Tang et al. 2024; Yi et al. 2024]. For instance, DreamGaussian combines a generative 3DGS model with mesh extraction and texture refinement for better visual quality. Zhou et al. [2024] introduce HeadStudio, which innovatively creates animatable avatars from textual prompts by combining 3DGS with FLAME-based 3D representations. However, the rendering images of HeadStudio-generated assets are overly saturated, and the animation quality suffers due to the reliance on the SDS loss.

### 2.3 Post-Editing

*NeRF editing.* Recent work in 3D modeling addresses the editing limitations of NeRF-based and GAN-based methods, enhancing customization capabilities. For instance, FENeRF [Sun et al. 2022b] focuses on generating consistent and editable 3D portraits, IDE-3D [Sun et al. 2022a] enables precise adjustments to individual facial attributes. Aneja et al. [2023] uses the GAN model and Yue et al. [2023] uses diffuse-based models to modify facial images by text prompts SketchFaceNeRF [Lin et al. 2023] offers an intuitive interface for sketch controlled face generation. However, in these methods, all the modifications are done in the neural latent space, resulting in imprecise control and resolution bottleneck. Besides,
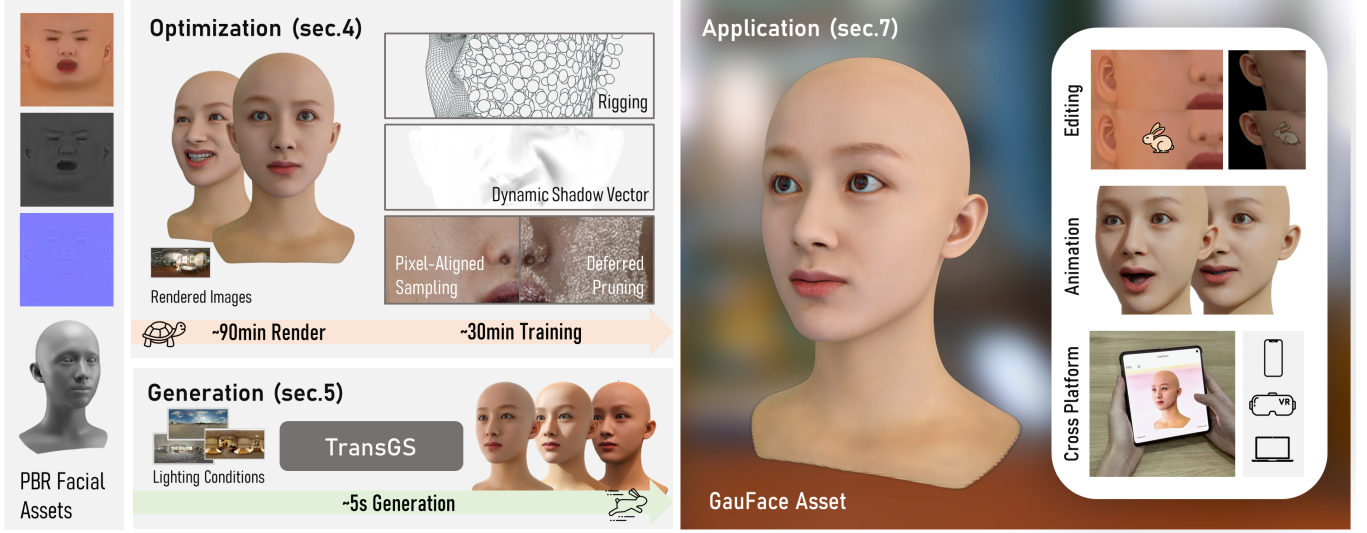
Fig. 2. **Overview.** We present two methods for obtaining relightable dynamic Gaussian facial assets. The first method (Sec. 4) render high-quality multi-view images and optimize the *GauFace* representation. The second method (Sec. 5), which we introduce as *TransGS*, directly generates *GauFace* assets from textures and models in approximately 5 seconds.

these methods are not compatible with the CG industry, where modifications are performed by editing the explicit geometry, image textures, etc.

*3DGS Editing.* Recent 3DGS-based editing methods like Fang et al. [2024] and Chen et al. [2024] introduce innovative text-based techniques for editing 3D Gaussian representations, facilitating precise adjustments to shapes and textures. However, to perform the editing, these methods require additional Gaussian optimizations, which is cumbersome for timely preview and iterations.

## 3 PRELIMINARY

### 3.1 Physically-Based Rendering Facial Assets

A Physically-Based Rendering (PBR) facial asset is a digital representation of a human face optimized for rendering using a physically accurate lighting and shading process. In this paper, we define a PBR facial asset as a collection of mesh neural geometry $G$, image textures $I$, including the diffuse map, normal and specular map, and expression blendshapes $\mathcal{B}$.

We define the UV mapping function as follows:

$$(x, y, z) = M(u, v; G), \tag{1}$$

which maps the points on the 2D UV space with coordinates $\boldsymbol{\mu} = (u, v)$ to the 3D position $(x, y, z)$ on the surface of neural mesh geometry $G$.

### 3.2 3D Gaussian Splatting

3DGS represents the 3D scene via a set of explicit 3D Gaussians. Each 3D Gaussian has multiple learnable parameters, including the center $\boldsymbol{\mu}$, covariance $\Sigma$, opacity $\sigma$, and view-dependent colors represented by Spherical Harmonics $\boldsymbol{c}$. The shape of each 3D Gaussian is defined

as:

$$X = e^{-\frac{1}{2}\boldsymbol{\mu}\Sigma^{-1}\boldsymbol{\mu}}. \tag{2}$$

When rendering a novel view, the 3D Gaussians are projected to 2D screen space ordered by depth. The color $C$ of a pixel is computed by $\alpha$-blending of these projected Gaussians:

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j), \tag{3}$$

where $c_i$ is the evaluated view-dependent colors.

During optimization, the gradients of all attributes are computed by comparing the difference between the rendered image and the ground truth via differentiable rendering. Pruning and densification are applied according to the gradient statistics, enabling efficient scene representation and fast optimization.

Next, as illustrated in Fig. 2, we first introduce the design and optimization of *GauFace* in Sec. 4. Following that, we present our generative model, *TransGS*, in Sec. 5. We then conduct a detailed evaluation in Sec. 6 and discuss the broad applications empowered by our approach in Sec. 7.

## 4 *GAUFACE* : BRIDGING PBR FACIAL ASSETS TO 3D GAUSSIAN SPLATTING

Recently, several works have extended 3DGS to handle facial animation [Chen et al. 2023; Saito et al. 2024; Wang et al. 2024]. However, they focus on reconstructing from the real world and novel view synthesis. There lacks an efficient method for converting CG assets to 3DGS for efficient rendering.

We introduce *GauFace* to bridge the gap between fine-grained PBR facial assets and high-quality and efficient 3DGS representation. Leveraging a uniform topology and UV mapping common in CG facial asset libraries [Li et al. 2020a], as shown in Fig.3, we define
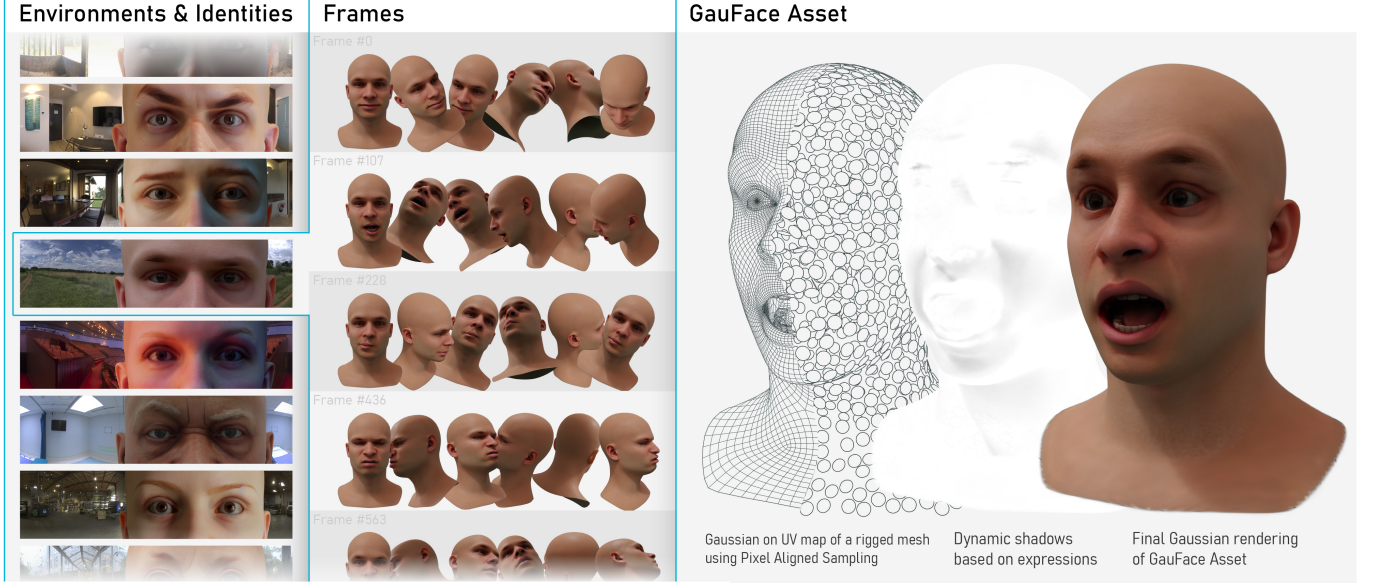
Fig. 3. **PBR facial assets and *GauFace* representation.** *Left:* We collect 143 facial assets under 134 lighting conditions, with a total of 1,023 combinations. *Middle:* For each combination, we render 1,071 frames under 153 different expressions with random camera positions. *Right:* Our *GauFace* asset defines the center of Gaussians on the UV map consistent across different identities and introduces dynamic shadow vectors to disentangle the deformation-dependent and deformation-agnostic shading effects.

Gaussian points within the shared UV space. This approach leads to a natural support of blendshape-based facial animation. To decouple the deformation-dependent shading effects from the deformation-agnostic parts, we construct *Dynamic Shadow Vector* to efficiently adapting to the dynamic lighting and shadows produced by facial animation.

Specifically, a *GauFace* asset is a collection of $N$ orderless Gaussian points $A = \{p_i\}_{i=1}^{N}$. Each Gaussian point $p$ contains the following parameters:

$$p = (\boldsymbol{\mu}, d, \boldsymbol{s}, \theta, \sigma, \boldsymbol{c}, \boldsymbol{l}), \tag{4}$$

where $\boldsymbol{\mu} = (u, v)$ is the Gaussian position defined on the UV space, $d$ is the deviation to the mesh surface, $\boldsymbol{s}$ and $\theta$ define the shape of the Gaussian point, $\sigma$ is the opacity of the Gaussian point, $\boldsymbol{c}$ is the color represented via Spherical Harmonics, and $\boldsymbol{l}$ is a dynamic shadow vector to represent the deformation-dependent shadings.

### 4.1 Rigging 3D Gaussians

We rig 3D Gaussians to the mesh surface to support facial animation. The 3D position of a Gaussian point $p$ is obtained as follows:

$$\mu_{3D} = M(\boldsymbol{\mu}; G) + d\boldsymbol{n}_f, \tag{5}$$

where $M$ is the UV mapping function, $\boldsymbol{\mu}$ is the UV position, $f$ is the contained triangle, $d$ is a learnable parameter and $\boldsymbol{n}_f$ is the normal vector of the triangle. This defination allows the Gaussian point to move according to the mesh deformation, and also deviate from the mesh surface along the normal direction.

We follow Guédon and Lepetit [2024]; Huang et al. [2024] to constrain $p$ as thin shells, by setting the scaling vector as follows:

$$\boldsymbol{s} = [\epsilon, s_1, s_2], \tag{6}$$

where $\epsilon$ is a pre-defined small value, and $s_1, s_2$ are optimizable parameters. For rotation, we follow [Guédon and Lepetit 2024] to first calculate the rotation matrix of triangle $f$ as $R_f = [R^{(0)}, R^{(1)}, R^{(2)}]$. Then, the rotation matrix of $p$ is defined as follows:

$$R_p = [R^{(0)}, xR^{(1)} + yR^{(2)}, -yR^{(1)} + xR^{(2)}] \tag{7}$$

where $x + iy$ is a normalized complex number. Eq. 6 and Eq. 7 constrain 3D Gaussians to be thin shells attaching to the mesh surface, where two degrees of freedom ($s_1, s_2$) are allowed to control the size of 3D Gaussians on the tangent space of their attached triangles, and one degree of freedom ($\theta$) is allowed to rotate along the normal direction of their attached triangles.

Since our 3D Gaussians are attached to a mesh, it naturally supports blendshape animations by updating the mesh vertex positions at every frame. Given $B$ different blendshape vectors, the mesh vertex positions at frame $i$ are updated as follows:

$$V_i = \bar{V} + \mathcal{B}\boldsymbol{b}_i, \tag{8}$$

where $\bar{V}$ is the mesh vertices of neural geometry $G$, $\mathcal{B}$ is the blendshape matrix and $\boldsymbol{b}_i \in [0, 1]^B$ is the the blendshape weights at frame $i$. The 3D position of each 3D Gaussian at each frame is updated via Eq. 5 and the rotation matrix is updated via Eq. 7.

### 4.2 Dynamic Shadow Vector for Deformation-Dependent Shading

To adapt 3DGS to handle facial animation, deformation-dependent shading effects need to be supported. Recent methods either omit this effect [Qian et al. 2024; Shao et al. 2024], or use neural networks to infer frame-dependent colors [Saito et al. 2024; Xu et al. 2024]. The latter approach entangles deformation-dependent shading effects

with deformation-agnostic parts, which hinders the generalizability to unseen facial expressions. Instead, we fully leverage the geometry prior provided by the PBR facial dataset, and explicitly define attributes for the deformation-dependent shading effects.

We observe that for a face under a specific lighting condition, the change of shadows and self-occlusions contributes most to the deformation-dependent colors. These effects are mainly view-agnostic. Thus, for each Gaussian point $p$, we define a shadow vector $l \in [0, 1]^B$ to express these changes caused by the facial deformations. The final color of a Gaussian point is calculated as follows:

$$c^*_{\omega,i} = l^T b_i \, c_\omega, \tag{9}$$

where $b_i$ is the blendshape weight of frame $i$, and $c_\omega$ is the color expressed by Spherical Harmonics at camera view $\omega$. $l$ is optimized in the same way as the SH color attributes $c$, receiving gradients from differentiable rendering. Introducing the dynamic shadow vector for deformation-dependent shading only adds $B$ new parameters to each Gaussian point, which maintains the efficiency of the color representation.

### 4.3 Constraining *GauFace* for Generative Modeling

The original 3DGS optimization algorithm initializes very few Gaussian points in the beginning and gradually densifies them according to the complexity of the target scene. Although this design ensures high efficiency of the 3DGS representation, it makes the optimized Gaussian asset too scene-specific to form a learnable distribution.

*Pixel Aligned Sampling.* To constrain the distribution of Gaussian points, in *GauFace*, we sample uniformly on the UV plane during initialization, and then lock the position and turn off densification and pruning operations during optimization. We call this *Pixel Aligned Sampling*. This strategy guarantees that all different Gaussian assets share the same number of Gaussian points and their UV positions after optimization, which leads to a unified sampling pattern across all different assets.

*Deferred Pruning.* Although *Pixel Aligned Sampling* provides a rigid pattern for generative modeling, it sacrifices the rendering efficiency. Because the initialization is over-parametrized in many face parts, and pruning of unnecessary Gaussian points is disabled. However, we found that during optimization, unnecessary Gaussian points on over-parameterized areas lead to close-to-zero opacity values. This is because the opacity of all Gaussian points will be reset to zero once every several iterations, and unnecessary Gaussian points then receive little image-based gradients since the area is already represented well. As shown in Fig. 4, after optimization finished, nearly half of the Gaussians can be pruned via opacity thresholding with minimum visual degradation. Thus, we can do pruning just before the runtime. In this way, we can keep the Gaussian point sampling positions as a prior during generative modeling while achieving similar rendering efficiency as the vanilla 3DGS.

### 4.4 *GauFace* Dataset

We propose the *GauFace* dataset to construct a mapping from PBR facial assets to their *GauFace* counterparts. We collect 143 PBR facial assets generated from Zhang et al. [2023a], including the meshes, 51 ARKit blendshapes, and diffuse, normal, and specular maps in
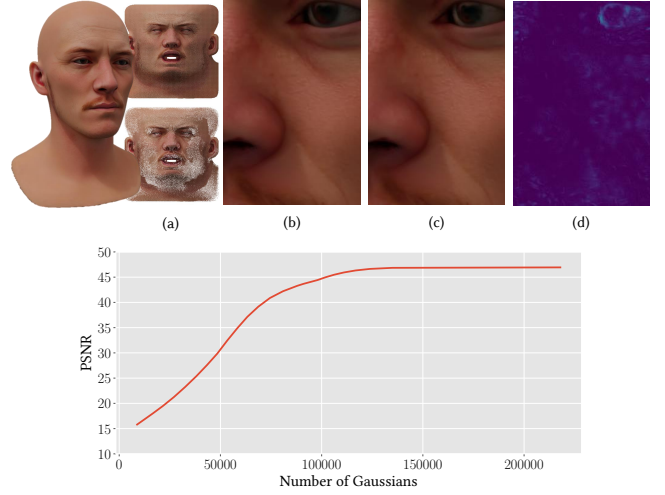


(a)    (b)    (c)    (d)

Fig. 4. **Deferred Pruning.** *Upper*: (a) *GauFace* and it's UV space base color visualization before and after pruning. (b) Without pruning, 22k points. (c) Pruned with opacity $\sigma \leq 0.1$, 11k points. (d) Difference between (b) and (c). *Lower*: PSNR vs. Number of Gaussians Curve. PSNR is calculated on 306 testing images with different expressions and camera positions. Gaussian points are pruned with different opacity value.

4K resolution. Each asset contains 5 components: *Foreface, Back-head, Teeth, Lefteye* and *Righteye*, which share the same topology as ICT-FaceKit [Li et al. 2020a]. We render the facial assets under 134 different lighting conditions, leading to a total of 1,023 combinations. For each combination, we prepare a human performance of 153 different expressions and render 7 different views for each expression using a customized PBR shader with Blender Cycles. This leads to 1,071 images per combination and over 1 million images in total.

We optimize each combination to a *GauFace* asset. For each *GauFace* asset, the Gaussian points are uniformly initialized on the 4K UV map, with a density of 10 pixels per point for the *Foreface* and 16 pixels per point for the remaining. This leads to a total of 228,083 Gaussian points per asset. Each *GauFace* asset is optimized with the same hyperparameters and losses as Kerbl et al. [2023], with the learning rate of the dynamic shadow vector $l$ set to $1/B$ of the learning rate of the SH base color component. We optimize each *GauFace* for 30,000 iterations, with around 30 minutes on a NVIDIA RTX 3090.

## 5 *TRANSGS* : INSTANT GAUSSIAN ASSET GENERATION

On top of the *GauFace* representation and dataset, we propose *TransGS*, a Gaussian Splatting Translator that maps PBR facial asset to its *GauFace* counterpart in *seconds*, with close to off-line rendering quality. *TransGS* supports inference-level relighting via HDR environment maps and editing by geometry and textures.

As shown in Fig. 5, conditioning on the image textures $I$, geometry $G$ and lighting information $L$, our generator $\mathcal{G}$ synthesizes the corresponding *GauFace* asset $A$:

$$A = \mathcal{G}(I, G, L). \tag{10}$$

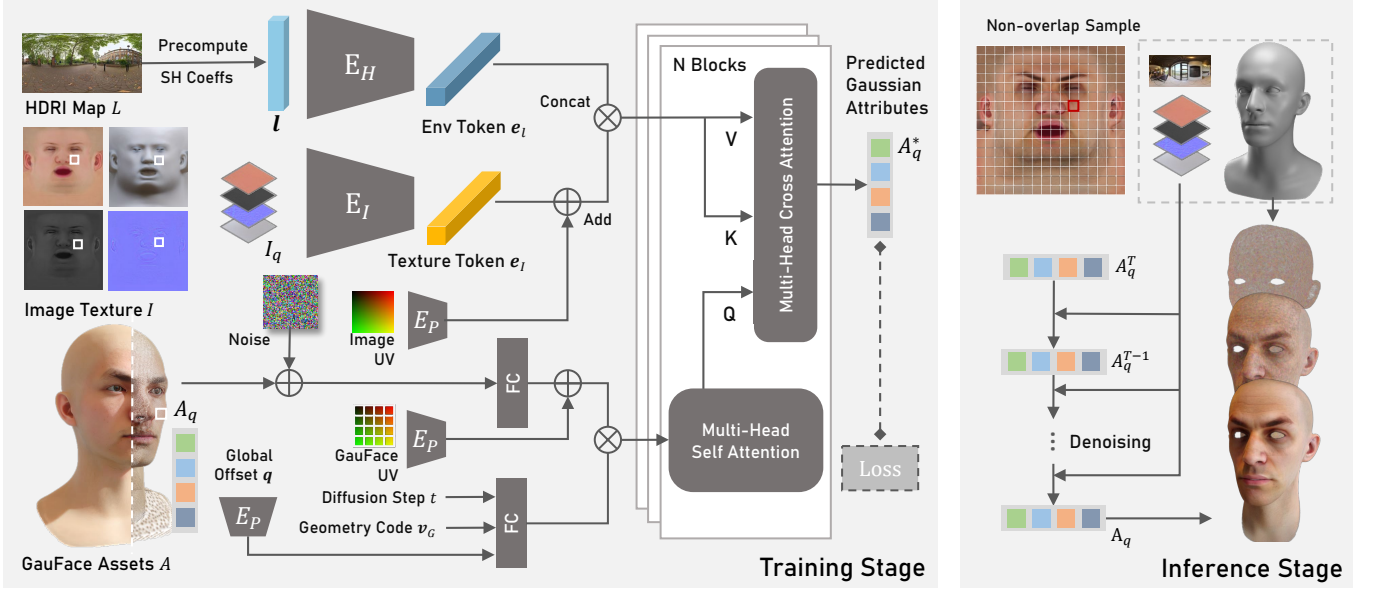Fig. 5. **TransGS architecture.** We condition *TransGS* on the image textures $I$, geometry code $\boldsymbol{v}_G$ and HDRI map $L$, to generate the *GauFace* asset $A$ in a patch-based manner. *Left:* during training, a random global offset $\boldsymbol{q}$ is sampled, and the corresponding Image patch $I_q$ and *GauFace* patch $A_q$ are fed to the diffusion transformer. *Right:* at inference, the full *GauFace* asset can be synthesized in a single pass.

We adopt a diffusion transformer as our backbone and design a patch-based learning strategy to handle the vast number of Gaussian points per asset (Sec. 5.1). We detail the model condition on PBR facial assets in Sec. 5.2 and describe a novel positional encoding to focus the transformer on Gaussian-texture relations in Sec. 5.3.

## 5.1 Patch-based Diffusion Transformer

Though we have constrained *GauFace* assets via multiple design choices (Sec. 4.3), the mapping between a PBR facial asset under specific lighting, and its *GauFace* translation is still one-to-many. Fig. 6 visualize the difference between two *GauFace* assets optimized with the same training data and settings. Gaussian points of two optimized *GauFace* assets on the same sampled UV position have different attribute values. Thus, we utilize a diffusion process to learn the one-to-many mapping better. Besides, Gaussian assets are essentially structureless point clouds, of which Transformer is preferred for learning correlations [Nichol et al. 2022] as it is agnostic to the input sequence order without positional encoding [Vaswani et al. 2017].

Since each *GauFace* asset contains a huge number of Gaussian points (roughly 230k), it is impossible to treat all Gaussian points together as a single sequence. Besides, we observe that *GauFace* assets share similarities in the same UV region across different identities. Thus, we split the UV space into small patches and couple all Gaussian points inside a patch together to serve as the input. This patch-based processing strategy not only reduces the sequence length of the transformer but also augments the training data by treating small patches independently.

Specifically, given a Gaussian asset $A$, we sample a random *global offset* $\boldsymbol{q} \in [0, 1]^2$ on the UV plane and collect all the Gaussian points
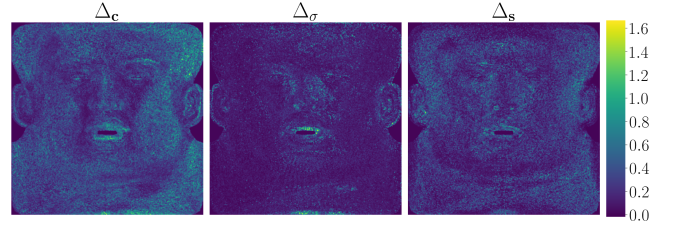


Fig. 6. Difference between attributes of two *GauFace* optimized from the same training images, visualized by plotting attributes to the UV sampling position of Gaussian points. $\Delta_c, \Delta_\delta, \Delta_s$ are the difference of SH base color, opacity, and specular intensity under two different runs, respectively.

inside the square patch $[\boldsymbol{q}, \boldsymbol{q} + \boldsymbol{P}]$, denoting as a *Gaussian patch* $A_q$. $\boldsymbol{P}$ is a hyperparameter that determines the size of the patch. The denoising process is defined as follows:

$$A_q = \mathcal{M}(A_q^t, t, Y), \tag{11}$$

where $A_q^t$ is the noised attributes of the Gaussian patch, $t \in [1, ..., T]$ is the diffusion timestep, $Y$ is the condition and $\mathcal{M}$ is the denoising network. $A_q^t$ is mapped to Gaussian tokens $\boldsymbol{e}_A$ through a linear layer, $t$ is mapped to a timestep token $\boldsymbol{e}_T$ similar to DDPM [Ho et al. 2020].

## 5.2 PBR Facial Asset Conditioning

The condition of our transformer, $Y$, includes image textures, geometry, and lighting. *Image Textures* are the main conditions of the model. For each PBR facial asset, we concatenate the diffuse, normal, and specular map along the feature dimension to serve as the image condition $I$. For each Gaussian patch $A_q$, we select the *image patch*
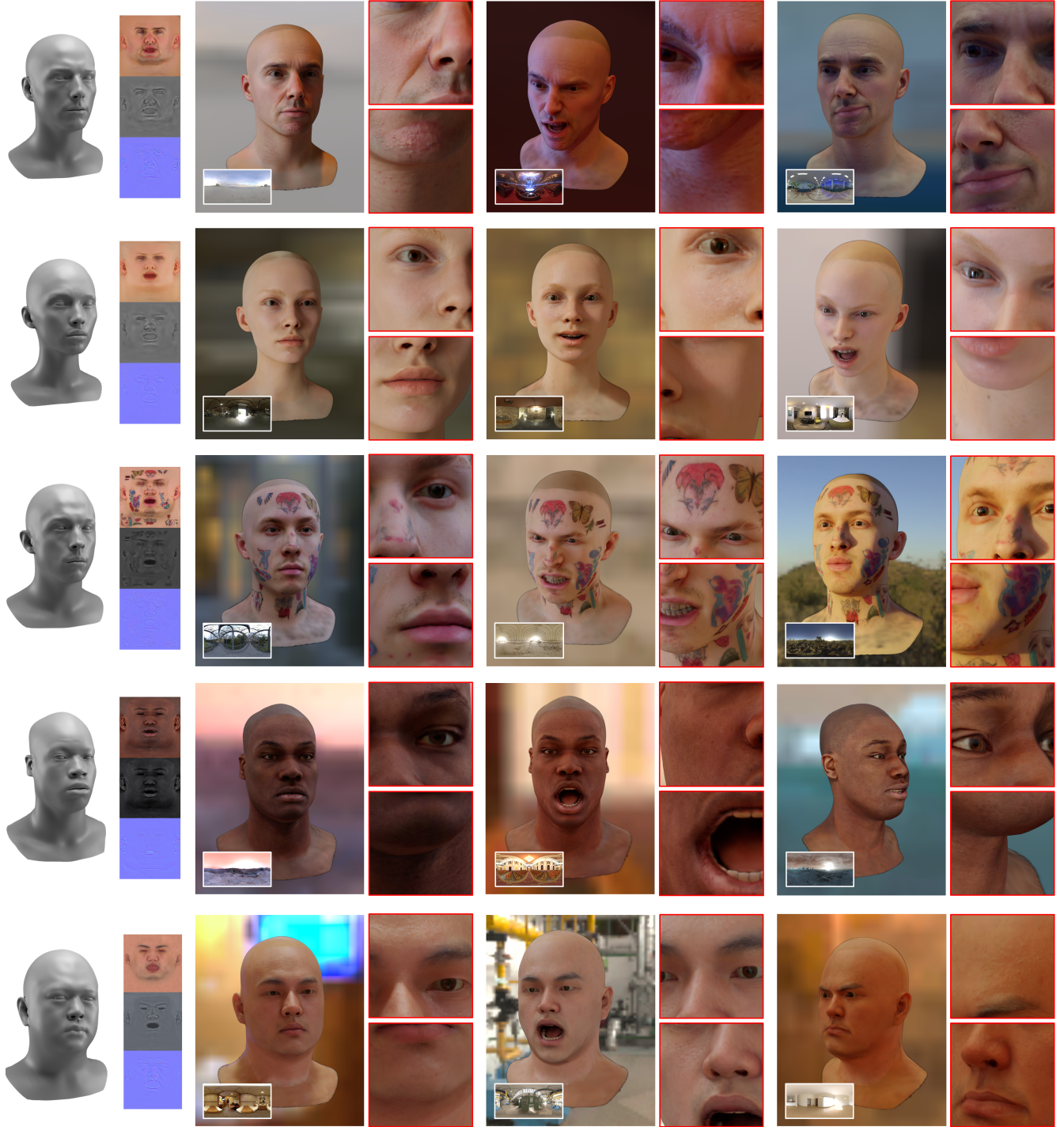
Fig. 7. **GauFace** assets synthesized from **TransGS** . Conditioning on the PBR facial asset, *TransGS* generates the *GauFace* counterpart in **5 seconds**, with close to off-line rendering quality and supports real-time facial animation and interaction with 30fps@1440p on a *Snapdragon® 8 Gen 2* mobile phone. The PBR facial assets are obtained from diverse sources. 1-3: Generated from *DreamFace* [Zhang et al. 2023a]. 4: Downloaded from web. 5: Scanned from a Light Stage [Debevec et al. 2000]. All figures are rendered under our cross-platform Unity3D *GauFace* render engine.

$I_q$ with the same global offset and UV patch size and map it to image tokens via the image encoder $E_I$, i.e., $\boldsymbol{e}_I = E_I(I_q)$.

*Geometry* information is injected via a PCA-based *geometry code*. We utilize a geometry PCA space similar to [Li et al. 2020a]. For each input geometry $G$, the geometry code $\boldsymbol{v}_G$ is obtained by projecting $G$ to the PCA bases. $\boldsymbol{v}_G$ is mapped to the model latent space through a linear layer and then added to the timestep token $\boldsymbol{e}_T$.

*Lighting* is extracted from the HDR environment map by Spherical Harmonics (SH) decomposition [Ramamoorthi and Hanrahan 2001]. Given an environment map $L$, we compute the 12-order SH coefficients $\boldsymbol{l}$, and map it to a lighting token $\boldsymbol{e}_L$ via a linear layer. Additionally, to provide detailed shadow information, we bake a low-resolution shadow map $I_s$ and concatenate it with the image condition $I$, to serve as an additional input to the image encoder.

## 5.3 UV Positional Encoding

We design a novel positional encoding (PE) based on the UV location of inputs to guide the transformer's attention toward textures proximal to the Gaussian sampling positions. Specifically, given a UV sampling position $\boldsymbol{\mu} = (u, v)$, the positional encoding (PE) is calculated as:

$$PE(\boldsymbol{\mu}) = \begin{cases} sin(2^j \pi u), \ j = 4k & cos(2^j \pi u), \ j = 4k+1 \\ sin(2^j \pi v), \ j = 4k+2 & sin(2^j \pi v), \ j = 4k+3 \end{cases}, \quad (12)$$

where $j$ is the dimension of the encoding. The positional encoding is further processed through a projection MLP $E_P$.

Given a pair of patch-based data, we apply positional encoding separately to the Gaussian patch $A_q$, the image patch $I_q$, and the global offset $\boldsymbol{q}$. For each Gaussian patch with a collection of UV sampling positions $\boldsymbol{\mu}_{A,q}$, we add the *relative positional encoding* calculated by $E_P(PE(\boldsymbol{\mu}_{a,q} - \boldsymbol{q}))$ to the Gaussian tokens $\boldsymbol{e}_A$. In addition, given the UV pixel coordinates $\boldsymbol{\mu}_{I,q}$ of the image patch, a similar positional encoding, $E_P(PE(\boldsymbol{\mu}_{I,q} - \boldsymbol{q}))$ is added to the image tokens $\boldsymbol{e}_I$. To inject the global offset information to the denoising network, we add the *global positional encoding*, $E_P(PE(\boldsymbol{q}))$ to the diffusion timestep token $\boldsymbol{e}_T$.

## 5.4 Training

Our denoising network $\mathcal{M}$ is a decoder-only transformer. During training, we draw random tuples $(A_q, t, I_q, G, L)$ from the training set and add random Gaussian noise to corrupt $A_q$ to $A_q^t$. After obtaining all input tokens, i.e. Gaussian tokens $\boldsymbol{e}_A$, Image tokens $\boldsymbol{e}_I$, timestep token $\boldsymbol{e}_T$, and light token $\boldsymbol{e}_L$, we concatenate $\boldsymbol{e}_A$ with $\boldsymbol{e}_T$ to serve as the query to $\mathcal{M}$, and concatenate $\boldsymbol{e}_I$ with $\boldsymbol{e}_L$ to be the key and value.

We divide the training into two stages: a main stage and a fine-tuning stage. During the main stage, we only consider the simple loss :

$$\mathcal{L}_s = ||A_q - \mathcal{M}(A_q^t, t, I_q, G, L)||_2^2. \quad (13)$$

During the fine-tuning stage, we additionally compute the image loss based on the rendered images of the estimated and ground-truth Gaussian assets:

$$\mathcal{L}_i = ||\mathcal{R}(A_q^*) - \mathcal{R}(A_q)||_1 + SSIM\left(\mathcal{R}(A_q^*) - \mathcal{R}(A_q)\right). \quad (14)$$

where $A_q^*$ is the model prediction and $\mathcal{R}$ is the *GauFace* rendering process. The total loss at this stage is a weighted summation:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_s + \lambda \mathcal{L}_i. \quad (15)$$

## 6 EXPERIMENTS

In this section, we present the experimental results of *TransGS* . We first introduce the implementation details, then present a gallery of *TransGS* generated assets. Then, we compare the rendering quality of our generated assets against typical offline and online render engines through visualization and a user study. We also compare with neural rendering methods under a time-constrained setting. Finally, we conduct a comprehensive evaluation and ablation studies of our modules and key design choices.

*Implementation Details.* We train our model using the *GauFace* dataset (Sec. 4.4). We split it into a training set with 983 assets and a test set with 40 assets. We train separate models for the five facial parts (foreface, backhead, teeth, and left/right eyes, as described in section 4.4) and set the patch size as $\mathbf{P} = (\frac{1}{256}, \frac{1}{256})$, leading to 256 non-overlapping patches per facial component. During training, we randomly sample positions on the UV map as starting points. At inference, we use a consistent set of 256 non-overlapping starting points to cover the entire UV space.

Our *Foreface* model contains a 12-layer transformer decoder with 512 latent dimensions and a CNN-based image encoder similar to the ControlNet Conditional Embedding [Zhang et al. 2023b]. Models for other facial parts share the same architecture but are scaled down in size. During training, we use 100 diffusion steps and reduce this to 10 steps for inference. We first train the models for 800 epochs in the main stage and finetune it for another 100 epochs with the image loss activated. The training of the *Foreface* model took 5 days on 8 NVIDIA RTX 4090 and others took 1 day each on a single NVIDIA RTX 4090. All modules are trained from scratch in an end-to-end manner. During inference, all patches can be synthesized in a single forward pass, taking 5 seconds in-total on a NVIDIA RTX 4090.

Once trained, *TransGS* can convert PBR facial assets to *GauFace* translations in seconds. We showcase the synthesized *GauFace* assets under Gaussian rendering in Fig. 7. The *GauFace* assets are conditionally generated under a variety of lighting conditions and diverse backgrounds, demonstrating the robustness and versatility of our approach. Whether under soft, ambient lighting or stark, directional illumination, the avatars retain their high level of realism, showcasing dynamic shading and realistic light interaction. From neutral, calm expressions to more animated and intense poses, the rendered avatars capture and convey emotions with impressive accuracy.

## 6.1 Comparision

Here, we provide a thorough evaluation of the rendering quality of *TransGS* -generated assets.

*6.1.1 Comparison with CG renders.* We qualitatively compare the visual quality and the rendering speed of *TransGS* synthesized assets against Blender Cycles, Blender EEVEE, and Unity3D. Blender Cycles is an offline ray-tracing rendering engine for production-level visual quality. It also defines the quality upper bound of our

Blender Cycles (0.1fps)   Optimized GauFace Asset (>100fps)   TransGS Generated Asset (>100fps)   Blender EEVEE (12.5fps)   Unity3D Built-in Pipeline (>100fps)
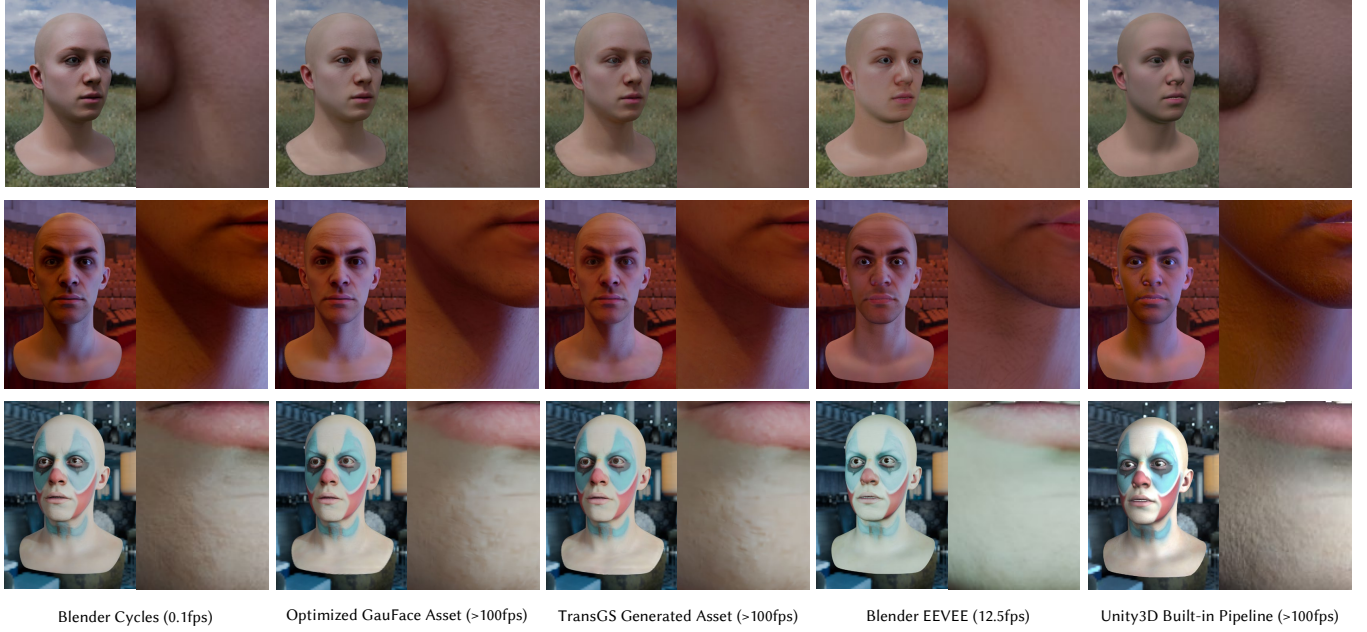
Fig. 8. **Quality comparisons of different rendering pipelines.** We provide the same geometry, image textures, and HDR environment maps to *TransGS* and traditional rendering pipelines. All images are rendered with 1080p resolution on a PC with Intel i9-12900K and NVIDIA RTX 4080. Note that the rendering quality of *GauFace* and *TransGS* assets are upper-bounded by Blender Cycles, which provides the ground-truth training images.



| Render | TransGS<br>——<br>5s | GA-4K<br>Sparse-153<br>30m | GA-4K<br>Sparse-51<br>30m | GHA-6K<br>Sparse-153<br>45m | GHA-6K<br>Sparse-51<br>45m | INSTA-16K<br>Sparse-153<br>17m | INSTA-16K<br>Sparse-51<br>17m |
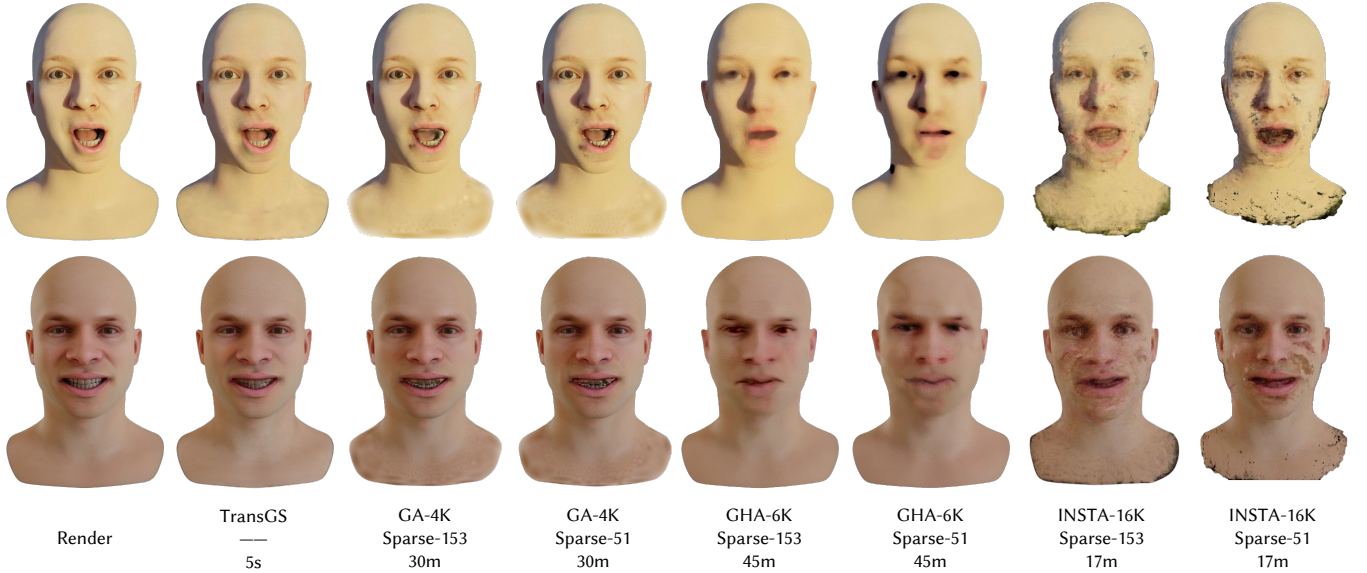
Fig. 9. **Render quality and generation time comparison against other methods**. While *TransGS* -generated assets deliver aesthetic rendering quality in 5s, GA fails to model the mouth region under mixed blendshape activations. GHA and INSTA fail to provide reasonable results due to both data and optimization constraints.

method as our dataset relies on Blender Cycles. Blender EEVEE and Unity3D are rasterization-based engines for interactive speed. As shown in Fig. 8, the *GauFace* assets synthesized by *TransGS* deliver aesthetic visual quality comparable to Blender Cycles with natural skin shading from HDR lighting and subsurface scattering, detailed skin textures, sharp shadows, and self-occlusions. In addition, due to the efficient representation, *GauFace* assets support real-time animation and interaction even on mobile platforms (Fig. 13). Note
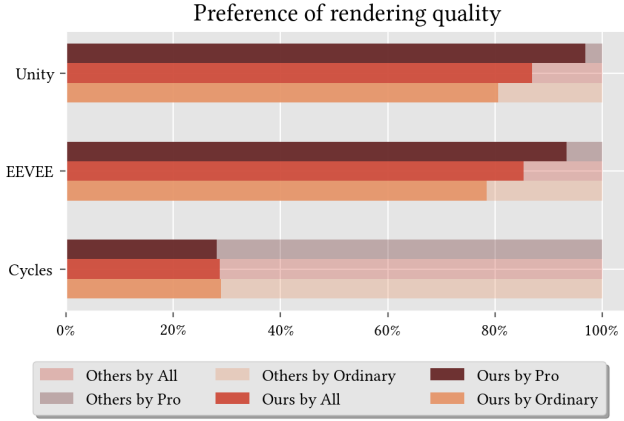
Fig. 10. **Quantitative Results of User Study**. Both professional artists (Pro) and ordinary people (Ordinary) prefer the rendering quality of our *TransGS* generated *GauFace* asset to Unity3D's build-in pipeline (Unity) and Blender EEVEE.

that the visual quality of *GauFace* assets is upper-bounded by that of Blender Cycles, of which a customized PBR shader is applied. A better shader can potentially lead to better *GauFace* rendering.

*User study.* In the CG industry, rendering quality has always been a subjective measure evaluated by humans. Therefore, we conducted a user study to comprehensively evaluate the rendering quality of *TransGS* generated *GauFace* assets. We designed pairwise comparisons with Blender Cycles, Blender EEVEE, and Unity3D's build-in pipeline, asking the question, "Which image has better facial rendering?" For fairness and comprehensiveness, we included 11 different identities and lighting scenarios. Each questionnaire presented 11 randomly chosen pairs of our renderings alongside one comparison object, with the display order randomized between our renderings and the comparison.

We distributed the questionnaire to both professional artists and ordinary people. We received a total of 23 responses from artists and 35 responses from non-artists. As shown in Fig. 10, over 80% of the participants preferred our rendering quality to those of Unity3D or Blender EEVEE. As a sanity check, over 70% participants prefer Blender Cycles to ours, as our rendering quality is upper-bounded by Blender Cycles. It is worth mentioning that professional artists tend to prefer our results over those from Unity and EEVEE, compared to ordinary individuals.

*6.1.2 Comparison with neural methods.* We first compare the properties of *TransGS* against other 3DGS-based facial avatars. As illustrated in Tab. 1, explicit methods like SplattingAvatar (**SA**) [Shao et al. 2024], GaussianAvatars (**GA**)) [Qian et al. 2024] and 3D Gaussian Blenshapes (**GBS**) [Ma et al. 2024] either omit the Deformation-Dependent Shading effects, or mix that with the Deformation-Agnostic parts (**GBS**). Implicit methods like FlashAvatar (**FA**) [Xiang et al. 2024], Gaussian Head Avatar (**GHA**) [Xu et al. 2024] and Relightable Codec Avatar (**RCA**) [Saito et al. 2024] leverage neural networks to enhance representation capacity, which sacrifice 3DGS's affinity with traditional rendering pipelines. Note that all methods except us

Table 1. **Comparison of properties of Gaussian-based facial avatars.** *D* shorts for 'Deformation'. *E* stands for explicit representation and *I* stands for implicit representations.

| Method | Geometry Representation | Shading Represenation | | Relighting Support | Instant Generation |
|---|---|---|---|---|---|
| | | *D*-Agnostic | *D*-Dependent | | |
| SA | *E* | *E* | ✗ | ✗ | ✗ |
| GA | *E* | *E* | ✗ | ✗ | ✗ |
| FA | *I* | *E* | ✗ | ✗ | ✗ |
| GBS | *E* | *E* | *E* | ✗ | ✗ |
| GHA | *I* | *I* | *I* | ✗ | ✗ |
| RCA | *I* | *I* | *I* | ✓ | ✗ |
| *TransGS* | *E* | *E* | *E* | ✓ | ✓ |

require per-avatar optimization, which typically consumes minutes or hours to finish on a high-end PC.

To further evaluate the performance and throughput of our novel pipeline, we compare our synthesized results against state-of-the-art optimization-based volume rendering pipelines. Since our method does not require additional data preparation and optimization time, for a fair comparison, we record and limit the runtime of other optimization-based methods.

Specifically, we pick 6 identities from our *GauFace* test set and design two evaluation variants: Sparse-153 and Sparse-51, which contain 3 and 1 rendered images of individual blendshape activations under random camera positions, respectively. Each case has a test set of another 153 images under different expressions and camera positions. Since each image requires roughly 10 seconds to render with an NVIDIA RTX 4080, the training set construction time of Sparse-153 takes 25m30s for each identity, and that of Sparse-51 takes 8m30s.

We compare the rendering quality of our *TransGS* synthesized asset against three recent advances in volume rendering pipelines designed explicitly for facial rendering: **GA**, **GHA** and **INSTA** [Zielonka et al. 2023]. GA and GHA are Gaussian-based methods, while INSTA is a NeRF variant. For a fair comparison, we replace GA's FLAME-based representation with our ground-truth geometry to improve its results. GHA and INSTA implicitly relate the volume representation to the geometry via neural networks, so we retain their original pipelines. For GHA, we use images and camera poses from our dataset, optimize the geometry with GHA's network, and perform the two-stage training in their official repo to refine the Gaussians. INSTA requires a segment of monocular RGB video as input; we extract head parameters from the dataset at various angles and expressions, following INSTA's method to obtain facial segmentation and landmarks. We train each method with two versions, varying the number of optimization iterations, denoted with a tail mark (e.g., GA-4K represents the GA method optimized for 4K iterations).

In Table. 2, we present the common PSNR, SSIM, and LPIPS metrics, along with the optimization time for each method. Additionally, we compute the lower 90% quantile of PSNR to emphasize the worst cases. Generally, more optimization iterations result in better performance, with Sparse-153 consistently outperforming Sparse-51, highlighting the data-intensive nature of these methods. However, INSTA is an exception; doubling the optimization iterations leads to worse results due to its facial landmark tracking failing on side

Table 2. **Visual quality and time consumption of various facial volume rendering methods**. GA, GHA, and INSTA take minutes of data preparation and optimization to reconstruct a scene, while *TransGS* generates the *GauFace* representation with the best rendering quality and takes only 5 seconds on a single NVIDIA RTX 4090.

| Data | Methods | PSNR$^\uparrow$ | SSIM$^\uparrow$ | LPIPS$^\downarrow$ | PSNR(90%)$^\uparrow$ | Time$^\downarrow$ |
|---|---|---|---|---|---|---|
| Sparse-153 | GA-4K | 28.02 | 0.9706 | 0.172 | 23.87 | 30m |
| (25m30s) | GA-2K | 25.75 | 0.9666 | 0.181 | 21.11 | 14m |
| | GHA-6K | 30.50 | 0.9722 | 0.112 | 27.70 | 45m |
| | GHA-3K | 29.50 | 0.9703 | 0.120 | 26.60 | 24m |
| | INSTA-16K | 18.31 | 0.8080 | 0.259 | 13.61 | 17m |
| | INSTA-8K | 18.39 | 0.8138 | 0.255 | 13.62 | 9m |
| Sparse-51 | GA-4K | 27.94 | 0.9695 | 0.181 | 23.56 | 30m |
| (8m30s) | GA-2K | 25.99 | 0.9667 | 0.172 | 21.81 | 14m |
| | GHA-6K | 28.64 | 0.9691 | 0.112 | 25.61 | 45m |
| | GHA-3K | 28.03 | 0.9677 | 0.120 | 25.06 | 24m |
| | INSTA-16K | 17.62 | 0.7945 | 0.265 | 13.33 | 17m |
| | INSTA-8K | 17.85 | 0.8004 | 0.261 | 13.35 | 9m |
| - | *TransGS* | **35.50** | **0.9936** | **0.045** | **34.01** | **5s** |

or back view renderings. Notably, even the most time-consuming method (GHA-6K on Sparse-153 with 45m) fails to achieve comparable visual quality to the *TransGS* synthesized representation.

The qualitative comparisons in Fig. 9 support the same conclusion. Compared to the Blender Cycles rendered ground-truth, the *TransGS* synthesized asset delivers similar visual quality. In contrast, GA struggles with modeling the inner mouth region under mixed blendshape activations due to insufficient training data. GHA exhibits significant blurring effects because its super-resolution module fails to converge within the given time constraints. INSTA fails to accurately model face geometry, resulting in prominent artifacts even in frontal renderings, primarily due to issues with facial landmark tracking.

## 6.2 GauFace Evaluations

In this section, we evaluate the performance of our *GauFace* representation and its key design choices.

*6.2.1 GauFace representation.* Our *GauFace* representation serves as a bridge, connecting PBR facial assets to Gaussian representation, and Gaussian representation to generative modeling. Here, we evaluate the performance of *GauFace*, illustrating its superior rendering quality while keeping a compact and efficient pipeline.

We use the same six PBR facial assets as described in Section 6.1.2, rendering eight images under 153 different expressions and frontal perspectives. For each expression, six images are used for training and two for evaluation. We omit rendering the *Backhead* due to its minimal appearance variation and limited visibility in frontal view renderings. We compare the render quality of *GauFace* optimized assets against GA, GHA, and INSTA, training all methods for the recommended number of iterations to maximize their representation capabilities.

As shown in Table 3, our *GauFace* representation achieves the highest visual quality. GA performs similarly well due to its explicit geometry representation. However, GA's omission of deformation-dependent shading effects slightly reduces its representation power compared to ours. GHA exhibits significant deviations in expressions and positions attributed to the limited fitting capability of its

Table 3. **Comparison of visual quality on our synthetic data.**

| Methods | PSNR$^\uparrow$ | SSIM$^\uparrow$ | LPIPS$^\downarrow$ | PSNR(90%)$^\uparrow$ |
|---|---|---|---|---|
| INSTA | 25.92 | 0.9122 | 0.1195 | 19.89 |
| GHA | 30.90 | 0.9559 | 0.0870 | 26.45 |
| GA | 45.41 | 0.9912 | 0.0589 | 41.02 |
| *GauFace* | **45.67** | **0.9931** | **0.0415** | **41.78** |

Table 4. **Visual quality of *GauFace* and its ablations.** The complex SH Dynamic Shadow Vector (SH DSV) achieves only a 0.3% improvement in PSNR(90%) at the expense of a 721.6% increase in parameters. Without DSV, *GauFace* cannot deliver high-quality rendering, as evidenced by the PSNR(90%) dropping below 40. The Normal Delta offers essential flexibility, enabling *GauFace* to represent visually correct facial deformations accurately.

| Methods | PSNR$^\uparrow$ | SSIM$^\uparrow$ | LPIPS$^\downarrow$ | PSNR(90%)$^\uparrow$ | Size$^\downarrow$ |
|---|---|---|---|---|---|
| *GauFace* | 45.67 | **0.9931** | 0.0415 | 41.78 | 106 |
| w/ SH DSV | **45.72** | **0.9931** | **0.0405** | **41.93** | 871 |
| w/o DSV | 44.11 | 0.9925 | 0.0428 | 39.68 | **55** |
| w/o Normal Delta | 39.90 | 0.9893 | 0.0451 | 34.71 | 105 |



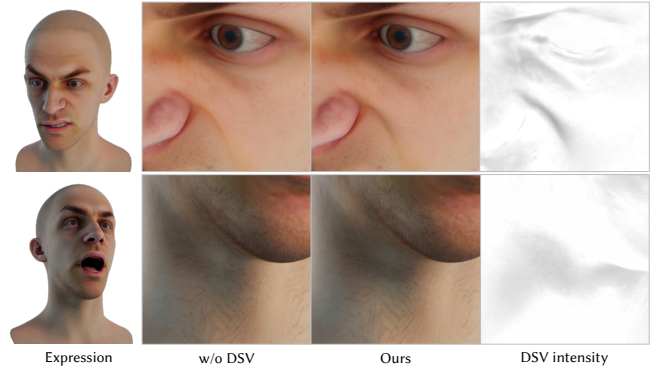Fig. 11. **Evaluation of Dynamic Shadow Vector** *Top*: the change of local self-occlusions around the nose and eye corners are effectively handled. *Bottom*: It helps to express the change of shadows on the neck caused by mouth-open expression.

geometric network. INSTA, lacking a clear geometric representation, fails to decouple angles and expressions, resulting in poorer performance. These results support our argument that existing volume-rendering methods, particularly those with implicit representation-geometry relationships, are not directly useable for mapping PBR facial assets with explicit geometry to volume representations.

*6.2.2 GauFace ablations.*

*Dynamic Shadow Vector.* We disentangle the deformation-dependent and deformation-agnostic shading effects via a per-blendshape Dynamic Shadow Vector (DSV) $l$ for each Gaussian Point. In Fig. 11, we qualitatively illustrate the effect of DSV under two different expressions. DSV effectively handles both local self-occlusion changes and shadow variations caused by deformations of distant facial parts.

We further validate the effectiveness of our DSV design by doing an ablation study under the same setting as in Sec.6.2 without the

Table 5. **Quantitative results of *TransGS* ablations.** Metrics show that the Finetune and UV PE help to increase the quality of *TransGS* generation, while PAS plays a fundamental role in the success of generative modeling.

| Methods | PSNR$^\uparrow$ | SSIM$^\uparrow$ | LPIPS$^\downarrow$ | PSNR(90%)$^\uparrow$ |
|---|---|---|---|---|
| Ours | **38.69** | **0.9971** | **0.033** | **37.66** |
| w/o Finetune | 37.23 | 0.9962 | 0.043 | 35.65 |
| w/o UV PE | 35.06 | 0.9945 | 0.072 | 33.45 |
| w/o PAS | 17.24 | 0.8416 | 0.192 | 14.58 |

DSV. Additionally, we provide a more complex DSV variant called SH DSV. Instead of applying the isotropic shadow factor after SH colors computation, SH DSV applies it for every SH component in an anisotropic way. This leads to an additional 765 optimizable parameters for each Gaussian Point. As shown in Tab. 4, this anisotropic approach achieves only a 0.3% performance gain in PSNR (90%). However, without DSV, the PSNR (90%) drops below 40, indicating a significant degradation in worst-case performance.

*Normal Delta.* We also evaluate a variant where the Gaussian points are strictly locked to the geometry surface, denoted as 'w/o Normal Delta'. As shown quantitatively in Tab. 4, this leads to a significant drop in rendering quality. Gaussian Points need this additional degree of freedom to correctly handle view-dependent and volume effects, including strong directional highlights on the forehead, the shadows inside the mouth and etc.

### 6.3 *TransGS* ablations

Here, we evaluate the key design choices related to *TransGS* on the *Foreface* model. For each ablation, we compute the PSNR, SSIM, LPIPS, and PSNR (90%) metrics of the rendered front images between synthesized *Foreface* Gaussian asset and the test ground-truth *GauFace* asset, with the same train-test split as described in Sec. 6.

*Pixel Aligned Sampling.* Pixel Aligned Sampling (PAS) provides a strong regularization to the sampling positions of Gaussian points, which is crucial for generative modeling. For ablation, we prepare the same *GauFace* dataset without applying the PAS, i.e., letting the center of Gaussian points $\mu$ update during optimizations and enable densification and pruning. We train the same *Foreface* model on this PAS disabled dataset. We disable the Positional Encoding since the sampling positions of Gaussian points are not available.

As shown in Table 5, without PAS our network fails to model the distribution of Gaussian attributes. The visualization in Fig. 12 illustrates this failure qualitatively. This ablation verifies our hypothesis that the vanilla 3DGS representation is not suitable for generative modeling, and our proposed PAS strategy effectively bridges this gap.

*Positional Encoding.* Our UV Positional Encoding (UVPE) helps *TransGS* to extract high-frequency details of Image conditions to the generated *GauFace* asset. Table. 5 indicates an overall degradation of visual quality. As shown in Fig. 12, the model fails to capture high-frequency details like melanoma without UVPE.

*Image Loss Finetuning.* In this ablation, after 800 epochs of training in the main stage, we train the model with another 100 epochs while disabling the image loss. As shown in Tab. 5 and Fig. 12, the
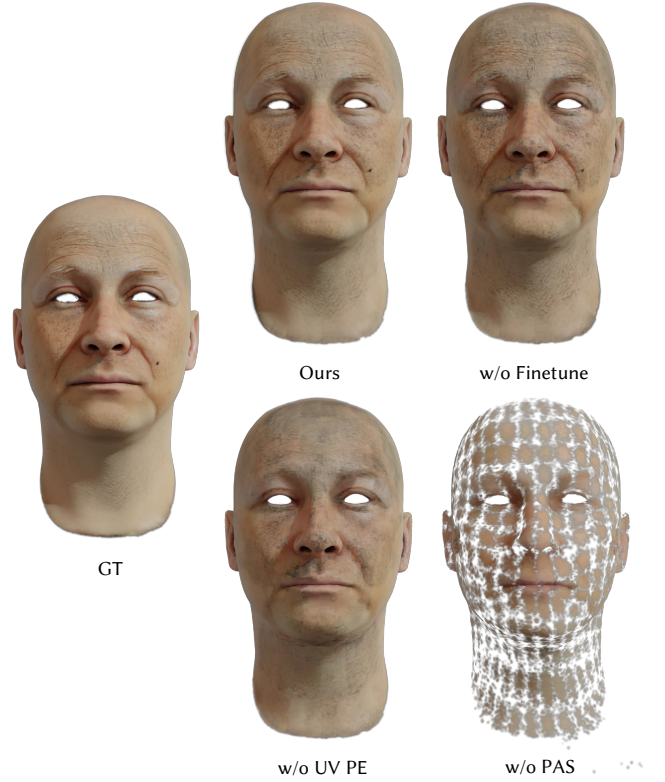


Fig. 12. **Qualitative results of *TransGS* ablations.** We compare the frontal rendering of synthesized *Foreface* assets. *TransGS* without PAS fails to learn a meaningful distribution of Gaussian points. UV PE helps to capture high-frequency details like melanoma. The Finetune stage provides additional rendering quality.

fine-tuning stage slightly increases the visual quality of rendered images.

## 7 APPLICATIONS

Our *GauFace* and *TransGS* pipeline empower a wide range of applications (Fig. 13). On the one hand, *TransGS* integrates PBR facial assets from various sources with Gaussian representations. As shown in Fig. 7, whether the assets are generated, scanned, or downloaded from the internet, *TransGS* can transform them into high-quality Gaussian assets under specific lighting conditions. On the other hand, *GauFace*'s explicit association with geometry enables support for various driving forms in traditional CG animation, such as ARKit facial capture, physics simulation, and audio-driven methods. Thanks to the explicit and efficient *GauFace* representation, unprecedented rendering quality with real-time animation and interaction can be delivered to mobile platforms. Please refer to our supplementary video for various application showcases.

### 7.1 Cross-Platform Real-time Rendering

We implement a *GauFace* renderer using Unity3D's built-in render pipeline. The explicit representation of *GauFace* allows us to leverage
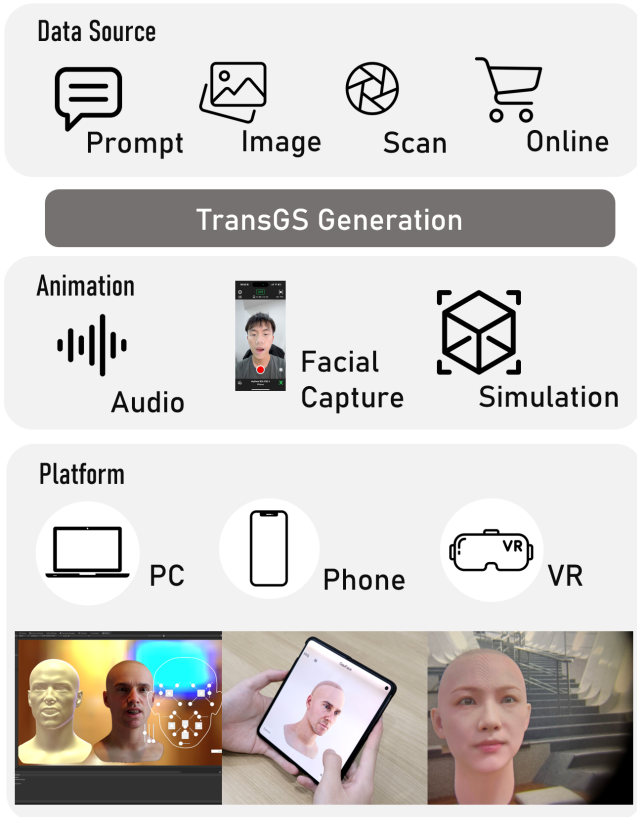
Fig. 13. **Application scenarios.** *Top: TransGS* accepts PBR facial asset conditions from diverse data sources, including prompt or image generated, scanned, online downloaded assets. *Middle: TransGS* generated *GauFace* asset can be controlled via various inputs, e.g., audio-driven signal, ARKit facial capture, and physical simulation, with cross-platform real-time rendering performance (*Bottom*).
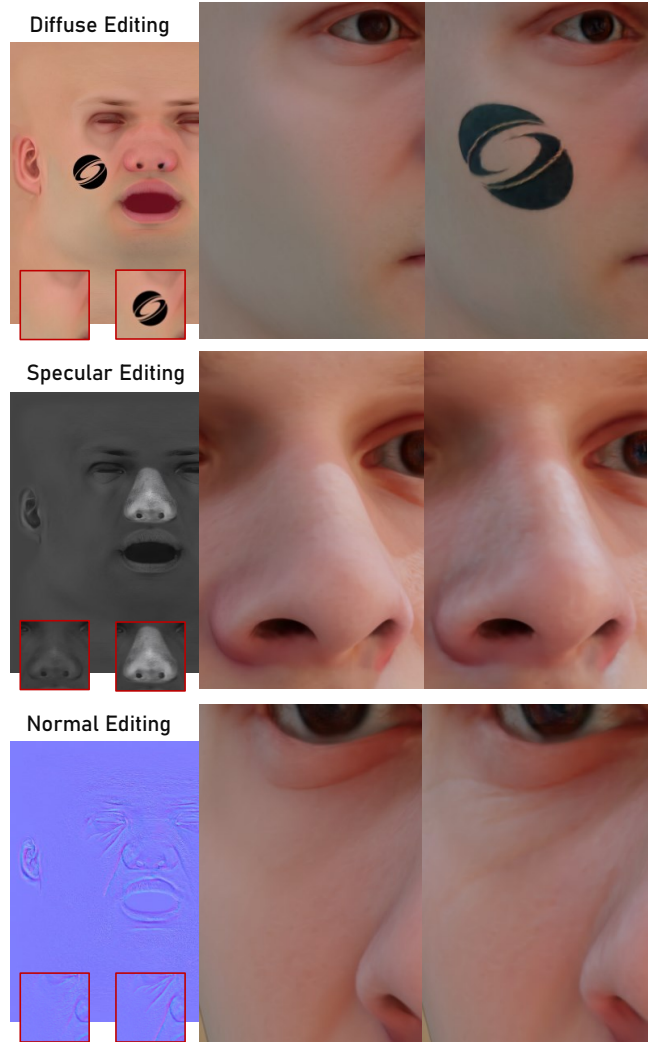


Fig. 14. **Various patch-wise editing.** From left to right: texture editing; rendering of the original *GauFace* asset; rendering of the edited *GauFace* asset. All *GauFace* assets are synthesized by *TransGS* . We can modify the diffuse (top), normal (middle), and specular (bottom) maps, and *TransGS* can faithfully transfer these details to the *GauFace* asset, with natural shading under the conditioned lighting.

Unity's multi-platform compilation support, enabling high-quality *GauFace* rendering and animation effects on platforms supporting DirectX 12 and the Vulkan Graphics API, including Windows Standalone, Android, and VR headsets. In a basic scene with a physical camera, a well-designed skybox, and a *GauFace* object containing 110K Gaussian points, we achieved 500+ fps on an NVIDIA RTX 4080 GPU during animations. On the Android platform, we reached 30 fps on a *Snapdragon*® *8 Gen 2* mobile platform at 1440p resolution. Similar performance is delivered to a Meta Quest Pro VR headset with on-chip computation. Please refer to our supplementary video for the live demo.

### 7.2 Interactive *GauFace* Editing

Since *TransGS* is conditioned on traditional assets, any modifications made to the traditional assets can also be transferred to *GauFace* through the model. Furthermore, because our generative pipeline is patch-based, the network only needs to handle local modification information, making the entire updating process compact and efficient.

We showcase in Fig. 14 three editing scenarios. In the first row of the image, we overlaid a previously unseen logo onto the diffuse map. As shown in the rightmost render, the logo has been successfully transferred to the Gaussian asset, maintaining high clarity and sharp edges while seamlessly integrating into the lighting environment. In the second row, we brightened the specular map for the character's nose. Compared to the previous more diffuse rendering (middle), the enhanced nose (right) now exhibits more pronounced highlights. In the third row, we modified the normal map to add new wrinkles around the character's left eye and nose. The modified Gaussian asset is able to render these wrinkles with a high degree of realism and natural detail, showcasing a rich and detailed appearance.

# 8 CONCLUSION

We have presented *TransGS*, a novel method for translating PBR facial assets into 3D Gaussian Splatting representations instantaneously, supporting relighting and high-quality real-time interaction across different rendering platforms. To achieve this, we proposed *GauFace*, a novel Gaussian representation for efficient rendering of facial interaction. Leveraging strong geometric priors and constrained optimization, *GauFace* ensures neat and rigid representation, bridging the gap between PBR facial assets and Gaussian representation and between Gaussian representation and generative modeling. We then designed a diffusion transformer that translates PBR facial assets to *TransGS* assets. We proposed a novel pixel-aligned sampling scheme and UV positional encoding to ensure the throughput and rendering quality of *TransGS* generation. Through *TransGS*, PBR facial assets can be rapidly converted into efficient Gaussian assets, enabling more realistic real-time facial expression rendering under different lighting conditions across various platforms. This offers new possibilities for immersive interactive experiences, enhanced storytelling and etc. Additionally, *TransGS* endows Gaussian assets with the capability to be modified and driven by traditional CG pipelines, greatly expanding their application scope.

*Limitations.* As an initial attempt, our method has several limitations. For instance, we did not perform Gaussian Splatting modeling and generation for hair, which is a complex task beyond our current scope. Our relighting capability, driven by *TransGS*, while fast, cannot be applied in real-time applications. Therefore, exploring ways to enhance our *GauFace* representation for real-time relighting is a promising direction. Although our approach leverages PBR facial assets and uses Blender Cycles, a physically based rendering engine, to provide ground-truth images, the rendering of our generated *TransGS* assets is not physically accurate because we do not explicitly encode physically correct shading or apply any physical constraints to the generator. Additionally, we model different facial subparts separately, neglecting the correlations of shading between components. Lastly, the rendering quality of our *TransGS* generated assets is constrained by our customized Blender shader; improving this shader could potentially enhance the visual quality of *TransGS* synthesized assets.

*Potential ethical implications. TransGS*'s ability to create highly realistic facial Gaussian avatars raises concerns about consent and control over one's digital representation. Individuals may not have control over how their digital likeness is used, leading to potential harm if their likeness is used without permission or in ways that are harmful or misleading. Additionally, there may be societal implications related to the uncanny valley effect and its impact on human perception and interaction. Highly realistic facial avatars may blur the lines between virtual and real identities, potentially affecting social dynamics and human relationships. It is important for future work to consider the privacy protection and intellectual property control capabilities of *GauFace* assets and the personal data required for generating *GauFace* assets should only be used with explicit permission and authorization to avoid infringement of personal privacy rights.

# 9 ACKNOWLEDGEMENTS

# REFERENCES

Oleg Alexander, Mike Rogers, William Lambeth, Jen-Yuan Chiang, Wan-Chun Ma, Chuan-Chang Wang, and Paul Debevec. 2010. The Digital Emily Project: Achieving a Photorealistic Digital Actor. *IEEE Computer Graphics and Applications* 30, 4 (2010), 20–31. https://doi.org/10.1109/MCG.2010.65

Shivangi Aneja, Justus Thies, Angela Dai, and Matthias Nießner. 2023. Clipface: Text-guided editing of textured 3d morphable models. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–11.

ShahRukh Athar, Zexiang Xu, Kalyan Sunkavalli, Eli Shechtman, and Zhixin Shu. 2022. Rignerf: Fully controllable neural 3d portraits. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*. 20364–20373.

Autodesk. 2024. Arnold. Software. https://www.arnoldrenderer.com/ Version 7.2.

Volker Blanz and Thomas Vetter. 2023. A morphable model for the synthesis of 3D faces. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 157–164.

Blender Foundation. 2024. Blender. Software. https://www.blender.org/ Version 4.1.

George Borshukov and John P Lewis. 2005. Realistic human face rendering for" The Matrix Reloaded". In *ACM Siggraph 2005 Courses*. 13–es.

Chen Cao, Tomas Simon, Jin Kyu Kim, Gabriel Schwartz, Michael Zollhoefer, Shun-Suke Saito, Stephen Lombardi, Shih-En Wei, Danielle Belko, Shoou-I Yu, et al. 2022. Authentic volumetric avatars from a phone scan. (2022).

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. 2022. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 16123–16133.

Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. 2021. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5799–5809.

Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. 2024. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yufan Chen, Lizhen Wang, Qijing Li, Hongjiang Xiao, Shengping Zhang, Hongxun Yao, and Yebin Liu. 2023. Monogaussianavatar: Monocular gaussian point-based head avatar. *arXiv preprint arXiv:2312.04558* (2023).

Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. 2000. Acquiring the reflectance field of a human face. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 145–156.

Eugene d'Eon and David Luebke. 2007. *Advanced Techniques for Realistic Real-Time Skin Rendering.* NVIDIA Corporation, Chapter 14. https://developer.nvidia.com/gpugems/gpugems3/part-iii-rendering/chapter-14-advanced-techniques-realistic-real-time-skin

Helisa Dhamo, Yinyu Nie, Arthur Moreau, Jifei Song, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. 2023. Headgas: Real-time animatable head avatars via 3d gaussian splatting. *arXiv preprint arXiv:2312.02902* (2023).

Craig Donner, Tim Weyrich, Eugene d'Eon, Ravi Ramamoorthi, and Szymon Rusinkiewicz. 2008. A layered, heterogeneous reflectance model for acquiring and rendering human skin. *ACM transactions on graphics (TOG)* 27, 5 (2008), 1–12.

Jiemin Fang, Junjie Wang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. 2024. GaussianEditor: Editing 3D Gaussians Delicately with Text Instructions.

Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. 2021. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8649–8658.

Xuan Gao, Chenglai Zhong, Jun Xiang, Yang Hong, Yudong Guo, and Juyong Zhang. 2022. Reconstructing personalized semantic facial nerf models from monocular video. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–12.

Philip-William Grassal, Malte Prinzler, Titus Leistner, Carsten Rother, Matthias Nießner, and Justus Thies. 2022. Neural head avatars from monocular rgb videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18653–18664.

Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. 2022. StyleNeRF: A Style-based 3D Aware Generator for High-resolution Image Synthesis. In *International Conference on Learning Representations*. https://openreview.net/forum?id=iUuzzTMUw9K

Antoine Guédon and Vincent Lepetit. 2024. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. (2024).

Ralf Habel, Per H. Christensen, and Wojciech Jarosz. 2013. Photon beam diffusion: A hybrid Monte Carlo method for subsurface scattering. *Computer Graphics Forum* 32, 4 (July 2013), 27–37.

Pat Hanrahan and Wolfgang Krueger. 2023. Reflection from layered surfaces due to subsurface scattering. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 279–288.

Christophe Hery et al. 2016. Pixar Deep Dive on Subsurface Scattering: SIGGRAPH Preview. fxguide. Available online: https://www.fxguide.com/fxfeatured/pixar-deep-dive-on-sss-siggraph-preview/.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. 2022. Headnerf: A real-time nerf-based parametric head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20374–20384.

Andrew Hou, Michel Sarkis, Ning Bi, Yiying Tong, and Xiaoming Liu. 2022. Face relighting with geometrically consistent shadows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4217–4226.

Andrew Hou, Ze Zhang, Michel Sarkis, Ning Bi, Yiying Tong, and Xiaoming Liu. 2021. Towards high fidelity face relighting with realistic shadows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14719–14728.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery. https://doi.org/10.1145/3641519.3657428

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

Henrik Wann Jensen, Stephen R Marschner, Marc Levoy, and Pat Hanrahan. 2001. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 511–518.

Yuheng Jiang, Zhehao Shen, Penghao Wang, Zhuo Su, Yu Hong, Yingliang Zhang, Jingyi Yu, and Lan Xu. 2024. HiFi4G: High-Fidelity Human Performance Rendering via Compact Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4401–4410.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8110–8119.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics* 42, 4 (2023), 1–14.

Alexandros Lattas, Stylianos Moschoglou, Baris Gecer, Stylianos Ploumpis, Vasileios Triantafyllou, Abhijeet Ghosh, and Stefanos Zafeiriou. 2020. AvatarMe: Realistically Renderable 3D Facial Reconstruction" in-the-wild". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 760–769.

Alexandros Lattas, Stylianos Moschoglou, Stylianos Ploumpis, Baris Gecer, Jiankang Deng, and Stefanos Zafeiriou. 2023. Fitme: Deep photorealistic 3d morphable model avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8629–8640.

Jiaman Li, Zhengfei Kuang, Yajie Zhao, Mingming He, Karl Bladin, and Hao Li. 2020b. Dynamic facial asset and rig generation from a single scan. *ACM Trans. Graph.* 39, 6 (2020), 215–1.

Ruilong Li, Karl Bladin, Yajie Zhao, Chinmay Chinara, Owen Ingraham, Pengda Xiang, Xinglei Ren, Pratusha Prasad, Bipin Kishore, Jun Xing, et al. 2020a. Learning formation of physically-based face attributes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 3410–3419.

Gao Lin, Liu Feng-Lin, Chen Shu-Yu, Jiang Kaiwen, Li Chunpeng, Yukun Lai, and Fu Hongbo. 2023. SketchFaceNeRF: Sketch-based facial generation and editing in neural radiance fields. *ACM Transactions on Graphics* (2023).

Xian Liu, Xiaohang Zhan, Jiaxiang Tang, Ying Shan, Gang Zeng, Dahua Lin, Xihui Liu, and Ziwei Liu. 2024. Humangaussian: Text-driven 3d human generation with gaussian splatting.

Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. 2021. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (ToG)* 40, 4 (2021), 1–13.

Huiwen Luo, Koki Nagano, Han-Wei Kung, Qingguo Xu, Zejian Wang, Lingyu Wei, Liwen Hu, and Hao Li. 2021. Normalized avatar synthesis using stylegan and perceptual refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11662–11672.

Jiahao Luo, Jing Liu, and James Davis. 2024. SplatFace: Gaussian Splat Face Reconstruction Leveraging an Optimizable Surface. *arXiv preprint arXiv:2403.18784* (2024).

Shugao Ma, Tomas Simon, Jason Saragih, Dawei Wang, Yuecheng Li, Fernando De La Torre, and Yaser Sheikh. 2021. Pixel codec avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 64–73.

Shengjie Ma, Yanlin Weng, Tianjia Shao, and Kun Zhou. 2024. 3D Gaussian Blendshapes for Head Avatar Animation. In *ACM SIGGRAPH Conference Proceedings, Denver, CO, United States, July 28 - August 1, 2024*.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.

Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. 2022. Point-E: A System for Generating 3D Point Clouds from Complex Prompts. (12 2022). https://doi.org/10.48550/arXiv.2212.08751

Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. 2022. Stylesdf: High-resolution 3d-consistent image and geometry generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13503–13513.

Foivos Paraperas Papantoniou, Alexandros Lattas, Stylianos Moschoglou, and Stefanos Zafeiriou. 2023. Relightify: Relightable 3d faces from a single image via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 8806–8817.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3 ed.). Morgan Kaufmann.

Bui Tuong Phong. 1998. Illumination for computer generated pictures. In *Seminal graphics: pioneering efforts that shaped the field*. 95–101.

Puntawat Ponglertnapakorn, Nontawat Tritrong, and Supasorn Suwajanakorn. 2023. DiFaReli: Diffusion face relighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 22646–22657.

Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. 2024. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ravi Ramamoorthi and Pat Hanrahan. 2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 497–500.

Alfredo Rivero, ShahRukh Athar, Zhixin Shu, and Dimitris Samaras. 2024. Rig3DGS: Creating Controllable Portraits from Casual Monocular Videos. arXiv:2402.03723 [cs.CV]

Shunsuke Saito, Gabriel Schwartz, Tomas Simon, Junxuan Li, and Giljoo Nam. 2024. Relightable Gaussian Codec Avatars. In *CVPR*.

Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. 2020. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems* 33 (2020), 20154–20166.

Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. 2024. SplattingAvatar: Realistic Real-Time Human Avatars with Mesh-Embedded Gaussian Splatting. In *Computer Vision and Pattern Recognition (CVPR)*.

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

Jingxiang Sun, Xuan Wang, Yong Zhang, Xiaoyu Li, Qi Zhang, Yebin Liu, and Jue Wang. 2022b. Fenerf: Face editing in neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7672–7682.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. 2024. DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=UyNXMqnN3c

A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, Z. Xu, T. Simon, M. Nießner, E. Tretschk, L. Liu, B. Mildenhall, P. Srinivasan, R. Pandey, S. Orts-Escolano, S. Fanello, M. Guo, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, D. B Goldman, and M. Zollhöfer. 2021. Advances in neural rendering. In *ACM SIGGRAPH 2021 Courses* (Virtual Event, USA) *(SIGGRAPH '21)*. Association for Computing Machinery, New York, NY, USA, Article 1, 320 pages. https://doi.org/10.1145/3450508.3464573

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

Jie Wang, Jiu-Cheng Xie, Xianyan Li, Feng Xu, Chi-Man Pun, and Hao Gao. 2024. GaussianHead: High-fidelity Head Avatars with Learnable Gaussian Derivation. arXiv:2312.01632 [cs.CV]

Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. 2023. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4563–4573.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2023. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. arXiv:2310.08528 [cs.CV]

Jun Xiang, Xuan Gao, Yudong Guo, and Juyong Zhang. 2024. FlashAvatar: High-fidelity Head Avatar with Efficient Gaussian Embedding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yuelang Xu, Benwang Chen, Zhe Li, Hongwen Zhang, Lizhen Wang, Zerong Zheng, and Yebin Liu. 2024. Gaussian Head Avatar: Ultra High-fidelity Head Avatar via Dynamic Gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. GaussianDreamer: Fast Generation from Text to 3D Gaussians by Bridging 2D and 3D Diffusion Models. In *CVPR*.

Dongxu Yue, Qin Guo, Munan Ning, Jiaxi Cui, Yuesheng Zhu, and Li Yuan. 2023. Chatface: Chat-guided real face editing via diffusion latent space manipulation. *arXiv preprint arXiv:2305.14742* (2023).

Hao Zhang, Tianyuan DAI, Yanbo Xu, Yu-Wing Tai, and Chi-Keung Tang. 2024. FaceD-NeRF: Semantics-Driven Face Reconstruction, Prompt Editing and Relighting with Diffusion Models. *Advances in Neural Information Processing Systems* 36 (2024).

Longwen Zhang, Qiwei Qiu, Hongyang Lin, Qixuan Zhang, Cheng Shi, Wei Yang, Ye Shi, Sibei Yang, Lan Xu, and Jingyi Yu. 2023a. DreamFace: Progressive Generation of Animatable 3D Faces under Text Guidance. *ACM Trans. Graph.* 42, 4, Article 138 (jul 2023), 16 pages. https://doi.org/10.1145/3592094

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023b. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3836–3847.

Zhongyuan Zhao, Zhenyu Bao, Qing Li, Guoping Qiu, and Kanglin Liu. 2024. PSAvatar: A Point-based Morphable Shape Model for Real-Time Head Avatar Creation with 3D Gaussian Splatting. *arXiv preprint arXiv:2401.12900* (2024).

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

Zhenglin Zhou, Fan Ma, Hehe Fan, and Yi Yang. 2024. HeadStudio: Text to Animatable Head Avatars with 3D Gaussian Splatting. *arXiv preprint arXiv:2402.06149* (2024).

Yiyu Zhuang, Hao Zhu, Xusen Sun, and Xun Cao. 2022. Mofanerf: Morphable facial neural radiance field. In *European conference on computer vision*. Springer, 268–285.

Wojciech Zielonka, Timo Bolkart, and Justus Thies. 2023. Instant volumetric head avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4574–4584.