# GGHead: Fast and Generalizable 3D Gaussian Heads

TOBIAS KIRSCHSTEIN, Technical University of Munich, Germany
SIMON GIEBENHAIN, Technical University of Munich, Germany
JIAPENG TANG, Technical University of Munich, Germany
MARKOS GEORGOPOULOS, Independent Researcher, Switzerland
MATTHIAS NIESSNER, Technical University of Munich, Germany

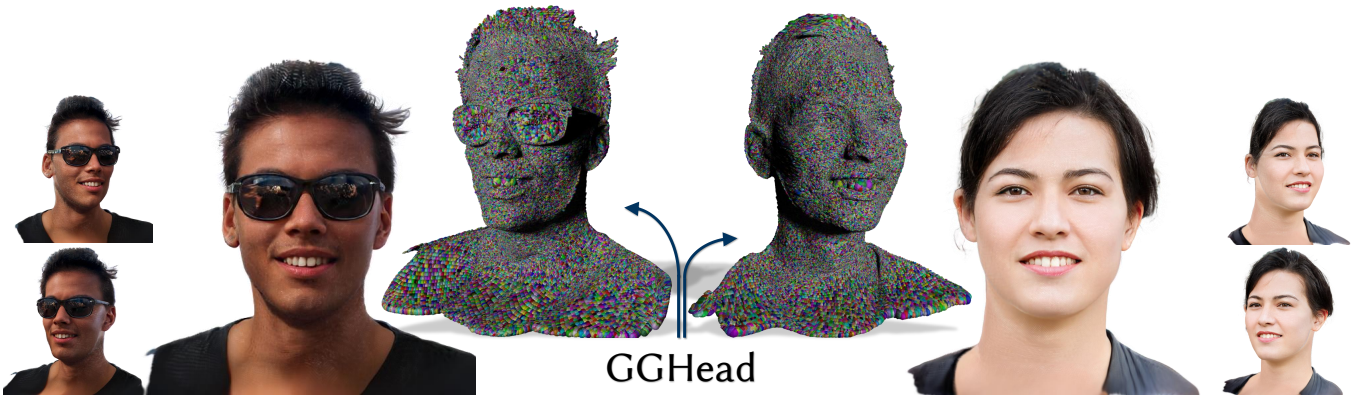



Fig. 1. **GGHead:** Our method can generate diverse 3D head representations based on 3D Gaussian Splatting. The sampled persons exhibit detailed geometry and appearance. Both generation and rendering can be conducted in real-time at full image resolution without the need for 2D super-resolution networks.

Learning 3D head priors from large 2D image collections is an important step towards high-quality 3D-aware human modeling. A core requirement is an efficient architecture that scales well to large-scale datasets and large image resolutions. Unfortunately, existing 3D GANs struggle to scale to generating samples at high resolutions due to their relatively slow train and render speeds, and typically have to rely on 2D superresolution networks at the expense of global 3D consistency. To address these challenges, we propose Generative Gaussian Heads (GGHead), which adopts the recent 3D Gaussian Splatting representation within a 3D GAN framework. To generate a 3D representation, we employ a powerful 2D CNN generator to predict Gaussian attributes in the UV space of a template head mesh. This way, GGHead exploits the regularity of the template's UV layout, substantially facilitating the challenging task of predicting an unstructured set of 3D Gaussians. We further improve the geometric fidelity of the generated 3D representations with a novel total variation loss on rendered UV coordinates. Intuitively, this regularization encourages that neighboring rendered pixels should stem from neighboring Gaussians in the template's UV space. Taken together, our pipeline can efficiently generate 3D heads trained only from single-view 2D image observations. Our proposed framework matches the quality of existing 3D head GANs on FFHQ while being both substantially faster and fully 3D consistent. As a result, we demonstrate real-time generation and rendering of high-quality 3D-consistent heads at $1024^2$ resolution for the first time.

Project Website: https://tobias-kirschstein.github.io/gghead

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*SA Conference Papers '24, December 3–6, 2024, Tokyo, Japan*
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1131-2/24/12.
https://doi.org/10.1145/3680528.3687686

CCS Concepts: • **Computing methodologies** → **Reconstruction**; **3D imaging**; **Adversarial learning**.

Additional Key Words and Phrases: 3D GAN, 3D head prior, 3D Gaussian Splatting

**ACM Reference Format:**
Tobias Kirschstein, Simon Giebenhain, Jiapeng Tang, Markos Georgopoulos, and Matthias Nießner. 2024. GGHead: Fast and Generalizable 3D Gaussian Heads. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–6, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3680528.3687686

## 1 INTRODUCTION

A high-quality 3D generative model that can sample human heads with diverse geometry and appearance has to satisfy two important constraints: A high rendering resolution and strict 3D consistency of generated imagery. Once met, these properties enable exciting use-cases such as single-image-to-3D reconstruction, 3D content creation, or 3D-consistent editing of images.

Unfortunately, obtaining such a high-quality 3D prior over human appearances is extremely challenging. Not only does such a model have to explain all diverse human appearances, but there is also a serious lack of large-scale 3D human datasets due to expensive data capture procedures, rendering any attempt to learn 3D human priors from 3D datasets alone impractical. Hence, methods need to be able to learn 3D appearance and geometry from large 2D image collections, requiring them to render during training. As a result, to meet the criteria for quality and 3D-consistency, the employed rendering procedure has to be scalable, both in terms of speed and rendering resolution.

A well-established approach are 3D Generative Adversarial Networks (3D-GANs) that can learn unconditional 3D generation models from 2D images with corresponding camera poses. These methods have shown remarkable 3D-aware image synthesis capabilities, rapidly closing the quality gap to methods that directly generate 2D face images without view control. Under the hood, 3D GANs combine a differentiable rendering pipeline with a view-conditioned discriminator architecture to supervise 3D generation solely with 2D images. To obtain high-quality 3D samples with this approach, a high rendering resolution is crucial. A commonly chosen and successful differentiable rendering pipeline is Neural Radiance Fields (NeRFs) [Mildenhall et al. 2021]. However, so far, scaling the rendering resolution of NeRFs has been bottle-necked by the expensive ray marching procedure necessary to synthesize the final image. Several works have attempted to alleviate this limitation: EG3D [Chan et al. 2022] renders images at lower resolutions and then upsamples them using 2D superresolution, sacrificing 3D consistency in the process. GRAM [Deng et al. 2022] and GRAM-HD [Xiang et al. 2023] speed up ray marching by simplifying the 3D representation to a set of 2D manifolds, allowing cheap ray intersection tests. However, the simplistic 3D representation worsens rendering quality especially for side-views. Mimic3D [Chen et al. 2023] generates a pseudo-GT dataset from EG3D to supervise the 3D representation directly with patch-based rendering. Due to the inherent view-inconsistencies in the generated multi-view images from EG3D, only a perceptual loss can be used for supervision to gloss over the misalignment. Most recently, Trevithick et al. [2024a] investigate how ray marching can be sped up with proposal networks, allowing training at $512^2$ resolution. Nevertheless, rendering speeds during both training and inference remain a significant challenge. For instance, currently, none of the existing methods are capable of natively generating and rendering images at full $512^2$ resolution in real-time. Since the creation of a 3D GAN involves several million render passes, this imposes a considerable computational burden on training.

To address these scalability challenges, we propose to replace the commonly used NeRF 3D representation with the recent 3D Gaussian Splatting (3DGS) pipeline [Kerbl et al. 2023]. Based on rasterization instead of expensive ray marching, 3DGS has shown remarkable rendering speeds harboring great potential for more scalable 3D GAN pipelines. To generate a set of 3D Gaussian primitives, we exploit the UV space of a template mesh, allowing us to leverage powerful 2D CNN architectures such as StyleGAN2 [Karras et al. 2020b]. From the predicted 2D maps for position, scale, rotation, color, and opacity, we sample 3D Gaussians and place them in the 3D scene relative to the template. Finally, the 3D representation is rendered and supervised with a view-point aware discriminator. The purpose of the template mesh is two-fold: (i) Simplify the generation process by predicting structured 2D UV maps instead of unstructured 3D primitives, (ii) Regularize the predicted 3D representation. The latter is necessary because 3D Gaussians, as opposed to radiance fields, are highly sensitive to just a few parameters. Therefore, bad gradients during the early stages of adversarial training can cause blow-up. To counteract this, we propose a novel UV total variation loss that makes the 3D representation more well-behaved by enforcing continuity of rendered pixels with respect to their UV coordinates. Our final model can learn 3D head geometry and appearance priors from 2D data collections alone. Furthermore, GGHead generates and renders photorealistic 3D heads in real-time thanks to its efficient 3D representation, allowing it to scale to larger resolutions.

In summary, our contributions are as follows:

- We propose GGHead, a 3D GAN for human heads based on 3D Gaussian Splatting that is trained only from 2D image observations.
- We parameterize 3D Gaussian Heads as UV maps that can be generated with efficient 2D CNNs in conjunction with a novel UV total variation regularization that improves geometric fidelity.
- Our approach is highly scalable, facilitating training at 1k resolutions while enabling both real-time sample generation and head rendering.

## 2 RELATED WORK

In this section, we discuss prior related work in the field of generative modeling of 3D heads. We begin by reviewing existing 3D GANs and their application to 3D heads. We then discuss 3DGS-based models and their extension to generalizable representations.

### 2.1 3D Generative adversarial models

Following the success of their 2D counterpart, 3D GANs employ adversarial training to obtain a generative model of 3D representations from 2D images. Earlier works utilise implicit 3D representations and directly render pixels [Chan et al. 2021; Schwarz et al. 2020] or features that are decoded with a CNN-based neural renderer [Niemeyer and Geiger 2021; Xue et al. 2022]. The latter approach has also been adopted by efforts that aim to repurpose successful 2D GAN architectures (e.g., [Karras et al. 2020a, 2019a, 2020b]) and lift the representation to 3D using implicit neural representations [Chan et al. 2022; Gu et al. 2021; Or-El et al. 2022]. Due to their success, these approaches have been extended to introduce different modalities of control, e.g., semantics [Sun et al. 2022].

To obtain higher resolutions, some methods such as EG3D [Chan et al. 2022] apply a super-resolution network to the rendered features. While efficient, this approach can introduce unwanted artefacts in the form of 3D inconsistencies as well as low-resolution geometry. To alleviate this issue, several works focus on rendering in the pixel-space while maintaining fidelity and efficiency. Among them, Skorokhodov et al. [2022] propose to render patches instead of full images to reduce the computational cost. While this allows supervising a generative 3D representation without super-resolution, it impacts global consistency due to the discriminator's limited field of view. Mimic3D [Chen et al. 2023] proposes an imitation strategy that forces the 3D representations to mimic the result of the 2D super-resolution branch. Similarly, GRAM-HD [Xiang et al. 2023] utilizes radiance manifolds and performs super-resolution on 2D manifolds instead of the rendered images. More recently, Trevithick et al. [2024b] introduce a learning-based sampling strategy to effectively render every pixel and to model high-resolution geometry. Orthogonally to these work, our method aims to improve rendering speed by adopting 3D Gaussian Splatting as a 3D representation instead of NeRFs.

### 2.2 Generalizable 3D Gaussian Splatting

To mitigate the computational overhead of volumetric rendering imposed by implicit 3D representations, Kerbl et al. [2023] introduced the seminal work of 3DGS. In this work, a set of Gaussians is optimized from multi-view images using volume splatting [Zwicker et al. 2001]. The means of the Gaussians are initialized using points from SfM [Schonberger and Frahm 2016] while subsequent efforts try to break this dependency [Fu et al. 2023]. Inspired by its efficient rendering, a plethora of methods have been proposed to extend 3DGS for real-time human avatar modeling utilising 3DMMs [Hu and Liu 2023; Jiang et al. 2024; Moreau et al. 2023; Qian et al. 2023a,b; Saito et al. 2023; Svitov et al. 2024; Xu et al. 2023; Zielonka et al. 2023]. Although these efforts focus on modeling a single scene, 3DGS has been extended to generalizable methods for scene reconstruction [Xu et al. 2024; Zou et al. 2023], as well as text-driven synthesis using Score Distillation Sampling [Li et al. 2023; Ling et al. 2023; Ren et al. 2023; Tang et al. 2023; Yi et al. 2023]. Another line of work investigates the use of 3DGS for generalizable few-shot 3D reconstruction [Charatan et al. 2023; Wewer et al. 2024]. However, none of these works focus on learning an unconditional 3D generative model (e.g., a GAN) directly on 3DGS-based representations. Closer to our work, Gaussian Shell Maps (GSM) [Abdal et al. 2023] introduces a Gaussian-based 3D GAN for human bodies by relying on shellmaps of fixed Gaussians. In contrast, we propose a 3D GAN for human heads without the use of shell maps or an animatable mesh prior, and instead with freely moving Gaussians and a novel regularization scheme that exploits the UV space of a template mesh.

## 3 METHOD

Figure 2 shows an overview of our proposed pipeline. We pursue a 3D GAN approach, consisting of a generator that predicts 3D representations, a differentiable rasterizer, and a discriminator that supervises the image formation process. To make the setup scalable, we employ 3D Gaussian Splatting as an underlying 3D representation. In the following, we will discuss how we use a template mesh in conjunction with a 2D CNN to predict 3D Gaussian heads (section 3.2), and how we regularize the 3D representation to ensure stable training in an adversarial setting (sections 3.3 and 3.4).

### 3.1 3D Gaussian Splatting (3DGS)

3D Gaussian Splatting [Kerbl et al. 2023] is a point-based scene representation that assigns each point five different attributes: The Gaussian center $\mu \in \mathbb{R}^3$, scale $\mathbf{s} \in \mathbb{R}^3$, rotation parameterized as quaternion $\mathbf{q} \in \mathbb{R}^4$, color $\mathbf{c} \in \mathbb{R}^3$, and opacity $\sigma \in \mathbb{R}$.

$$\mathcal{G}^i = \{\mu, \mathbf{s}, \mathbf{q}, \mathbf{c}, \sigma\} \tag{1}$$

The resulting annotated point cloud can then be efficiently rendered into an image $I$ using the differentiable tile-based rasterizer $\mathcal{R}$ and camera parameters $\pi$:

$$I = \mathcal{R}(\mathcal{G}, \pi) \tag{2}$$

In the following, we will discuss how one can efficiently generate a 3D Gaussian pointcloud $\mathcal{G}$ using 2D CNN architectures.

### 3.2 Template-based 3D Gaussian Generation

3D Gaussians, being a point-based representation, are inherently unstructured. This poses a significant challenge for generation tasks due to issues such as order ambiguity or large regions of empty space. We therefore follow Gaussian Shell Maps [Abdal et al. 2023] and Relightable Gaussian Codec Avatars [Saito et al. 2023], and rig the 3D Gaussian representation to a template mesh with corresponding UV layout. This enables the use of an efficient 2D CNN backbone $\mathcal{B}$ to predict the Gaussian representation $\mathcal{G}$, considerably simplifying the generator's task. Formally, we generate one UV map $M$ for each Gaussian attribute:

$$M_\star = \mathcal{B}(z, \pi) \tag{3}$$

where $\star \in \{\text{POSITION, SCALE, ROTATION, COLOR, OPACITY}\}$ and $\mathcal{B}$ is a StyleGAN2 [Karras et al. 2020b] generator, mapping a normally distributed latent code $z \in \mathbb{R}^{512}$ to UV maps $M \in \mathbb{R}^{256\times256\times14}$.

For predicted positions, we add another learnable CNN layer $\mathcal{Z}$ with weights initialized as zeros following Zhang et al. [2023]:

$$M_{position} \leftarrow \mathcal{Z}(M_{position}) \tag{4}$$

This ensures that predicted position offsets are 0 at the beginning of training, causing Gaussians to be predicted on the template mesh. This is part of a set of measures that improve training stability as further elaborated in sections 3.3 and 3.4.

To obtain the actual 3D Gaussian primitives, we uniformly sample the predicted 2D maps $M$. Each Gaussian $\mathcal{G}^i$ has a fixed coordinate $x^i_{uv} \in [0, 1]^2$ in the template's UV space and will query its attributes from $M$:

$$\mathcal{G} = \textsc{GridSample}(M, x_{uv})$$

where the $\textsc{GridSample}(\cdot)$ operater performs lookup in the discrete maps $M$ via bilinear interpolation. The sampling scheme governs how many Gaussians will be created and is shared across all generated persons. Note that the generator is mostly agnostic to how many Gaussians are sampled meaning that the sampling scheme may be changed during training, e.g., to increase the number of Gaussians when rendering at higher resolutions.

### 3.3 Gaussian Attribute Modeling

*3.3.1 Gaussian Positions.* To ensure that the predicted Gaussian attributes always stay within a reasonable domain, we employ activation functions. Most notably, for the predicted positions, we add the Gaussian's corresponding 3D position $v^i$ on the template mesh and limit the predicted offset with a TANH activation function:

$$\mathcal{G}^i_{position} \leftarrow v^i + \gamma_{pos}\text{TANH}\left(\mathcal{G}^i_{position}\right) \tag{5}$$

where $\gamma_{pos}$ is the maximum allowed predicted offset from the template mesh. For our experiments on FFHQ, we use $\gamma_{pos} = 0.25$ meaning that Gaussians can move at most 25cm away from the template. This step is crucial, as it prevents the Gaussians from leaving the training view frustums during early stages of adversarial training.

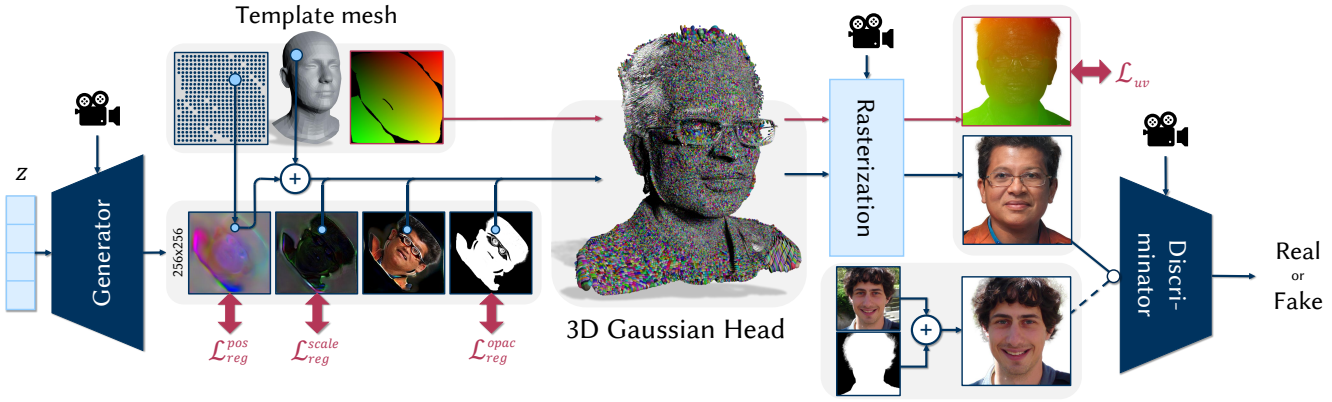



Fig. 2. **Method Overview:** We adopt 3D Gaussian Splatting inside a 3D GAN formulation to learn a distribution of 3D heads solely from 2D images. To build the 3D Gaussian representation, we exploit the UV space of a template mesh. The CNN generator then takes a normally distributed latent code $z$ and predicts 2D maps for each Gaussian attribute. To obtain the actual 3D Gaussian primitives for rasterization, we sample the UV maps uniformly and place the primitives relative to the template mesh. The generated 3D Gaussian representation is then rasterized and supervised by the discriminator. To increase training stability, especially during early stages of adversarial training, we regularize the predicted position offsets, scales and opacities via $\mathcal{L}_{reg}^{pos}$, $\mathcal{L}_{reg}^{scale}$ and $\mathcal{L}_{reg}^{opac}$. Furthermore, we propose a novel UV total variation loss $\mathcal{L}_{uv}$ to improve the geometric fidelity of generated 3D heads by enforcing UV renderings to be smooth.

*3.3.2 Gaussian Scales.* In similar spirit, we limit the range of predicted Gaussian scales:

$$\mathcal{G}_{scale}^{i} \leftarrow \exp\left(-s_{max} - \text{softplus}(-(G_{scale}^{i} - s_{init}) - s_{max})\right) \quad (6)$$

which ensures that scales are within $[0, e^{-s_{max}}]$ and initially default to $e^{-s_{init}}$ when the network predicts zeros. We set $s_{max} = 3$ and $s_{init} = 5$ which initializes Gaussian scales with ~7mm roughly covering the template mesh, and limits them from exceeding ~5cm.

## 3.4 Gaussian Geometry Regularization

Simply predicting 3D Gaussians in an adversarial setting leads to poor results. The reasons for this are two-fold: (i) The 3D Gaussians react very sensitively to gradient updates during early training stages where the discriminator is not calibrated yet. (ii) The 3D Gaussian representation is quite powerful. Therefore, the generator is capable of predicting scenes that look plausible when rendered, but the underlying geometry lacks realism.

We find that a set of regularizations that implement simple intuitions effectively stabilize training as well as improve the underlying geometry. To tackle training stability, we use a simple L2 regularization on the predicted position offsets and Gaussian scales. To improve the geometric fidelity of generated 3D representations, we propose to use a beta regularization on the predicted offsets as well as a novel total variation loss on UV renderings.

*3.4.1 Gaussian Position Regularization.* A common cause of instabilities during training are Gaussians moving around too much. We prevent this behavior with a position regularization on predicted offsets:

$$\mathcal{L}_{reg}^{pos} = \|M_{position}\|_2 \quad (7)$$

This term encourages predicted Gaussians to stay close to the template mesh.

*3.4.2 Gaussian Scale Regularization.* As opposed to implicit scene representations, such as NeRFs, 3D Gaussians are very sensitive to just a few parameters. For example, it is possible for a single Gaussian to grow tremendously in scale, covering the whole view frustum with just one primitive. Such cases are detrimental for training stability. We employ a simple regularization on the predicted Gaussian scales to avoid such degenerate 3D representations:

$$\mathcal{L}_{reg}^{scale} = \|M_{scale}\|_2 \quad (8)$$

In combination with eq. (6) this regularization pushes Gaussian scales to be close to ~7mm which is just large enough that Gaussians would cover the template mesh.

*3.4.3 Gaussian Opacity Regularization.* To improve the geometric fidelity of the generated Gaussian representation, we employ a beta regularization on the opacities, encouraging that Gaussians should be either fully transparent or fully opaque:

$$\mathcal{L}_{reg}^{opac} = \text{Beta}(M_{opacity}) \quad (9)$$

where Beta is the negative log-likelihood of a Beta(0.5, 0.5) distribution [Lombardi et al. 2019].

*3.4.4 UV Total Variation Loss.* A naive implementation of our method generates Gaussian representations with many degrees of freedom. As a result, it can happen during training that the generator "fakes" high-frequency detail by letting colors from further back shine through. This still generates high-quality images but causes noticeable artifacts when rendering videos. As the discriminator is not aware of the time dimension, this behavior is never punished during training. We therefore design a novel regularization scheme to improve the geometric fidelity of generated representations. The intuition behind our regularization is that neighboring pixels in a rendered image should be modelled by Gaussians that are also close in UV space. This naturally leads to smooth surfaces and penalizes

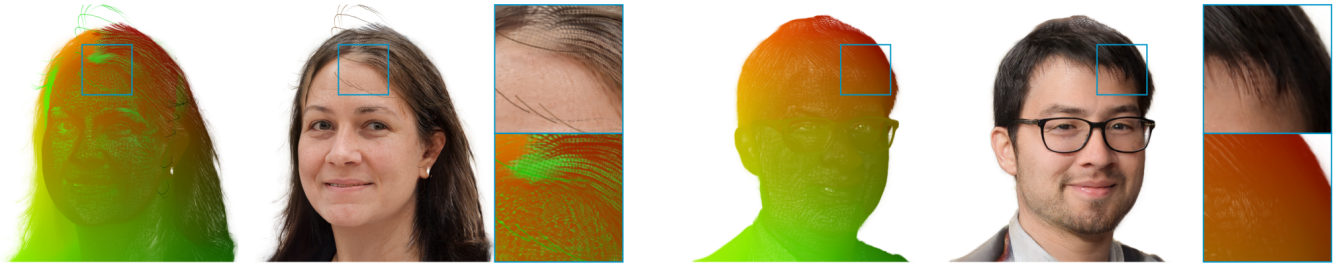



Fig. 3. **Effect of TV UV regularization:** Without $\mathcal{L}_{uv}$, the predicted Gaussian geometry is flawed. Two common failure cases are the emergence of floating lines of Gaussians that are especially visible in video renderings, and improper surfaces where skin texture is created by letting Gaussians from the back of the head shine through. Both cases are easily detectable in the UV renderings $R_{uv}$. Our novel UV total variation loss exploits this fact and effectively addresses both issues by enforcing $R_{uv}$ to be smooth.

holes. Formally, we apply a total variation (TV) regularization on rendered UV images as follows:

$$\hat{\mathcal{G}} \leftarrow \mathcal{G} \tag{10}$$

$$\hat{\mathcal{G}}^i_{color} \leftarrow (u^i, v^i, 0) \tag{11}$$

$$R_{uv} = \mathcal{R}(\hat{\mathcal{G}}, \pi) \tag{12}$$

$$R'_{uv} = \frac{R_{uv} - (1 - R_\alpha)}{R_\alpha} \tag{13}$$

$$\mathcal{L}_{uv} = TV(R'_{uv}) \tag{14}$$

where $u^i, v^i = x^i_{uv}$ are a Gaussian's UV-coordinates, $\pi$ is the camera, and $R_\alpha \in \mathbb{R}^{H \times W}$ is the rendered alpha map containing the per-pixel accumulations from rasterization. We refer to $R_{uv}$ as a *UV rendering*. Equation (13) reverses the alpha compositing that is performed during rasterization and is necessary in order to disregard transparency for the regularization. Otherwise, the border region between foreground and background, which is quite likely to include semi-transparent pixels due to the fuzzy nature of Gaussians, would contribute consistently to the regularization term, incentivizing foreground Gaussians to cover the background. With our novel UV total variation loss in place, holes in the 3D representation are stitched during training, noticeably improving the generated geometry. As a positive side effect, the regularization also removes other types of artifacts such as floating Gaussians that can appear in front of the face. Figure 3 shows how the generated 3D representation is improved by our novel regularization term.

### 3.5 Training Pipeline

We generate UV maps $M$ at $256^2$ resolution and initially sample one Gaussian per UV map texel resulting in $65k$ Gaussians. We begin training at $256 \times 256$ rendering resolution and apply progressive growing after $7000k$ training images. Since the generator is resolution agnostic, we only need additional layers for the discriminator when increasing the rendering resolution. For progressive growing, we add a CNN block in front of the discriminator with a skip connection to ensure a smooth transition and blend in the contributions from the untrained layers over the course of $1000k$ images. We also increase the number of Gaussians when increasing the rendering resolution to facilitate modelling more high-frequency details. To achieve this, we simply increase the sampling density of Gaussians in UV space from $256^2$ to $512^2$ for a total of $262k$ Gaussians. This sudden increase in Gaussians causes the predictions to become temporarily more blurry. Nevertheless, training stability is not adversely affected by that and the generator quickly adapts to the presence of more Gaussians. Also note that the Gaussians are never all active at the same time for any given input latent $z$. The generator uses the opacity attribute to switch Gaussians on and off as needed, avoiding topological issues that would be implied by the template mesh. In practice, around half of the Gaussians are active on average.
The final optimization term for the generator is as follows:

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_p \mathcal{L}^{pos}_{reg} + \lambda_s \mathcal{L}^{scale}_{reg} + \lambda_o \mathcal{L}^{opac}_{reg} + \lambda_{uv} \mathcal{L}_{uv} \tag{15}$$

$$\mathcal{L}_{adv} = \text{SOFTPLUS}(-D(\mathcal{R}(\mathcal{G}), \pi)) \tag{16}$$

where $\mathcal{L}_{adv}$ is the standard non-saturating GAN loss [Goodfellow et al. 2014] applied on the rendered 3D Gaussian representation. The discriminator $D$ is trained with R1 gradient regularization [Mescheder et al. 2018] with weight 1. For the regularization weights, we choose $\lambda_p = 0.1$ and $\lambda_s = 0.05$. The opacity and UV total variation loss are initially turned off for training at $256^2$ resolution and are set to $\lambda_o = 1$ and $\lambda_{uv} = 100$ as soon as progressive growing is applied. We use a batch size of 32 with a learning rate of 0.0025 for the generator and 0.002 for the discriminator discriminator using the Adam optimizer [Kingma and Ba 2014]. We also incorporate the speed improvements by Durvasula et al. [2023] for faster differentiable rasterization of Gaussians. In total, we train our model for 25M images following Chan et al. [2022] which takes roughly 12 days on 2 RTX A6000 GPUs with 48G of VRAM each.

## 4 EXPERIMENTAL RESULTS

### 4.1 Datasets

We train our models on data from the FFHQ dataset [Karras et al. 2019b], which consists of 70k mostly frontal images of human faces. We use the preprocessed version of Chan et al. [2022] who aligned the images via facial landmarks, cropped them into $512^2$ resolution,

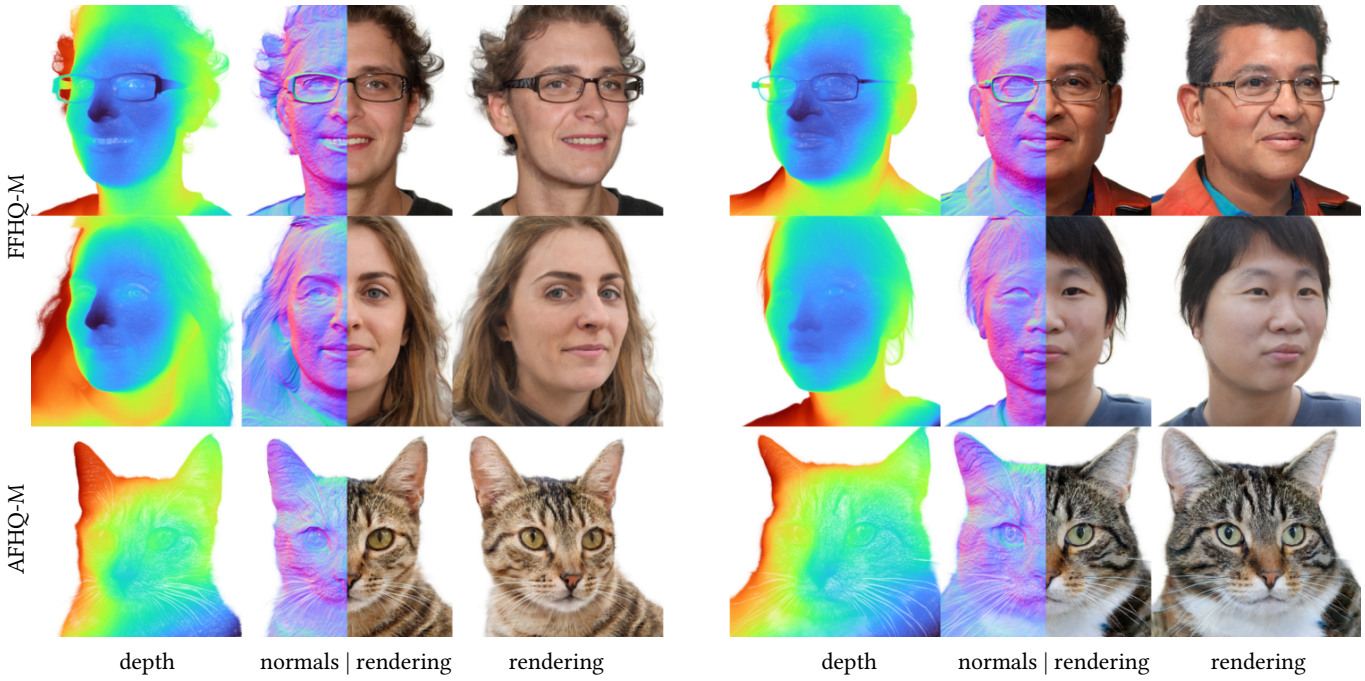



Fig. 4. **Generated Geometry:** We show samples from our models trained on FFHQ-M and AFHQ-M on $512^2$ resolution with corresponding depth and normal maps. GGHead can generate fine geometric details such as eyeglasses and hair strands.

and computed camera poses by resorting to a 3DMM fitting [Deng et al. 2019]. We further follow EG3D's processing pipeline to obtain a $1024^2$ resolution dataset of cropped and aligned face images with corresponding camera poses from the original FFHQ in-the-wild dataset. Finally, we employ the off-the-shelf background matting network MODNet [Ke et al. 2022] to replace the background with white pixels yielding a masked version of FFHQ denoted as FFHQ-M. That way, our models can focus their generation power solely on the actual 3D head. We also apply the same masking strategy to the AFHQv2 Cats dataset [Choi et al. 2020] yielding AFHQ-M.

## 4.2 Baselines

We compare our method to several competitive baseline methods from the 3D GAN literature. The baselines are selected based on three criteria: quality, speed/resolution, and architectural similarity.

*EG3D [Chan et al. 2022]* is a state-of-the-art 3D GAN that synthesizes high-quality face images by relying on a super-resolution network to upsample low-resolution renderings from generated TriPlanes.

*Mimic3D [Chen et al. 2023]* is SoTA among fully 3D-consistent GANs on FFHQ. It distills a TriPlane generator backbone by directly supervising rendered images with pseudo-GT multi-view images obtained from EG3D.

*GRAM-HD [Xiang et al. 2023]* proposed an efficient radiance field representation based on 2D manifolds, enabling faster rendering and training at $1024^2$ resolution.

*Gaussian Shell Maps (GSM) [Abdal et al. 2023]* shares important architectural design decisions with our method, most notably that it also predicts 3D Gaussians in the UV space of a template.

For EG3D and Mimic3D, we use the official implementation and retrain their methods on the masked FFHQ and AFHQ datasets. Since Gaussian Shell Maps was originally proposed for animatable full body synthesis, we adapt their implementation and retrain it on the FFHQ-M dataset. For GRAM-HD, we use official model checkpoints to conduct qualitative and timing comparisons.

Table 1. **Quantitative comparison:** We compare our method to other 3D GANs on FFHQ and AFHQ at $512^2$ resolution. FFHQ-M and AFHQ-M denote the resepective datasets with the background removed. Our method fares favorably against other fully 3D-consistent methods and comes close to the performance of EG3D despite not using any 2D super-resolution. GGHead also produces more consistent 3D surfaces while simultaneously being considerably faster.

| | | FFHQ-M | | | FFHQ | AFHQ-M |
|---|---|---|---|---|---|---|
| | Method | FID↓ | $PSNR_{mv}$↑ | $SSIM_{mv}$↑ | FID↓ | FID↓ |
| w/ SR | EG3D | **3.28** | 28.74 | 0.899 | **4.70** | **2.82** |
| w/o SR | GSM | 28.19 | 26.40 | 0.919 | - | - |
| | Mimic3D | 4.27 | <u>30.95</u> | <u>0.949</u> | 5.37 | 4.48 |
| | Ours | <u>4.06</u> | **33.29** | **0.964** | <u>5.15</u> | <u>3.40</u> |

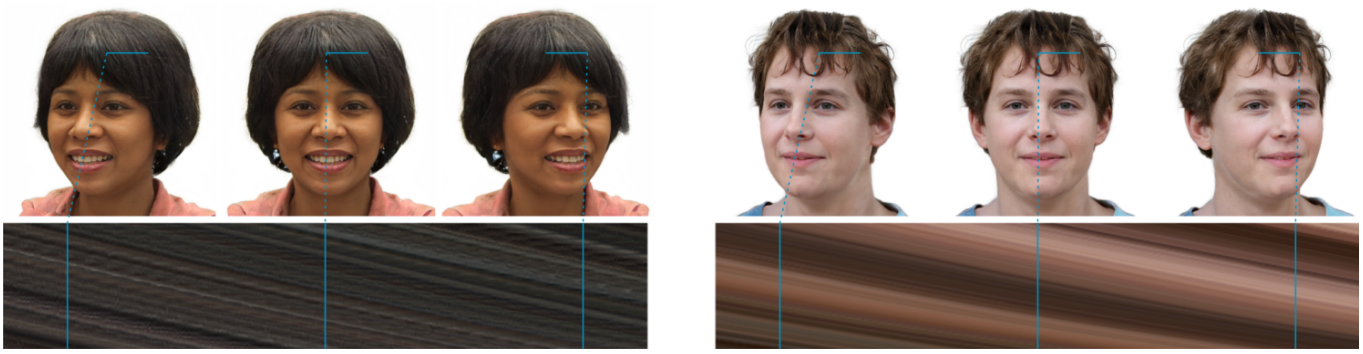

Fig. 5. **Qualitative Comparison:** We show uncurated samples (seeds 0-4) of different 3D GANs trained on FFHQ at $512^2$ resolution. Our method matches the quality of existing approaches while being strictly 3D-consistent and one order of magnitude faster to render. Times measured on an RTX A6000 GPU with a batch size of 1.

### 4.3 Quantitative Comparison

We conduct quantitative comparisons with EG3D, Mimic3D, and Gaussian Shell Maps on the FFHQ and AFHQv2 Cats datasets at $512^2$ resolution. For comparisons on the unmasked FFHQ dataset, we extend our generator to predict 3 additional channels to produce a background image and simply blend that with the rendered Gaussian head. For AFHQ-M, we take a model trained on FFHQ-M and finetune it on the much smaller cats dataset following Chan et al. [2022]. Since there is no direct ground truth to evaluate the quality of generated 3D representations, we resort to measuring the Fréchet Inception Distance (FID) [Heusel et al. 2017]. To do that, we generate 50k samples for each trained model using the training camera distribution and compare its statistics to the full dataset. The results are shown in table 1. Our method matches the performance of current state-of-the-art 3D GANs, obtaining slightly better FID scores than Mimic3D. It is only surpassed by EG3D, which employs a super-resolution module to achieve its rendering quality. To quantify the adverse effects of screen-space super-resolution on 3D consistency, we follow the procedure laid out by GRAM-HD [Xiang et al. 2023]. For each method, we generate 30 images from different viewpoints of the same person, and then train the surface reconstruction method NeuS2 [Wang et al. 2023] on the generated images. We then measure the reconstruction error with PSNR and SSIM scores. The reasoning behind this is that view-inconsistencies in the images given to NeuS2 will lead to a higher reconstruction error. In practice, we observe that this metric also measures "surfaceness" of a 3D representation, since NeuS2 favors smooth and well-defined surfaces by design. We report the multi-view surface reconstruction results averaged over 100 randomly generated heads in table 1. GGHead leads to significantly better surface reconstructions than EG3D, underscoring its great 3D consistency.

### 4.4 Qualitative Comparison

Figure 4 shows samples from our model alongside extracted depth and normal maps for both FFHQ-M and AFHQ-M. The generated 3D representations exhibit a great amount of geometric detail and



EG3D [Chan et al. 2022] Ours

Fig. 6. **Analysis of 3D Consistency:** We show spatio-temporal line textures akin to the Epipolar Line Images (EPI) [Bolles et al. 1987] obtained by rotating the camera horizontally and stacking the pixels of a fixed horizontal line segment. EG3D suffers from texture-sticking artifacts due to its use of a 2D super-resolution network, leading to staircases in the epiploar line images. In contrast, our method provides smooth renderings without any flickering.

GRAM-HD [Xiang et al. 2023] Mimic3D [Chen et al. 2023] Ours

Fig. 7. **Analysis of Side-views:** We show renderings of GRAM-HD, Mimic3D and our model from extreme side-views. GRAM-HD exposes the peculiarities of its underlying 3D representation in this scenario since the predicted 2D manifolds often run in parallel, causing striped artifacts. Mimic3D suffers from blurred colors when viewed from the side. In contrast, the 3D Gaussian representation generated by our method still looks appealing.

diversity. We further compare samples from GGHead to those of EG3D, Mimic3D, GRAM-HD and GSM in fig. 5. Our method produces renderings of great quality matching the current state-of-the-art at much higher speed. Interestingly, GSM synthesizes images with noticeable artifacts, indicating that generating 3D Gaussians alone is not sufficient for high-quality renderings. We attribute this to the design of GSM which disallows Gaussians from moving, severely restricting the expressiveness of the generated 3D representation.

Next, we analyse the view-consistency of generated images from EG3D and our method. EG3D generally can synthesize high quality static images, but this comes at the cost of view-inconsistencies due to the use of a 2D super-resolution network. In fig. 6, we present the spatio-temporal texture images similar to Epipolar Line Images (EPI) [Bolles et al. 1987] for EG3D and our method. Specifically, we smoothly rotate the camera around the generated head while stacking the pixels from a fixed horizontal line segment for each timestep. For multi-view consistent images, the resulting spatio-temporal texture should appear smooth. Conversely, edged artifacts and noise in the spatio-temporal texture indicate 3D inconsistencies. For EG3D, such artifacts are prevalent and manifest as flickering in video rendering. In contrast, our method generates strictly 3D-consistent images.

Furthermore, we investigate the generations of Mimic3D and GRAM-HD in more detail in fig. 7. We find that both methods generally produce high-quality renderings but expose idiosyncracies of the underlying 3D representation when viewed from extreme angles. While GRAM-HD's use of 2D manifolds becomes visible in this scenario, Mimic3D produces slightly fuzzy surfaces, which is typical for NeRFs, causing blurry renderings for side views. Instead, our method produces more well-behaved renderings for such extreme camera angles. We attribute this to the fact that generating 3D Gaussian primitives in UV space forces them to lie on a single 2D manifold. This acts as an inductive bias towards more surface-like 3D representations which helps GGHead to generalize to such challenging scenarios.

### 4.5 Compute Resource Requirements

A core motivation for our method is to lower the time needed for generating and rendering 3D heads, ultimately leading to faster training of 3D GANs at higher resolutions. We benchmark training and inference times of existing 3D GANs, and compare them to our method in table 2. As expected, we observe a huge performance gap between the NeRF-based approaches (EG3D, Mimic3D, GRAM-HD) and methods that employ 3D Gaussian Splatting (GSM, Ours). This is true not only for the actual rendering of the 3D representation itself, but also for the required time for a full training iteration where rendering also has to be performed. Note that due to expensive ray marching, EG3D and Mimic3D have considerably slower training

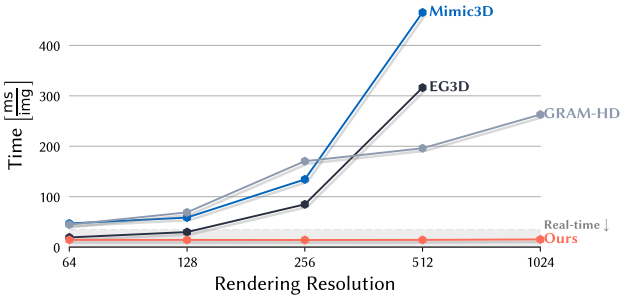



Fig. 8. **Generation and Rendering Resolution Benchmark:** We measure the rendering times (including generation of the 3D representation) on a single RTX A6000 GPU with a batch size of 1. Our method can effortlessly generate high-quality 3D representations and render them at 1k native resolution in real-time while still matching other competitive 3D GANs in terms of FID.

Table 2. **3D GAN performance benchmark:** We measure the time it takes to generate 3D representations (Gen.), render them (Rend.), and perform a full forward-backward pass during training (Forw. + Backw.). "Res." denotes the native rendering resolution *excluding* super-resolution. Note that EG3D and Mimic3D have to lower the native rendering resolution substantially during training. In contrast, our method runs much faster at full resolution without sacrificing rendering quality. Times averaged over 1000 images.

| | Train | | Inference | | |
|---|---|---|---|---|---|
| | Res. | Forw. + Backw.↓ | Res. | Gen.↓ | Rend.↓ |
| EG3D | 128 | 209.61 ms | 128 | **6.1 ms** | 23.4 ms |
| Mimic3D | 64 | 298.73 ms | 512 | 20.3 ms | 286.7 ms |
| GRAM-HD | - | - | 512 | 157.6 ms | 38.2 ms |
| GSM | 512 | **120.16 ms** | 512 | 8.8 ms | 5.2 ms |
| Ours | 512 | 127.78 ms | 512 | 10.4 ms | **3.8 ms** |

iterations *despite* rendering only $\frac{1}{16}$th of the pixels.
We further study the scalability of existing 3D GAN methods by measuring the required time for a full forward pass under different rendering resolutions. The scaling behavior is plotted in fig. 8. Again, our method demonstrates a considerably better scaling behavior on larger rendering resolutions thanks to the efficient 3D Gaussian Splatting rasterizer.
Finally, we record GPU memory requirements and the total training time in table 3. Our method is significantly more memory-efficient than other 3D GANs and finishes training noticeably faster.

## 4.6 Ablations

In the following, we analyze important design decisions of our method, including the choice of template mesh, the number of generated Gaussians, and the effect of our novel UV total variation loss.

*4.6.1 Effect of Regularization.* In the upper part of table 4, we analyze the impact of the scale and position regularization terms $\mathcal{L}_{reg}^{scale}$ and $\mathcal{L}_{reg}^{pos}$, as well as the usefulness of the zero-initialized CNN layer $\mathcal{Z}$ that enforces small position offsets predictions in the beginning. We find that all measures improve training stability which translates to a better FID score. Without scale regularization, predicted Gaussians even disappear entirely leading to total training collapse.

Table 3. **Compute Resource Requirements:** We measure the GPU memory consumption during training on a single RTX A6000 GPU (48GB VRAM). Out-of-memory is denoted as *OOM*. Our method is significantly more memory-efficient and therefore scales better with larger batch sizes. We also record the total training time in equivalent days on a single RTX A6000.

| | Batch Size | | | | |
|---|---|---|---|---|---|
| Method | 4 | 8 | 16 | 32 | Train time |
| Mimic3D | 20.4 GB | 38.7 GB | *OOM* | *OOM* | 80.2 days |
| EG3D | 13.4 GB | 25.1 GB | *OOM* | *OOM* | 39.0 days |
| Ours | **8.6 GB** | **12.8 GB** | **19.4 GB** | **35.0 GB** | **23.4 days** |

Table 4. **Ablation study:** We analyze the effect of regularization, different template meshes, different numbers of Gaussians, and our novel UV total variation loss when training our model on FFHQ-M.

| | | Method | FID↓ |
|---|---|---|---|
| 256 | Regularizations | w/o $\mathcal{L}_{reg}^{scale}$ | collapse |
| | | w/o $\mathcal{L}_{reg}^{pos}$ | 10.92 |
| | | w/o zero-init layer $\mathcal{Z}$ | 6.71 |
| | Template | Sphere Template | 4.59 |
| | | Plane Template | 4.27 |
| | | FLAME Template | 4.74 |
| | | Adjusted FLAME Template | **4.03** |
| 512 | Gaussians | 65k Gaussians | 6.39 |
| | | 262k Gaussians | 4.28 |
| | | + $\mathcal{L}_{uv}$ (Ours) | **4.06** |

*4.6.2 Effect of Template Mesh.* In table 4, we compare multiple versions of GGHead trained with different template meshes on the masked FFHQ dataset at $256^2$ resolution. We consider different types of template meshes, ranging from simple domain-agnostic shapes, like planes and spheres, to meshes that are designed for the face domain. Here, we compare two different head meshes: FLAME [Li et al. 2017] and an adapted version of the FLAME mesh taken from Abdal et al. [2023] which has a more efficient UV layout and the back of the head removed. It turns out that GGHead is fairly robust to the exact choice of template mesh since all templates produce reasonable results in terms of FID, with the adjusted FLAME mesh scoring best. We hypothesize that the efficiency of the UV layout and the density of allocated points in the frontal face region are more important than the actual shape of the template mesh.

*4.6.3 Effect of UV Total Variation Loss.* Being fundamentally a 3D point representation, the 3D Gaussians lack the notion of a proper surface. Consequently, it can happen during training that Gaussians in the face area become too small, allowing other Gaussians from

further back to shine through. This can create high frequency detail that satisfies the discriminator loss and looks plausible in static renderings. However, such a 3D representation remains flawed and is easily exposed in video renderings. Since we cannot supervise the time dimension during training, however, we instead employ our novel UV total variation loss. The effect can be seen in fig. 3. The regularization term effectively reduces the amount of Gaussians that shine through, considerably improving the surface quality of the generated 3D representations. As a side effect, the loss also resolves another failure case where lines of Gaussians would appear that mimic the appearance of hair strands but occur out of place. In table 4, we also study the effect of the UV total variation loss quantitatively and find that it slightly improves FID.

*4.6.4 Effect of Number of Gaussians.* Due to our generator design, we can flexibly sample different numbers of Gaussians from the same model. We can use this to our advantage when progressively growing to larger resolutions where more high-frequency detail needs to be synthesized. As opposed to directly training with all Gaussians from the beginning, this has two advantages: Training is both faster and more stable due to fewer degrees of freedom. We analyze the effect of more Gaussians on the final results in table 4 where increasing the number of Gaussians (262k) gives a noticeable improvement in quality over keeping the same number of Gaussians (65k) as in lower resolution training.

## 5 LIMITATIONS AND FUTURE WORK

We have demonstrated that GGHead can generate and render 3D heads at high quality. However, the generated 3D representations only provide a user with viewpoint control. Having additional control over the facial expressions would enable further use-cases such as 3D-consistent expression editing of single images. Since our method already employs a coarse template mesh, it stands to reason to extend it in similar fashion as Abdal et al. [2023]; Qian et al. [2023a]; Sun et al. [2023], e.g., by employing FLAME and training on FFHQ with corresponding expression codes. One step further, the improved scalability of our method may enable training on large facial video datasets [Zhu et al. 2022], ultimately paving the way for building a photorealistic 3DMM from 2D data alone.

Another axis of improvement is generality. We have shown how a 3D generative model can be obtained with our pipeline for the narrow domain of human heads. However, extending our method to other domains such as ImageNet categories [Deng et al. 2009] could enable learning more generic high-quality 3D priors. This could be done by having one template mesh per category and making the mesh itself learnable. As a result, the model could uncover suitable templates on its own. Additionally, an approach like that of Skorokhodov et al. [2023] could be employed to drop the requirement for domain-specific keypoint detectors and alignment procedure.

## 6 CONCLUSION

In this work, we have proposed Generative Gaussian Heads (GGHead), a fast method for training 3D generative models for human heads on large 2D data collections. At the core of our approach lies the combination of a 3D GAN pipeline with efficient rendering from 3D Gaussian Splatting. We achieve this by parameterizing 3D Gaussian Heads as a set of 2D maps that live in the UV space of a template mesh and that can be efficiently predicted with powerful 2D CNN architectures. We further propose a novel UV total variation loss that exploits the design of our generator to improve geometric fidelity of generated 3D heads. In experiments on FFHQ, we demonstrate that our method matches the image synthesis quality of the current state of the art while being both fully 3D-consistent and considerably faster. In timing benchmarks, we confirm the great scalability of our method giving rise to real-time generation and rendering of photo-realistic 3D heads at $1024^2$ resolution.

## Acknowledgements

This work was supported by the ERC Starting Grant Scan2CAD (804724) and the German Research Foundation (DFG) Research Unit "Learning and Simulation in Visual Computing". We would also like to thank Karla Weighart for proofreading and Angela Dai for the video voice-over.

## REFERENCES

Rameen Abdal, Wang Yifan, Zifan Shi, Yinghao Xu, Ryan Po, Zhengfei Kuang, Qifeng Chen, Dit-Yan Yeung, and Gordon Wetzstein. 2023. Gaussian shell maps for efficient 3d human generation. *arXiv preprint arXiv:2311.17857* (2023).

Robert C Bolles, H Harlyn Baker, and David H Marimont. 1987. Epipolar-plane image analysis: An approach to determining structure from motion. *International journal of computer vision* 1, 1 (1987), 7–55.

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.

David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. 2023. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. *arXiv preprint arXiv:2312.12337* (2023).

Xingyu Chen, Yu Deng, and Baoyuan Wang. 2023. Mimic3d: Thriving 3d-aware gans via 3d-to-2d imitation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2338–2348.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. 2020. StarGAN v2: Diverse Image Synthesis for Multiple Domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

Yu Deng, Jiaolong Yang, Jianfeng Xiang, and Xin Tong. 2022. Gram: Generative radiance manifolds for 3d-aware image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10673–10683.

Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. 2019. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 0–0.

Sankeerth Durvasula, Adrian Zhao, Fan Chen, Ruofan Liang, Pawan Kumar Sanjaya, and Nandita Vijaykumar. 2023. DISTWAR: Fast Differentiable Rendering on Raster-based Rendering Pipelines. *arXiv preprint arXiv:2401.05345* (2023).

Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A Efros, and Xiaolong Wang. 2023. Colmap-free 3d gaussian splatting. *arXiv preprint arXiv:2312.07504* (2023).

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. 2021. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. *arXiv preprint arXiv:2110.08985* (2021).

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

Shoukang Hu and Ziwei Liu. 2023. Gauhuman: Articulated gaussian splatting from monocular human videos. *arXiv preprint arXiv:2312.02973* (2023).

Yujiao Jiang, Qingmin Liao, Xiaoyu Li, Li Ma, Qi Zhang, Chaopeng Zhang, Zongqing Lu, and Ying Shan. 2024. UV Gaussians: Joint Learning of Mesh Deformation and Gaussian Textures for Human Avatar Modeling. *arXiv preprint arXiv:2403.11589* (2024).

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. 2020a. Training Generative Adversarial Networks with Limited Data. In *Proc. NeurIPS*.

Tero Karras, Samuli Laine, and Timo Aila. 2019a. 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, and Timo Aila. 2019b. 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. 2020b. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8110–8119.

Zhanghan Ke, Jiayu Sun, Kaican Li, Qiong Yan, and Rynson WH Lau. 2022. Modnet: Real-time trimap-free portrait matting via objective decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 1140–1147.

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.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. 2017. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36, 6 (2017), 194:1–194:17. https://doi.org/10.1145/3130800.3130813

Xinhai Li, Huaibin Wang, and Kuo-Kun Tseng. 2023. GaussianDiffusion: 3D Gaussian Splatting for Denoising Diffusion Probabilistic Models with Structured Noise. *arXiv preprint arXiv:2311.11221* (2023).

Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. 2023. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. *arXiv preprint arXiv:2312.13763* (2023).

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751* (2019).

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. 2018. Which training methods for gans do actually converge?. In *International conference on machine learning*. PMLR, 3481–3490.

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.

Arthur Moreau, Jifei Song, Helisa Dhamo, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. 2023. Human gaussian splatting: Real-time rendering of animatable avatars. *arXiv preprint arXiv:2311.17113* (2023).

Michael Niemeyer and Andreas Geiger. 2021. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11453–11464.

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.

Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. 2023a. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. *arXiv preprint arXiv:2312.02069* (2023).

Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 2023b. 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. *arXiv preprint arXiv:2312.09228* (2023).

Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. 2023. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142* (2023).

Shunsuke Saito, Gabriel Schwartz, Tomas Simon, Junxuan Li, and Giljoo Nam. 2023. Relightable gaussian codec avatars. *arXiv preprint arXiv:2312.03704* (2023).

Johannes L Schonberger and Jan-Michael Frahm. 2016. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4104–4113.

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.

Ivan Skorokhodov, Aliaksandr Siarohin, Yinghao Xu, Jian Ren, Hsin-Ying Lee, Peter Wonka, and Sergey Tulyakov. 2023. 3D generation on ImageNet. In *International Conference on Learning Representations*. https://openreview.net/forum?id=U2WjB9xxZ9q

Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. 2022. Epigraf: Rethinking training of 3d gans. *Advances in Neural Information Processing Systems* 35 (2022), 24487–24501.

Jingxiang Sun, Xuan Wang, Yichun Shi, Lizhen Wang, Jue Wang, and Yebin Liu. 2022. 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, Lizhen Wang, Xiaoyu Li, Yong Zhang, Hongwen Zhang, and Yebin Liu. 2023. Next3d: Generative neural texture rasterization for 3d-aware head avatars. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 20991–21002.

David Svitov, Pietro Morerio, Lourdes Agapito, and Alessio Del Bue. 2024. HAHA: Highly Articulated Gaussian Human Avatars with Textured Mesh Prior. *arXiv preprint arXiv:2404.01053* (2024).

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. 2023. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653* (2023).

Alex Trevithick, Matthew Chan, Towaki Takikawa, Umar Iqbal, Shalini De Mello, Manmohan Chandraker, Ravi Ramamoorthi, and Koki Nagano. 2024a. What You See is What You GAN: Rendering Every Pixel for High-Fidelity Geometry in 3D GANs. *arXiv preprint arXiv:2401.02411* (2024).

Alex Trevithick, Matthew Chan, Towaki Takikawa, Umar Iqbal, Shalini De Mello, Manmohan Chandraker, Ravi Ramamoorthi, and Koki Nagano. 2024b. What You See is What You GAN: Rendering Every Pixel for High-Fidelity Geometry in 3D GANs. *arXiv preprint arXiv:2401.02411* (2024).

Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. 2023. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3295–3306.

Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. 2024. latentSplat: Autoencoding Variational Gaussians for Fast Generalizable 3D Reconstruction. *arXiv preprint arXiv:2403.16292* (2024).

Jianfeng Xiang, Jiaolong Yang, Yu Deng, and Xin Tong. 2023. Gram-hd: 3d-consistent image generation at high resolution with generative radiance manifolds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2195–2205.

Yuelang Xu, Benwang Chen, Zhe Li, Hongwen Zhang, Lizhen Wang, Zerong Zheng, and Yebin Liu. 2023. Gaussian head avatar: Ultra high-fidelity head avatar via dynamic gaussians. *arXiv preprint arXiv:2312.03029* (2023).

Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. 2024. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arXiv preprint arXiv:2403.14621* (2024).

Yang Xue, Yuheng Li, Krishna Kumar Singh, and Yong Jae Lee. 2022. Giraffe hd: A high-resolution 3d-aware generative model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18440–18449.

Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. 2023. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529* (2023).

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

Hao Zhu, Wayne Wu, Wentao Zhu, Liming Jiang, Siwei Tang, Li Zhang, Ziwei Liu, and Chen Change Loy. 2022. CelebV-HQ: A large-scale video facial attributes dataset. In *European conference on computer vision*. Springer, 650–667.

Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. 2023. Drivable 3d gaussian avatars. *arXiv preprint arXiv:2311.08581* (2023).

Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. 2023. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. *arXiv preprint arXiv:2312.09147* (2023).

Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. 2001. EWA volume splatting. In *Proceedings Visualization, 2001. VIS'01.* IEEE, 29–538.