GSGAN: Adversarial Learning for Hierarchical Generation of 3D Gaussian Splats

Sangeek Hyun
Sungkyunkwan University

Jae-Pil Heo[*]
Sungkyunkwan University

Abstract

Most advances in 3D Generative Adversarial Networks (3D GANs) largely depend on ray casting-based volume rendering, which incurs demanding rendering costs. One promising alternative is rasterization-based 3D Gaussian Splatting (3D-GS), providing a much faster rendering speed and explicit 3D representation. In this paper, we exploit Gaussian as a 3D representation for 3D GANs by leveraging its efficient and explicit characteristics. However, in an adversarial framework, we observe that a naïve generator architecture suffers from training instability and lacks the capability to adjust the scale of Gaussians. This leads to model divergence and visual artifacts due to the absence of proper guidance for initialized positions of Gaussians and densification to manage their scales adaptively. To address these issues, we introduce GSGAN, a generator architecture with a hierarchical multi-scale Gaussian representation that effectively regularizes the position and scale of generated Gaussians. Specifically, we design a hierarchy of Gaussians where finer-level Gaussians are parameterized by their coarser-level counterparts; the position of finer-level Gaussians would be located near their coarser-level counterparts, and the scale would monotonically decrease as the level becomes finer, modeling both coarse and fine details of the 3D scene. Experimental results demonstrate that ours achieves a significantly faster rendering speed (×100) compared to state-of-the-art 3D consistent GANs with comparable 3D generation capability. Project page: https://hse1032.github.io/gsgan.

1 Introduction

Figure 1: Generated examples from the proposed method (FFHQ-512, AFHQ-Cat-512). Ours synthesize multi-view consistent images with a significantly faster rendering speed by leveraging 3D Gaussian representation. We represent a 3D scene as a composite of hierarchical Gaussians, where each level of Gaussian depicts coarse and fine details corresponding to its level. To visualize the effects of individual Gaussian, the right-most images are rendered by reducing the scale of Gaussians.

The research field of 3D generative models has recently emerged and shows impressive generation capability in various domains such as text-to-3D [1–7] and image-to-3D [8–10]. Among them,

---

[*]Corresponding author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

3D Generative Adversarial Networks [11–16] are an adversarial learning framework between a 3D generator and a 2D discriminator, capable of synthesizing 3D models solely by training with collections of 2D images or with the additional use of their corresponding camera poses. Specifically, the generator synthesizes a 3D model and renders it as a 2D image using given camera parameters, and then the 2D discriminator determines its realness and its match to the given camera pose.

The generally used volume rendering in 3D GANs is ray casting [17], which is widely adopted in NeRF literature [18–21]. Ray casting-based rendering demonstrates its prominent capability to model a 3D scene in various domains of research; however, it is also known for its excessive computational costs. Specifically, it requires $(H \times W \times D)$ point sampling for rendering an image from a single camera view, where H, W, D denotes height, width, the number of sampled points on a ray (depth). This demanding computational cost hinders previous methods in 3D GANs from performing the rendering process at higher resolutions [12] or forces them to use architectures optimized for efficient 3D representation, leading to degraded generation quality [13, 15].

Recently, 3D Gaussian Splatting (3D-GS) [22] has been introduced for the 3D reconstruction task. This method represents a 3D scene as a composition of 3D Gaussians, projecting and α-blending these Gaussians to render images. This rasterization-based method notably enhances rendering speed. Despite this innovative progress in the 3D reconstruction task, its application in 3D generative models has not yet been studied. One important challenge is the training algorithm of 3D-GS. Unlike the scene-independent and fully differentiable training of NeRFs, 3D-GS requires additional constraints such as the proper initialization of Gaussian positions for a given scene by SfM [23] and densification that heuristically manages the scale and number of Gaussians. Since 3D GANs deploy a single generator trained by gradient descent to model the distribution of 3D data, these non-differentiable and scene-dependent characteristics of 3D-GS restrict its application in 3D GANs.

In this paper, we extend the application of 3D Gaussian representation with rasterization to 3D GANs, leveraging its efficient rendering speed for high-resolution data. In an adversarial learning framework, we observe that a naïve generator architecture, which simply synthesizes a set of Gaussians without any constraints, suffers from training instability and imprecise adjustment of the scale of Gaussians. For example, at the early stage of training, all the Gaussians disappear in the rendered images, or there are visual artifacts created by long-shaped Gaussians despite its convergence.

To address these training difficulties, we devise a method to regularize the Gaussian representation for 3D GANs, focusing particularly on the position and scale parameters. To this end, we propose a generator architecture with a hierarchical Gaussian representation. This hierarchical representation models the Gaussians of adjacent levels to be dependent, encouraging the generator to synthesize the 3D space in a coarse-to-fine manner. Specifically, we first introduce a locality constraint whereby the positions of fine-level Gaussians are located near their coarse-level counterpart Gaussians and are parameterized by them, thus reducing the possible positions of newly added Gaussians. Then, we design the scale of Gaussians to monotonically decrease as the level of Gaussians becomes finer, facilitating the generator's ability to model the scene in both coarse and fine details. Based on this representation, we propose a generator architecture that effectively implements the hierarchy of Gaussians. With additional architectural details, we validate that the proposed generator successfully synthesizes realistic 3D models (Fig. 1) with stabilized training and enhanced generation quality.

Our contributions are summarized as follows:

- We firstly exploit a rasterization-based 3D Gaussian representation for efficient 3D GANs.
- We introduce GSGAN, a hierarchical 3D Gaussian representation that regularizes the parameters of Gaussians by building dependencies between Gaussians of adjacent hierarchies, stabilizing the training of Gaussian splatting-based 3D GANs.
- The proposed method achieves a significantly faster rendering speed compared to state-of-the-art 3D GANs while maintaining comparable generation quality and 3D consistency.

2 Related works

2.1 3D Generative Adversarial Networks

3D GANs [11, 16] learn the 3D representation from a collection of 2D images with corresponding camera parameters. Recent progress mostly focuses on generator architecture, especially for enhanc-

2

ing efficiency. For example, EG3D [12] introduces the tri-plane representation with an additional upsampling 2D image after volume rendering. Differently, there is another stream that directly renders a high-resolution image without a 2D upsampler. For instance, GRAM [13] learns the iso-surfaces for efficient sampling of points on a ray, based on an implicit-network generator [11]. Additionally, Voxgraf [24] exploits the sparse voxel grid instead of feature fields to represent the 3D scene. Epigraf [15] introduces a patch-wise training scheme to reduce the computational cost of volume rendering at high resolution (e.g., 512×512) during the training phase. Differently, Mimic3D [25] proposes a method to learn 3D representation at high resolution by distilling the 2D upsampling result of EG3D into the high-resolution tri-plane. Most recently, WYSIWIG [26] proposes SDF representation-based GANs for reducing the number of sampling points on rays in a high-resolution 3D representation.

While previous methods use ray casting as a volume rendering method with demanding rendering costs, we exploit efficient rasterization by adopting 3D Gaussians as a 3D representation.

2.2 Gaussian splatting for 3D representation

3D Gaussian Splatting (3D-GS) [22] represents a 3D scene as a composition of 3D Gaussians. This approach is known for its fast rendering speed, which rasterization-based rendering leads to, as well as its prominent reconstruction quality and faster convergence. Recently, various domains of research utilize 3D-GS, such as human avatar modeling[27–29], facial editing [30], text-to-3D synthesis [1–7], and image-to-3D synthesis [8–10]. These approaches typically substitute the neural radiance field with a Gaussian representation to leverage its efficient characteristics.

The research efforts also try to apply 3D Gaussian representation to an adversarial learning framework, especially with structural prior such as the human body and facial template. For example, Gaussian Shell Maps [31] focus on the task of 3D human generation based on SMPL template [32] with an adversarial learning framework. Given a human body template, the generator learns to synthesize multiple shell maps containing Gaussian parameters, achieving 3D consistent generation with faster rendering speed. Similarly, GGHead [33] focuses on 3D face generation using FLAME template [34]. Similar to Gaussian Shell Maps, they synthesize texture maps containing Gaussian parameters and position offsets by 2D generator based on StyleGAN2 [35]. However, these methods largely depend on the pre-defined template as structural prior.

Differently, in this paper, we focus on extending the application of 3D-GS to Generative Adversarial Networks without any geometric prior such as SfM that 3D-GS traditionally requires.

3 Proposed method

3.1 Preliminaries

3D Generative Adversarial Networks 3D GANs [16, 11, 12] is an adversarial learning framework between the 3D generator $g(z, \theta)$ and 2D discriminator $d(I)$. Specifically, for a given randomly sampled latent code $z \in \mathbb{R}^{d_z} \sim p_z$ and camera pose $\theta \in \mathbb{R}^{d_\theta} \sim p_\theta$, the generator $g(z, \theta)$ synthesizes a 3D scene and renders it to a 2D fake image $I_f \in \mathbb{R}^{H \times W \times 3}$ based on the given camera pose. Then, the discriminator learns to discriminate the real image $I_r \in \mathbb{R}^{H \times W \times 3}$ and the fake image I_f, while the generator learns to deceive the discriminator. In addition, the discriminator often uses the camera pose to encourage d to be aware of 3D information by making d estimate the camera parameter [13, 36] or performing conditional adversarial loss [12].

3D Gaussian splatting 3D Gaussian representation for 3D modeling is recently introduced by 3D Gaussian Splatting [22]. It represents a scene as a composition of anisotropic 3D Gaussians estimated by multi-view images and their corresponding camera poses. This Gaussian representation contains parameters such as a position $\mu \in \mathbb{R}^3$, a scale $s \in \mathbb{R}^3$, a quaternion $q \in \mathbb{R}^4$, an opacity $\alpha \in \mathbb{R}^1$, and a color $c \in \mathbb{R}^3$. In world space, Gaussian is defined as follows:

$$G(x) = e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}, \quad \Sigma = RSS^T R^T, \tag{1}$$

where $R \in \mathbb{R}^{4 \times 4}$ and $S \in \mathbb{R}^{4 \times 4}$ are a rotation matrix and a scaling matrix obtained from a quaternion q and scale s, and x is a world coordinate.

3

(a) Illustration of hierarchical representation of Gaussians

(b) Example of generated Gaussians of various hierarchy levels
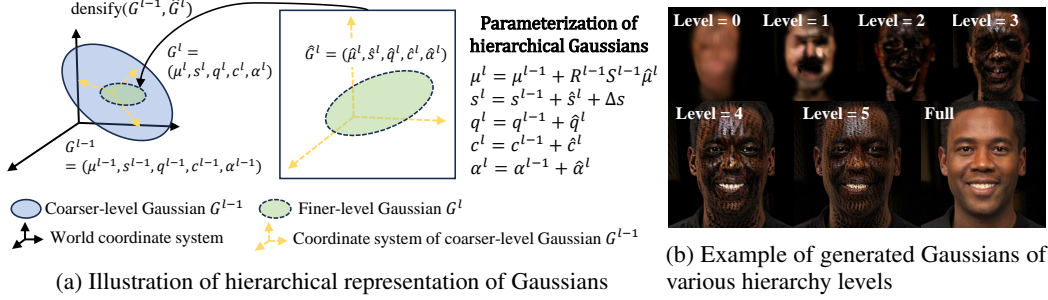
Figure 2: Illustration and examples of hierarchical Gaussian representation. (a) We parameterize the finer-level Gaussians by the parameters of coarser-level counterparts for regularizing the scale and position of synthesized Gaussians. (b) Example of synthesized Gaussians across multiple hierarchy levels. Gaussians represent coarse or fine details according to its hierarchy level.

To render the image, the color of each pixel C is determined by blending the contributions of all N 3D Gaussians that overlap with the pixel as follows:

$$C = \sum_{i=1}^{N} c_i \alpha_i' \prod_{j=1}^{i-1} (1 - \alpha_j'), \tag{2}$$

where c_i is the color of each Gaussian, and α' is blending weight of 2D projection of Gaussian multiplied by a per-point opacity α. The order of Gaussians is sorted by their depth.

3.2 Hierarchical 3D Gaussian representation

Our focus is on utilizing Gaussians as the 3D representation of the generator in 3D GANs. We begin with a simple generator that takes a randomly sampled latent code z as input and outputs N Gaussians, without any restrictions. However, in this scenario, we observe that this naïve application suffers from training instability and fails to properly manage the position μ and scale s, as shown in examples in Fig. 8. Therefore, we concentrate on guiding the position and scales that the generator synthesizes.

To this end, we propose a hierarchical structure of Gaussians to effectively regularize the position and scale of Gaussians, as illustrated in Fig. 2. Firstly, we define the hierarchy level $l \in \{0, ..., L-1\}$, from coarse to fine levels, where each level contains a set of Gaussian parameters. In detail, we establish a dependency between the Gaussian parameters of adjacent levels. For simplification, we explain the dependency between two hierarchically adjacent Gaussians, G^l and G^{l-1}, where G^l originates from G^{l-1}. We aim to model the 3D representation in a coarse-to-fine manner, by assigning coarser- and finer-level Gaussians G^{l-1} and G^l to be responsible for coarser and finer details of the 3D scene, respectively.

For the position parameter, we impose a locality constraint that bounds the position μ^l of the finer-level Gaussian G^l with its corresponding G^{l-1}. Specifically, we introduce the local position parameter $\hat{\mu}^l$ defined in the local coordinate system, which is centered, rotated, and scaled by the position μ^{l-1} and scale s^{l-1} and quaternion q^{l-1} of the coarser-level Gaussian G^{l-1}. Then, the position μ^l in world space is formulated by transforming the local position $\hat{\mu}^l$ as follows:

$$\mu^l = \mu^{l-1} + R^{l-1} S^{l-1} \hat{\mu}^l, \tag{3}$$

where R^{l-1} and S^{l-1} are a rotation and scaling matrix obtained from q^{l-1} and s^{l-1}. This operation ensures the position of Gaussians at finer levels depends on coarser-level Gaussians, while residing near the location of coarser-level counterparts.

For the scale parameter, we enforce the scale parameter to monotonically decrease to a certain degree as its hierarchy level increases. In detail, we define the scale of the finer-level s^l using the relative scale difference $\hat{s}^l$ against the coarser-level scale s^{l-1}. Additionally, we restrict this scale difference $\hat{s}^l$ to always be a vector of negative values. Furthermore, we introduce the constant Δs which further lowers the scale of the finer-level. This process is defined as follows:

$$s^l = s^{l-1} + \hat{s}^l + \Delta s, \quad \text{where} \quad \hat{s}^l, \Delta s < 0. \tag{4}$$

4

Generator architecture	Hierarchical 3D Gaussians	

(a) Illustration of generator architecture with hierarchical 3D Gaussians (b) `block` architecture
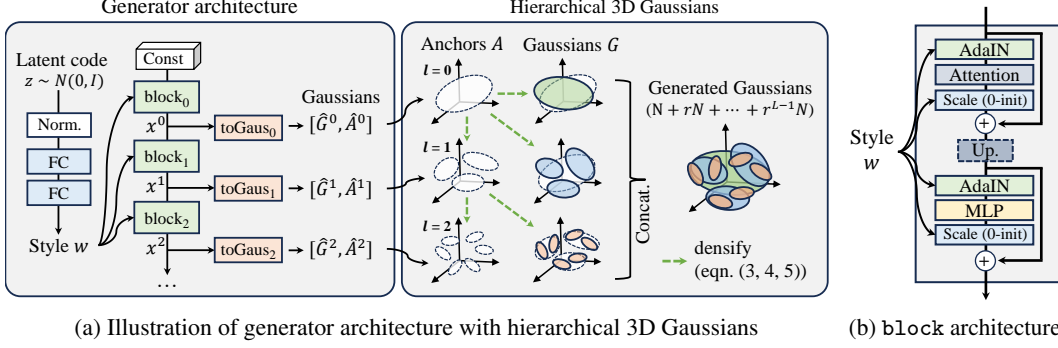
Figure 3: The architecture of the generator and its block. (a) Generator synthesizes the multiple-level of anchors and Gaussians, which contains the residual parameters $\hat{A}^l$ and $\hat{G}^l$. Anchors are utilized to regularize the finer-level Gaussians, while Gaussians are used for actual rendering. After generating these parameters, we combine them with anchors from previous level A^{l-1} by `densify` operation, as defined in eqn. 3, 4, 5 (green arrow). (b) The generator consists of stacks of `blocks`, each of which is a sequence of attention and MLP layers. The latent code z is conditioned on the generator through AdaIN and layerscale, where the modulation and scaling parameters are derived from style code w.

For the other parameters, we additionally define the residual Gaussian parameters $\hat{\alpha}^l, \hat{q}^l, \hat{c}^l$ at level l, which are added to the Gaussian parameters of the previous level as follows:

$$\alpha^l = \alpha^{l-1} + \hat{\alpha}^l, \quad q^l = q^{l-1} + \hat{q}^l, \quad c^l = c^{l-1} + \hat{c}^l. \tag{5}$$

We call this hierarchical relationship between G^{l-1} and Gaussians with residual parameters $\{\hat{\mu}^l, \hat{s}^l, \hat{\alpha}^l, \hat{q}^l, \hat{c}^l\} \in \hat{G}^l$ as `densify`$(G^{l-1}, \hat{G}^l)$, and it enables the generator to model the 3D space in a coarse-to-fine manner, where the fine-level Gaussians depict the detailed part of the coarse-level counterparts. Importantly, it stabilizes the training of GANs by significantly reducing the possible positions of Gaussians and encourages the generator to use various scales of Gaussians, thereby boosting the generation capability to model both coarse and fine details.

3.3 GSGAN; the generator architecture with hierarchical 3D Gaussians

In this section, we propose a generator architecture for leveraging the aforementioned hierarchical structure of Gaussians (Fig. 3). Basically, we adopt a transformer-based architecture, composed of stacks of attention and MLP layers, which is a generally used architecture to handle unstructured 3D point cloud data [37–39].

First of all, we define a generator $g(z, \theta)$ as a sequence of generator blocks, where `block`$_l$ denotes the generator block at a specific level l. At the coarsest level $l = 0$, `block`$_0$ takes input as N learnable positions `const` $\in \mathbb{R}^{N \times 3}$ and latent code z, where N is the number initial Gaussians, then outputs the high-dimensional features x^0. Then, for a feature of ith Gaussian x_i^0, we process the feature x_i^0 by the output layer `toGaus`$_0$ to obtain a Gaussian parameter $\{\mu_i^0, s_i^0, q_i^0, \alpha_i^0, c_i^0\} \in G_i^0$. For arbitrary level l, `block`$_l$ takes input as feature x_i^{l-1} from the previous block and latent code z, then outputs the feature x_j^l. Importantly, the output block `toGaus`$_l$ does not directly synthesize the Gaussian parameters G_j^l. Instead, the intermediate output $\hat{G}_j^l$ contains the local position $\hat{\mu}_j^l$ and relative scale difference $\hat{s}_j^l$, as well as the other residual parameters $\hat{c}_j^l, \hat{\alpha}_j^l, \hat{q}_j^l$. This intermediate output $\hat{G}_j^l$ is combined with the corresponding Gaussians G_i^{l-1} from the previous level to finally synthesize the Gaussian G_j^l at level l, following the operations in Sec. 3.2. This process, which establishes the hierarchical dependency between Gaussians of adjacent level G_i^{l-1} and G_j^l, is defined as follows:

$$x_j^l = \text{block}_l(x_i^{l-1}, z), \quad \hat{G}_j^l = \text{toGaus}_l(x_j^l), \quad G_j^l = \text{densify}(G_i^{l-1}, \hat{G}_j^l), \tag{6}$$

where `densify` operation denotes the combining process of parameters in hierarchically adjacent G_i^{l-1} and $\hat{G}_j^l$ mentioned in eqn. 3, 4, 5.

5

As the fine-level Gaussians have smaller scales compared to the coarse-level ones, the number of Gaussians should be increased as the hierarchy increases to successfully synthesize the fine details. Thus, we expand the number of Gaussian parameters G_i^l to have a total of $r^l N$ vectors for each parameter, where r is an upsampling ratio. In other words, we define the Gaussian G_j^l to be dependent on G_i^{l-1}, where $j = ri + k$ and $k \in \{0, 1, ..., r-1\}$.

After synthesizing the Gaussian parameters of every level, we use all of them for generating the image (i.e. total $(N + rN + ... + r^{L-1}N)$ Gaussians are used for rendering). For rendering, we use a tile-based rasterizer following 3D-GS [22].

Anchor Gaussians to decompose Gaussians for regularization and rendering   In the aforementioned architecture, Gaussians are used not only to represent a 3D scene but also to regularize their coarser-level Gaussian counterparts. This means Gaussians must be trained to precisely guide the parameters of finer-level Gaussians while simultaneously depicting the sharp details in real-world images. However, achieving both of these objectives can be challenging. For instance, we observe that the scale of Gaussians can become nearly zero along a specific axis, leading to excessively strong regularization on the position of finer-level Gaussians. To handle this issue, we introduce an auxiliary set of Gaussians that only contributes to regularization, instead of actual rendering.

Specifically, we introduce an anchor Gaussian A^l for a specific level l, which has identical parameterization to G^l. However, this type of Gaussian is only used for regularization by deploying it as the input of `densify`, especially for the coarser-level Gaussian input. Therefore, an anchor Gaussian A_i^{l-1} only learns to guide the parameters of their finer-level counterpart G_j^l, so its usage relieves the effects of strong regularization caused by zero variance. To generate it, we simply make `toGaus`$_l$ synthesize two sets of Gaussians, $\hat{G}_j^l$ and $\hat{A}_j^l$, for a given feature x_j^l. This process is achieved by re-defining the eqn. 6 as follows:

$$[\hat{G}_j^l, \hat{A}_j^l] = \texttt{toGaus}_l(x_j^l), \quad G_j^l = \texttt{densify}(A_i^{l-1}, \hat{G}_j^l), \quad A_j^l = \texttt{densify}(A_i^{l-1}, \hat{A}_j^l). \quad (7)$$

As the Gaussians of coarsest level $l = 0$ does not have their coarser-level counterparts, we define $A_i^0 = \hat{A}_i^0$ and $G_i^0 = \texttt{densify}(A_i^0, \hat{G}_i^0)$.

Architectural details   Following the conditioning convention of previous GANs [35, 40], we utilize the mapping network to modify the latent code z into the style code w. Then, the style code affects the synthesis process by AdaIN [41]. As noted, the generator block is a stack of attention and MLP layers, then we replace the layer norm in attention and MLP by AdaIN, following generally used approach for transformer-based GANs [42, 43]. For blocks of coarser levels, we utilize the general self-attention without positional encoding as the attention mechanism, whereas we use local attention [44] for finer levels, as the interaction between $r^{l-1}N$ points is computationally demanding. For expanding the features in generator blocks after the coarsest level, we simply use the subpixel operation [45] with skip connection and repeat Gaussians G^l in r times.

One important architectural design is the usage of layerscale [46], which is a learnable vector that adjusts the effects of the residual block by multiplying it by the output of the residual block. Typically, it is initialized by a zero-valued vector, removing the effects of layers in the early stage of training. We observe it is essential for stabilizing the position of Gaussians in early iterations. In addition, we use the adaptive version of layerscale [47] conditioned by latent code z on every attention and MLP layer in the generator.

Also, we use the camera direction as a condition for the color layer in `toGaus` to model the view-dependent characteristics and employ the background generator, which resembles the generator architecture but with reduced capacity and results in the Gaussians located within a sphere of a radius of 3, while the foreground resides in the [-1, 1] cube. For further details, we elaborate on them in the Appendix A.1.

3.4   Training objectives

Similar to previous 3D GANs [12], we adopt the non-saturating adversarial loss [48] with R1 regularization [49]. Formally, these objective functions are defined as follows:

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{z \sim P_z, \theta \sim P_\theta}[f(d(g(z, \theta)))] + \mathbb{E}_{I_r \sim P_{I_r}}[f(-d(I_r)) + \lambda ||\nabla d(I_r)||^2], \quad (8)$$

6

Table 1: Quantitative comparison on FFHQ and AFHQ-Cat datasets in terms of FID-50K-full and rendering time. We mainly compare ours with the 3D consistent models, except EG3D utilizing the 2D upsampler. FID scores are taken from previous work [25]. Rendering time is measured on a single RTX A6000 GPU. *The rendering time of EG3D at 512 resolution consists of the time for volume rendering at 128 resolution and 2D upsampling operations.

Methods	3D consistency	FFHQ		AFHQ-Cat		Rendering time (ms)	
		256×256	512×512	256×256	512×512	256×256	512×512
EG3D [12]		4.80	4.70	3.41	2.72	-	15.5*
GRAM [13]	✓	13.8	-	13.4	-	-	-
GMPI [54]	✓	11.4	8.29	-	7.67	-	-
EpiGRAF [15]	✓	9.71	9.92	6.93	-	-	-
Voxgraf [24]	✓	9.60	-	9.60	-	-	-
GRAM-HD [55]	✓	10.4	-	-	7.67	173.0	197.9
Mimic3D [25]	✓	**5.14**	**5.37**	4.14	4.29	106.8	402.1
GSGAN (Ours)	✓	6.59	5.60	**3.43**	**3.79**	**2.7**	**3.0**

where $f(t) = -\log(1 + \exp(-t))$ is a softplus function and λ is R1 regularization strength.

We additionally guide the 3D information to the discriminator and generator by introducing contrastive learning between pose embedding obtained from the images and camera parameters. Specifically, the discriminator has a pose branch d_p that estimates the pose embedding p_I from an input image. Then, we introduce a pose encoder that consists of MLP layers and encodes the camera parameter θ into the pose embedding p_θ. Similar to previous work [36], we utilize the contrastive objective which enhances similarity between corresponding p_I and p_θ. Formally, this objective is defined as follows:

$$\mathcal{L}_{\text{pose}} = -\log\left(\frac{\exp(\text{sim}(p_I, p_\theta^+)/\tau)}{\sum_{b=1}^{B}(\exp(\text{sim}(p_I, p_\theta^b)/\tau))}\right), \tag{9}$$

where $\text{sim}(\cdot, \cdot)$ is a cosine similarity and B is a batch size and p_θ^+ is a positive sample corresponding to a pose embedding p_I from the image, and τ is a temperature scaling parameter. For the discriminator, we calculate $\mathcal{L}_{\text{pose}}$ using real data, while using fake data for training the generator.

Furthermore, we introduce two losses to regularize the position of anchor Gaussians in the coarsest level, μ_A^0. We first regularize the averaged position of μ_A^0 to be zero for encouraging the center of Gaussians residing near the origin of the world space. Secondly, we reduce the distance between the positions of the K nearest anchor Gaussians to prevent anchor Gaussians from falling apart from the others. These two regularization losses are defined as follows:

$$\mathcal{L}_{\text{center}} = \frac{1}{N}||\sum_{j=1}^{N}\mu_{A,j}^0||^2, \quad \mathcal{L}_{\text{knn}} = \frac{1}{NK}\sum_{j=1}^{N}||\sum_{k=1}^{K}(\mu_{A,j}^0 - \text{KNN}(\mu_{A,j}^0, k))||^2, \tag{10}$$

where $\text{KNN}(\mu_{A,j}^0, k)$ is the position of kth nearest neighbor of jth anchor Gaussian of μ_A^0.

To sum up, the final objective function $\mathcal{L}$ is as follows:

$$\mathcal{L} = \mathcal{L}_{\text{adv}} + \lambda_{\text{pose}}\mathcal{L}_{\text{pose}} + \lambda_{\text{center}}\mathcal{L}_{\text{center}} + \lambda_{\text{knn}}\mathcal{L}_{\text{knn}}, \tag{11}$$

where λ_{pose}, λ_{center}, and λ_{knn} are strengths of the corresponding objective function.

4 Experiments

4.1 Experimental settings

Following the experimental settings of previous 3D GANs [12, 25], we use FFHQ [35] and AFHQ-Cat [50] datasets with 256×256 and 512×512 resolutions. For example, we augment the datasets with the horizontal flip and additionally use adaptive data augmentation [51] for AFHQ-Cat dataset, which has a limited size. Camera pose labels are obtained from the official repository of EG3D [12], which are predicted by off-the-shelf pose estimators [52, 53]. We train the model from scratch on each dataset. For further implementation details, please refer to Appendix A.1.

7

Figure 4: Qualitative results of the proposed method with truncation psi ($\psi = 0.7$).
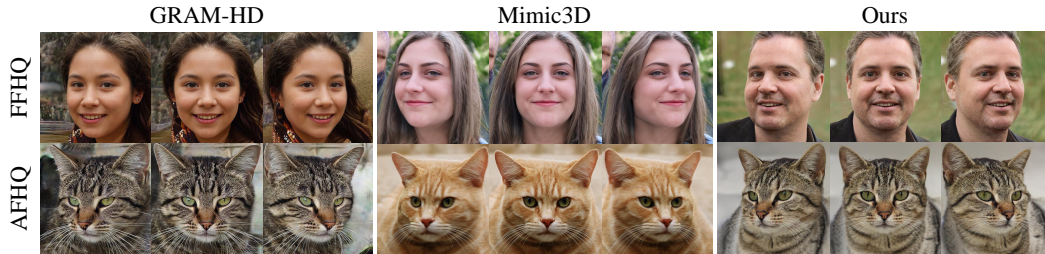
Figure 5: Qualitative comparison with 3D consistent methods with truncation psi ($\psi = 0.7$).

4.2 Experimental results

Quantitative results We mainly compare the proposed method with previous 3D consistent GANs, in terms of FID. These methods have strict 3D consistency, which directly renders the high-resolution images from 3D representation without any 2D upsampling operations. As reported in Tab. 1, we validate that the proposed method surpasses most of the previous methods and also achieves comparable generation capability compared to the state-of-the-art, Mimic3D. Especially for AFHQ-Cat dataset, we achieve a much lower FID, even comparable to the non-3D consistent baseline, EG3D. Next, we evaluate the rendering speed enhancement of the proposed method compared to baseline methods. To compute it, we first synthesize a 3D representation of each model and measure the processing time of rendering. As reported, ours shows significantly faster speeds compared to the baseline methods, achieving more than 100 times faster rendering than Mimic3D in $512{\times}512$ resolution. Moreover, it is even faster than non-3D consistent baseline EG3D, which is the model that exploits 2D upsampling operation for reducing the efficient rendering. In addition, one important point is that rendering time is almost identical regardless of the image resolution, suggesting that the proposed method can be more effective at higher resolutions. Additionally, the training time of the proposed methods is 28 RTX A6000 days, while the state-of-the-art Mimic3D requires 64 A100 days on the FFHQ-512 dataset. This notable gap in training cost implies that ours can achieve comparable generation capability efficiently in both rendering and training speed.

Qualitative results We present examples of generated images from the proposed method and the most recent 3D consistent GANs [25, 55], in Fig. 5. In both datasets, we observe that the proposed method successfully synthesizes the multi-view consistent images, validating its capability for synthesizing the realistic 3D scene. Also, we validate ours can generate both coarse and fine details such as coarse details of skin in human facial images and fine details of fur in cat images.

Level-by-level visualization To further understand how the synthesized Gaussians work, we visualize the Gaussians from an individual level. As depicted in Fig. 6, we observe that Gaussians capture the image components from overall structure to fine details as the level increases.

8

| $l = 0$ | $l = 1$ | $l = 2$ | $l = 3$ | $l = 4$ | $l = 5$ | Composited. |

Figure 6: Level-by-level visualization of synthesized Gaussians.

Table 2: Comparison of 3D consistency. We compare ours with the most recent 3D consistent GANs [55, 25]. Faces are segmented by off-the-shelf model [56] to remove effects of background.

	FFHQ-256		FFHQ-512	
	PSNR	SSIM	PSNR	SSIM
GRAM-HD	34.45	0.9648	32.15	0.9244
Mimic3D	40.36	**0.9927**	35.95	**0.9822**
GSGAN	**41.69**	0.9883	**37.85**	0.9695
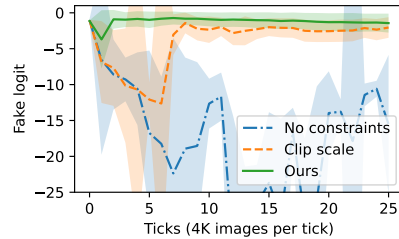
Figure 7: Fake logits of ablated models at the early stage of training.

Comparison with 3D consistency   As a 3D generative model, it is important to maintain 3D consistency across different views. To validate the 3D consistency of the synthesized 3D model, we measure how well the generated 3D scene is reconstructed by a surface estimation model, following previous works [25, 55]. Specifically, we fit the surface estimation model to a generated multi-view image and compute the reconstruction error of the multi-view images used for training. For surface estimation, we use Neus2 [57], applying a facial segmentation mask estimated from an off-the-shelf network [56] to eliminate the effects of the background. Furthermore, we utilize only the foreground generator for our method. As shown in Tab. 2, the proposed method significantly outperforms GRAM-HD while achieving performance comparable to Mimic3D. This experiment suggests that our approach generates 3D-consistent results by leveraging the explicit 3D representation provided by 3D Gaussians.

Training stability at the early stage of training   We conduct an experiment to assess the effect of the proposed method on training stability, particularly at the early stage of training, by observing the fake logit of the discriminator. As shown in Fig. 7, we observe that the model without any constraints exhibits rapid divergence, accompanied by a markedly low fake logit, indicating that the discriminator already distinguishes between real and fake data perfectly. When applying a minimal constraint that limits the scale to its predefined maximum, the model does not diverge but still suffers from instability, as evidenced by a large standard deviation of the logit. In contrast, ours demonstrates stable training compared to other models. Note that, the standard deviation is visualized by $1\text{-}\sigma$.

Ablation study   We perform an ablation study on the proposed components, particularly focusing on the proposed hierarchical architecture. When ablating the regularizations, we keep the residual representation of other parameters, (q, α, c). In the absence of any constraints, the model exhibited an FID score exceeding 300, signifying a failure to converge, in Tab. 3. With a minimal constraint that clips scales, the model does not diverge but significantly suffers from its low generation capability. As we gradually attach the proposed components, we observe enhancements in FID, showing the effects of the regularization of position and scale and the introduction of anchor Gaussians.

In Fig. 8, we provide visualizations of synthesized Gaussian positions. Initially, a simple clipping of the scale allow the model to synthesize images, but it leads to visual artifacts due to elongated Gaussians with large scales. Additionally, many Gaussians remain outside the scene, likely due to their tendency not to overlap, particularly when large-scale Gaussians exist. After applying position regularization, Gaussians become more densely located, although visual artifacts and unused

9

Table 3: Ablation study on FFHQ-256. "Clipping scale" means the model with the maximum limit of scale.

	FID
No constraints	300∼
+ Clipping scale	95.97
+ Position reg. (eqn. 3)	17.65
+ Scale reg. (eqn. 4)	13.80
+ Background generator	12.61
+ Anchor Gaussian	6.59
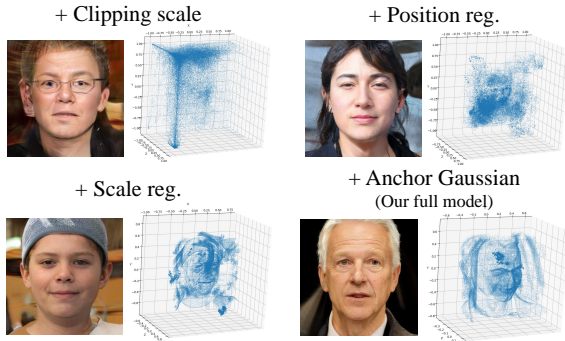
Figure 8: Visual comparison with ablated models. Points are visualized by the position of Gaussians.

Gaussians persist. Upon introducing scale regularization, the synthesized images no longer exhibit such artifacts, but the model struggles to capture precise geometries, as evidenced by the point cloud visualizations. Finally, the model incorporating anchor Gaussians successfully synthesizes realistic images while accurately estimating Gaussian positions. Note that, we clip the Gaussian positions to exist within the range of [-1, 1] in the cube.

Additional visualizations   We additionally provide 1) additional generated examples, 2) examples with densely changed camera positions, 3) latent interpolation and w+ inversion, 4) visualization of anchor Gaussians, and 5) effects of background generator in Appendix A, and code implementation with additional videos in supplementary zip file, so please refer to them.

5   Broader Impact and Limitations

Broader Impact   The proposed method follows the negative social impact of previous 3D generative models. For example, it may be used for synthesizing fake news or deepfake. Furthermore, since we boost the rendering speed of 3D GANs, ×100 faster than previous methods, it can encourage the generation of them by reducing computation cost for rendering the image.

Limitations   Different from the 3D-GS, which adaptively removes and introduces Gaussians by densification, the proposed method synthesizes a fixed number of Gaussians. This lack of adaptivity in the number of Gaussians remains a limitation, as the number of Gaussians can differ depending on the scene it makes. Also, the scale in hierarchical Gaussian representation is somewhat dependent on the hyperparameter such as Δs. These factors require adjustment of hyperparameters and can affect the performance of the generator.

6   Conclusion

In this paper, we exploit the 3D Gaussian representation in the domain of 3D GANs, leveraging its fast rendering speed with explicit 3D representation. To relieve the absence of proper initialization and densification process of 3D-GS, we propose hierarchical Gaussian representation which effectively regularizes the position and scale of generated Gaussians. We validate the proposed technique to stabilize the training of the generator with 3D Gaussians and encourage the model to learn the precise geometry of 3D scene, achieving almost ×100 faster rendering speed with a comparable 3D generation capability.

7   Acknowledgements

This work was supported in part by MSIT&KNPA/KIPoT (Police Lab 2.0, No. 210121M06), MSIT/IITP (No. 2022-0-00680, 2020-0-01821, 2019-0-00421, RS-2024-00459618, RS-2024-00360227, RS-2024-00437102, RS-2024-00437633), and MSIT/NRF (No. RS-2024-00357729).

10

References

[1] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 867–876, 2022.

[2] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

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

[4] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[5] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023.

[6] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12619–12629, 2023.

[7] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22246–22256, 2023.

[8] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. Dreambooth3d: Subject-driven text-to-3d generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2349–2359, 2023.

[9] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22819–22829, 2023.

[10] Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Realfusion: 360deg reconstruction of any object from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8446–8455, 2023.

[11] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. 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*, pages 5799–5809, 2021.

[12] 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. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16123–16133, 2022.

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

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

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

[16] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7588–7597, 2019.

[17] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

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

11

[19] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[20] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

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

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

[23] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006.

[24] Katja Schwarz, Axel Sauer, Michael Niemeyer, Yiyi Liao, and Andreas Geiger. Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids. *Advances in Neural Information Processing Systems*, 35:33999–34011, 2022.

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

[26] Alex Trevithick, Matthew Chan, Towaki Takikawa, Umar Iqbal, Shalini De Mello, Manmohan Chandraker, Ravi Ramamoorthi, and Koki Nagano. What you see is what you gan: Rendering every pixel for high-fidelity geometry in 3d gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22765–22775, 2024.

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

[28] Zhe Li, Zerong Zheng, Lizhen Wang, and Yebin Liu. Animatable gaussians: Learning pose-dependent gaussian maps for high-fidelity human avatar modeling. *arXiv preprint arXiv:2311.16096*, 2023.

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

[30] Zhenglin Zhou, Fan Ma, Hehe Fan, and Yi Yang. Headstudio: Text to animatable head avatars with 3d gaussian splatting. *arXiv preprint arXiv:2402.06149*, 2024.

[31] Rameen Abdal, Wang Yifan, Zifan Shi, Yinghao Xu, Ryan Po, Zhengfei Kuang, Qifeng Chen, Dit-Yan Yeung, and Gordon Wetzstein. Gaussian shell maps for efficient 3d human generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9441–9451, 2024.

[32] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866. 2023.

[33] Tobias Kirschstein, Simon Giebenhain, Jiapeng Tang, Markos Georgopoulos, and Matthias Nießner. Gghead: Fast and generalizable 3d gaussian heads. *arXiv preprint arXiv:2406.09377*, 2024.

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

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

[36] Kyungmin Jo, Wonjoon Jin, Jaegul Choo, Hyunjoon Lee, and Sunghyun Cho. 3d-aware generative model for improved side-view image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22862–22872, 2023.

[37] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021.

12

[38] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021.

[39] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

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

[41] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[42] Kwonjoon Lee, Huiwen Chang, Lu Jiang, Han Zhang, Zhuowen Tu, and Ce Liu. Vitgan: Training gans with vision transformers. *arXiv preprint arXiv:2107.04589*, 2021.

[43] Bowen Zhang, Shuyang Gu, Bo Zhang, Jianmin Bao, Dong Chen, Fang Wen, Yong Wang, and Baining Guo. Styleswin: Transformer-based gan for high-resolution image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11304–11314, 2022.

[44] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019.

[45] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[46] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 32–42, 2021.

[47] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

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

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

[50] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020.

[51] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in neural information processing systems*, 33: 12104–12114, 2020.

[52] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. 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*, pages 0–0, 2019.

[53] T. B. Lee. Cat hipsterizer,. https://github.com/kairess/cat_hipsterizer, 2018.

[54] Xiaoming Zhao, Fangchang Ma, David Güera, Zhile Ren, Alexander G Schwing, and Alex Colburn. Generative multiplane images: Making a 2d gan 3d-aware. In *European Conference on Computer Vision*, pages 18–35. Springer, 2022.

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

[56] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.

13

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

[58] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[59] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4432–4441, 2019.

[60] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Transactions on graphics (TOG)*, 42(1):1–13, 2022.

14

A Appendix

A.1 Implementation details

First of all, please refer to the attached code in supplementary for a more detailed implementation of the proposed method. For the coefficients of objective functions, we use $\lambda = 1$, $\lambda_{\text{pose}} = 1$, and $\lambda_{\text{knn}} = 10$. We train the model until the discriminator sees 10-15M images. Differently, we use $\lambda_{\text{center}} = 1, 10$ each for FFHQ and AFHQ-Cat datasets. Following prior works [12, 25], we also generator pose conditioning [12] for modeling pose-specific attributes. Other hyperparameters not mentioned are identical to EG3D [12].

For the architectural details, we use an upsampling ratio $r = 4$ and the initial number of Gaussians $N = 256$. For the number of hierarchical levels, we adopt $L = 5$ for the dataset with 256×256 resolution, and $L = 6$ for 512×512 resolution. We use the stack of 6 `blocks` at the lowest level and 1 `block` for the other levels, that is, 87K and 349K Gaussians are used for 256 and 512 resolutions. The number of channels is [512, 512, 256, 128, 128, 128] for the block of levels [0, 1, 2, 3, 4, 5], respectively. We use equalized learning rate [58] for every layer. Also, we only apply AdaIN and adaptive layerscale on the layers lower than level 3, as we observe the modulation on higher levels does not affect the rendering results much. For discriminator architecture, we use StyleGAN2 discriminator following EG3D [12], but with an additional pose embedding layer composed of the MLP layer.

For the representation of Gaussian parameters, we follow the 3D-GS [22] that uses the exponential activation function for the scale parameters and normalizes the quaternion to make it valid. As these operations are performed in the rendering process, we omit them for simplicity.

Attention layer   We deploy the attention layer for modeling the interaction between the Gaussians. For the lowest levels $l = 0$, we utilize the general self-attention. Since the self-attention on a large number of Gaussians requires too much computation cost, we use local attention similar to EdgeConv [44], which computes attention scores against KNN points ($k = 16$), for the levels $l > 0$. As calculating the KNN points also becomes demanding where the number of points to be enlarged, we model the points to interact with the points from the identical parent point, for levels $l > 3$, instead of computing the actual KNN points.

Output layer (`toGaus`)   `toGaus` layer is basically a set of linear layers that synthesize the Gaussian parameter. That is, it consists of 5 linear layers which are parallelly computed to output $\hat{\mu}, \hat{s}, \hat{\alpha}, \hat{q}, \hat{c}$, respectively. They have an input as the intermediate feature x^l, and an additional camera parameter θ only for the color $\hat{c}$. In addition, the layer for the relative position $\hat{\mu}$ has a tanh function as the activation to restrict the range of output position, and the layer for the scale difference $\hat{s}$ has a softplus function multiplied by -1 as the activation to make it have negative values. Thus, `toGaus` layer at a specific level is defined as follows:

$$p^l = \text{Lin}_p(x^l), \quad \hat{\mu}^l = \tanh(\text{Lin}_{\hat{\mu}}(x^l)), \quad \hat{s}^l = -\text{softplus}(\text{Lin}_{\hat{s}}(x^l)), \quad \hat{c}^l = \text{Lin}_{\hat{c}}(x^l, \theta), \quad (12)$$

where $p \in \{\hat{\alpha}, \hat{q}\}$ and $\text{Lin}_{(\cdot)}$ is a linear layer for the Gaussian parameter. Also, we initialize the bias of linear layers to set the quaternion to be an identity rotation matrix and the opacity to be 0.1 and the scale to be $e^{-\frac{1}{\sqrt{N}}}$.

The constant Δs reducing the scale along with hierarchy is adjusted by the resolution of the given data as below:

$$\Delta s = -\log(\sqrt{H \times W}/(L \times \sqrt{N})), \quad (13)$$

where H, W are the height and width of the image, L is the total level of hierarchy, and N is the initial number of Gaussians at level $l = 0$.

Background generator   The architecture of the background generator is similar to the proposed generator, but the architecture of `block` is different. In detail, we do not utilize the attention and MLP layers, instead using simple linear layers with demod- and modulation [35] to reduce the computational cost. Also, the position parameters synthesized by the background generator are normalized to make the generated Gaussians located on the sphere. We set the radius of the sphere as 3.

15

FFHQ-512   AFHQ-Cat-512

Figure 9: Generated examples on FFHQ-512 and AFHQ-Cat-512

For the hyperparameters, we set the number of initial Gaussians $N = 2000$, and the number of hierarchy $L = 2$ with upsampling factor $r = 4$ and the channel of every layer as 128. Thus, the background is composed of 10,000 Gaussians in total.

Training trick for FFHQ dataset We observe that training on the FFHQ dataset lacks the capability to accurately model the precise structure of 3D scenes, particularly for structures viewed from different vertical angles. To address this issue, we augment the pose distribution during training. Specifically, with a probability of 0.5, we sample poses from a pre-defined distribution characterized by the standard deviation of yaw and pitch $(\sigma_{\text{yaw}}, \sigma_{\text{pitch}})$ of the FFHQ dataset. We define the yaw distribution as $N(0, \sigma_{\text{yaw}})$ and the pitch distribution as $U(-3\sigma_{\text{pitch}}, 3\sigma_{\text{pitch}})$, where U represents a uniform distribution. This oversampling strategy ensures better coverage of images with varied vertical perspectives.

A.2 Additional examples

We present additional examples generated by our method for the FFHQ and AFHQ-Cat datasets in Fig. 9, 10. Furthermore, we provide multi-view images synthesized from densely sampled camera positions in Fig. 11. Specifically, for each 3D scene, we sample yaw angles in the range of [-0.4, 0.4] radians and pitch angles in the range of [-0.4, 0.1] radians. These multi-view images demonstrate the 3D consistency of our method in a qualitative manner. For every visualization, we use truncation psi $\psi = 0.7$ and fix a camera condition as frontal view.

A.3 Latent interpolation

One of the notable characteristics of GANs is a semantic latent space. To validate that the proposed method also exhibits this characteristic, we synthesize images using linearly interpolated latent codes. As shown in Fig. 12, the interpolated images gradually transition from the given source to the target image, demonstrating smooth and coherent transformations. This indicates that our method maintains a meaningful and semantically rich latent space.

16

FFHQ-256   AFHQ-Cat-256
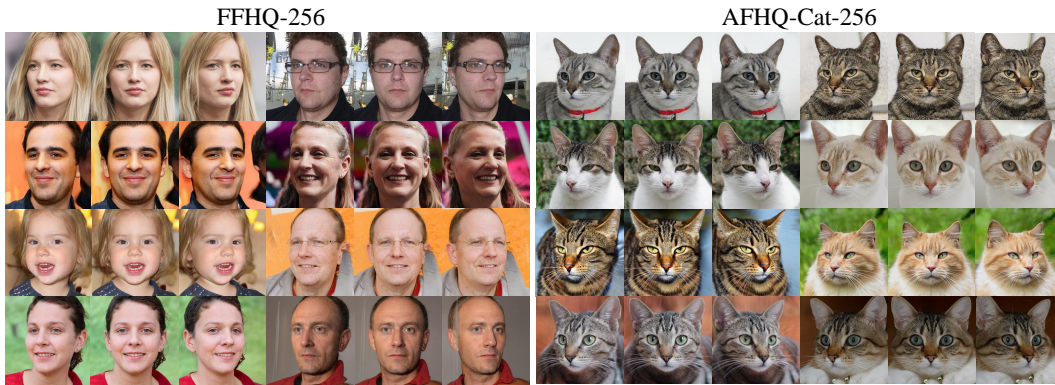
Figure 10: Generated examples on FFHQ-256 and AFHQ-Cat-256
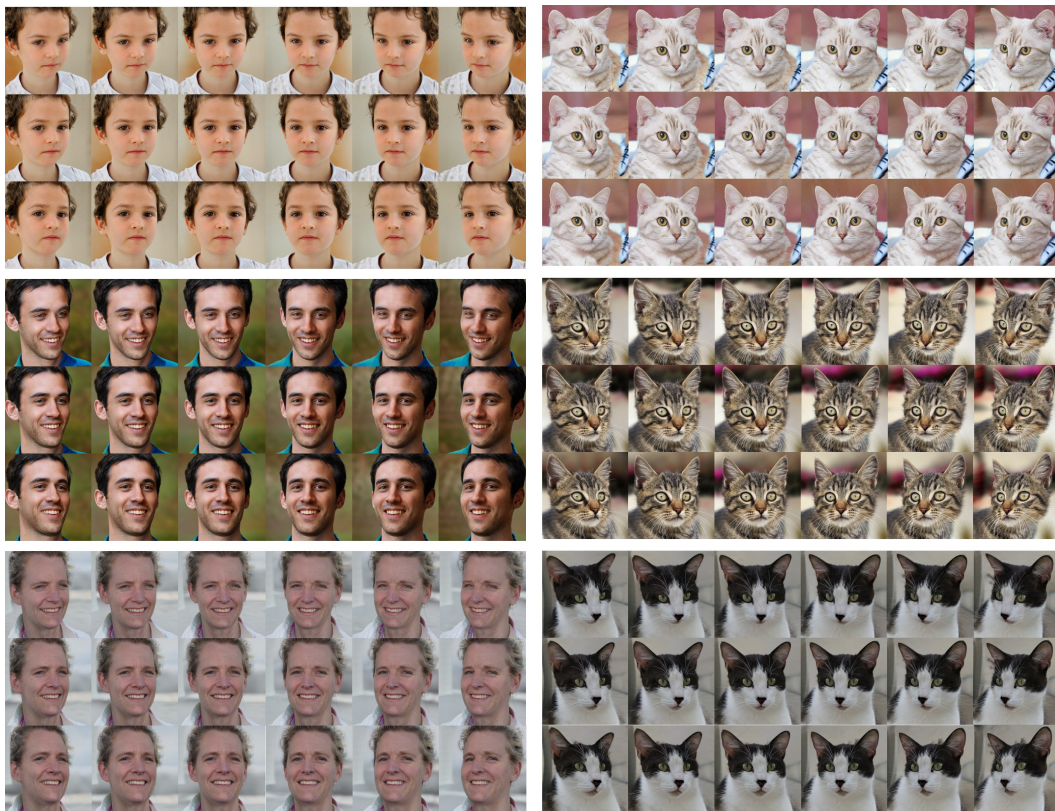
Figure 11: Examples of multi-view generation. We sample the camera pose of [-0.4, 0.4] radian for a yaw, and [-0.4, +0.1] for a pitch.

17
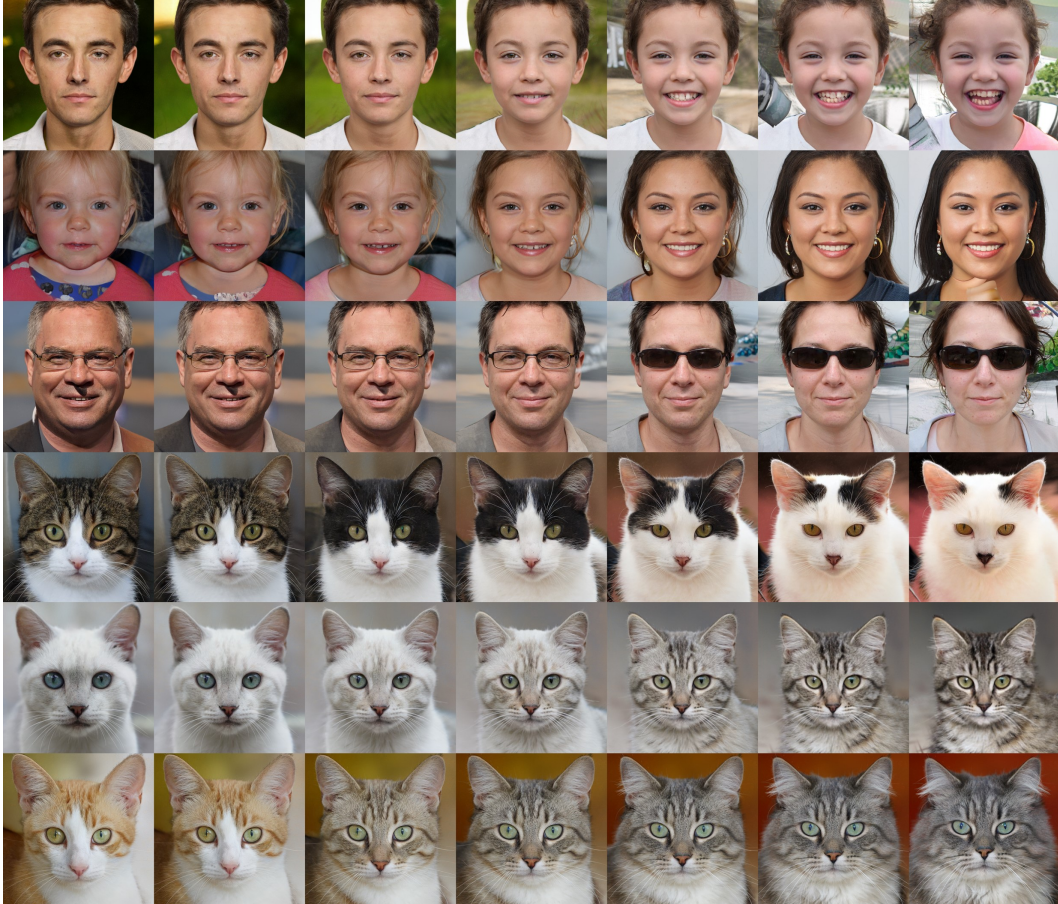
Figure 12: Linear interpolation between latent codes (w space)

A.4 w+ Inversion

We perform the w+ inversion [59] from a given single view in-the-wild image and predict novel view synthesis using the estimated 3D scene. As shown in Fig. 13, the proposed method successfully inverts the in-the-wild image and synthesizes novel views from it, demonstrating the method's ability to generalize and manipulate real-world images. We expect that recent inversion techniques, such as pivotal tuning [60], could further enhance the quality of the inversion.

A.5 Visualization of anchor Gaussians

We visualize the Gaussians and anchor Gaussians level-by-level in Fig. 14. Note that, we manually set the opacity of the anchors as sigmoid(1), as they do not have actual opacity for rendering. As visualized, we observe that the anchors typically have a larger scale compared to the Gaussians. We also verify it by calculating statistics of the scale of anchors and Gaussians. In detail, the average scale of anchors is $e^{-5.08}$ across 100 randomly sampled scenes, while the average scale of Gaussians is $e^{-7.12}$. We hypothesize that the Gaussians adjust their scales to represent sharp details in real-world images, whereas the anchor Gaussians are mainly used to constrain and regularize the parameters of the coarser-level Gaussians.

A.6 Effect of background generator

We visualize the Gaussians generated from the background generator in Fig. 15. On the left side, we synthesize the image with and without the background Gaussians. As visualized, the background Gaussians are responsible for the backgrounds, especially for the blurred area. For more visualization, we provide an image rendered by a camera far from the world-coordinate origin on the right side.

18

Target   Inverted result   Novel view synthesis
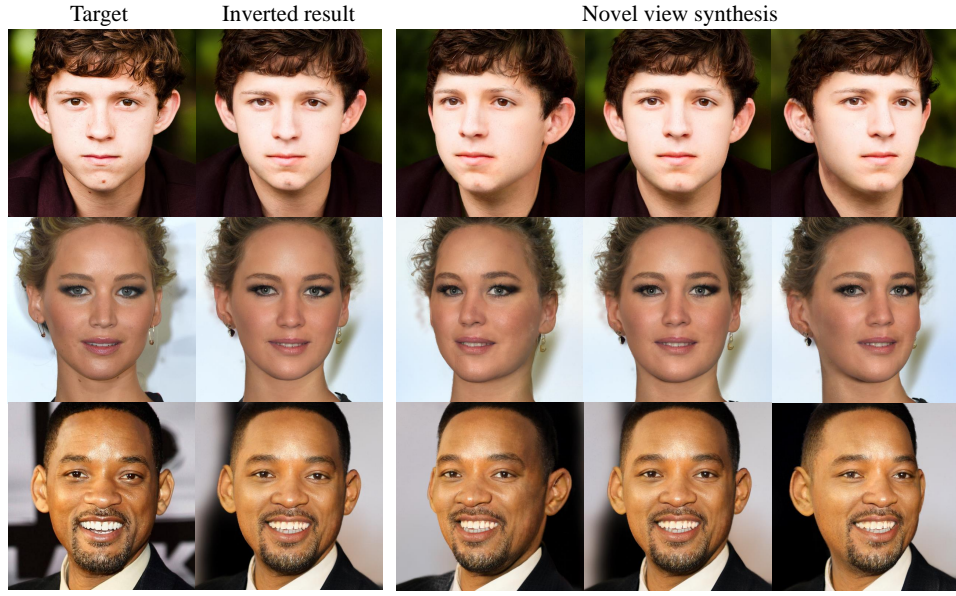
Figure 13: Novel view synthesis using w+ inversion.
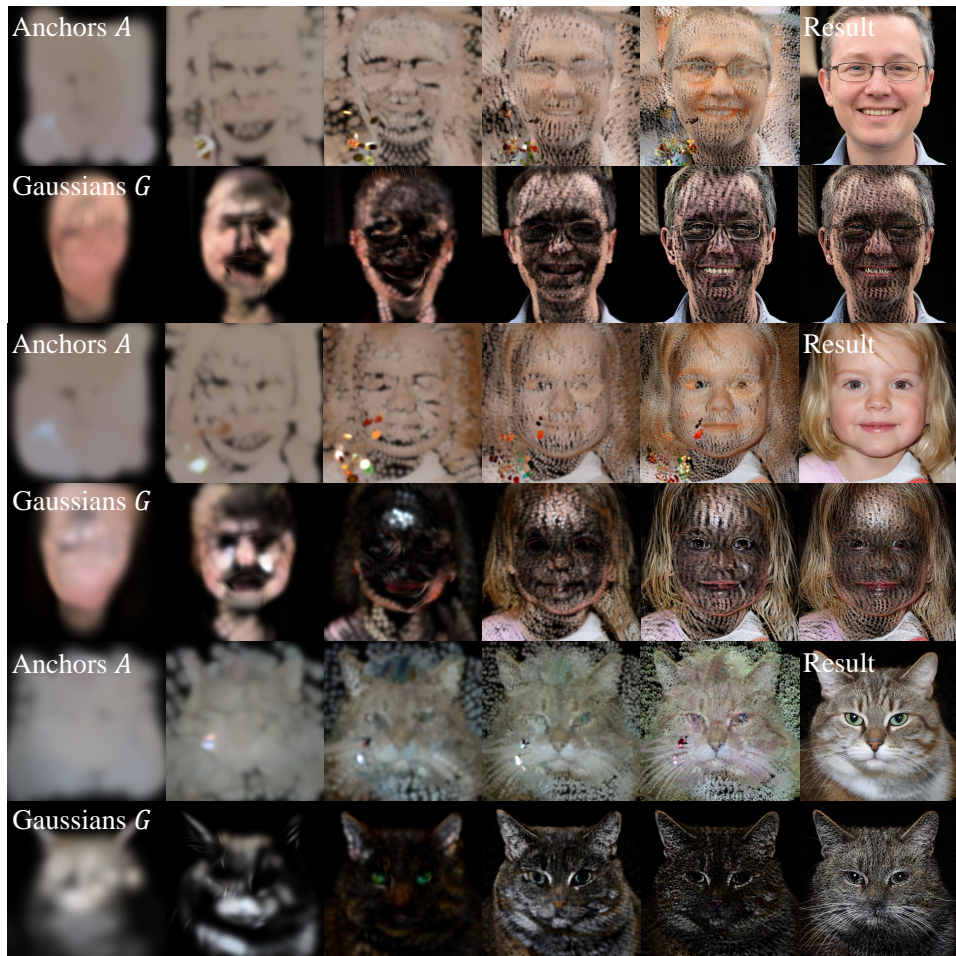
Figure 14: Visualization of Gaussians G and anchor Gaussians A for each level. We set opacity α of anchors as $\mathrm{sigmoid}(1)$ for visualization. From left-to-right, levels of Gaussians and anchors increase except right-top image, which is a composite of Gaussians from all levels.

19

w/ background   Visualization of background Gaussians
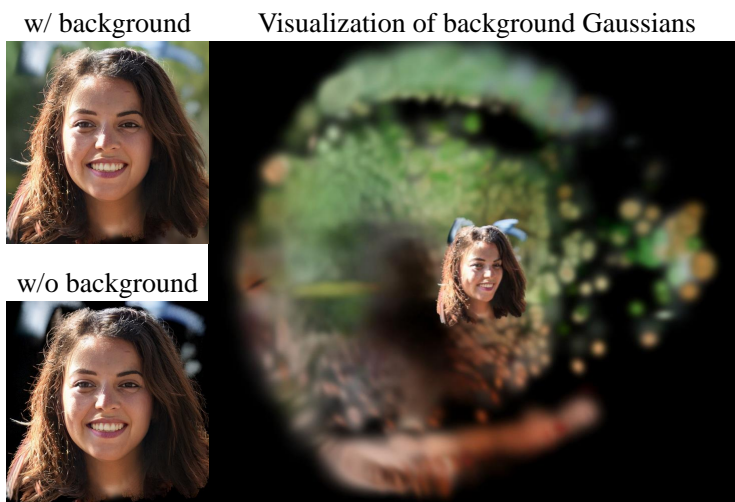
Figure 15: Visualization of effects of background generator

20

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We elaborate on our claims in both abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We refer to the limitation on sec. 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

21

Justification: This paper does not contain any theory assumptions.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We report the implementation details in sec. A.1 and reference of datasets in sec. 4.1, and also attach the implementation code in supplementary.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

22

Answer: [Yes]

Justification: We attach the implementation code in supplementary and will release it.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We report the detailed training and test details in appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Multiple runs of training are computationally too demanding in 3D GANs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

23

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We report the training resource in Sec. 4, which is in charge of most of the computational cost for the experiments in this paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We review the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We elaborate on the broader impacts in sec. 5.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

24

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

   Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

   Answer: [NA]

   Justification: The paper does not have such risks, as we do not use models and data concerned about any privacy issues.

   Guidelines:

   - The answer NA means that the paper poses no such risks.
   - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
   - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
   - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

   Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

   Answer: [Yes]

   Justification: We check datasets has license of CC BY-NC-SA 4.0 and CC-BY-NC 4.0.

   Guidelines:

   - The answer NA means that the paper does not use existing assets.
   - The authors should cite the original paper that produced the code package or dataset.
   - The authors should state which version of the asset is used and, if possible, include a URL.
   - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
   - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
   - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
   - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

25

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not provide new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

26