VoxGRAF: Fast 3D-Aware Image Synthesis with Sparse Voxel Grids

Katja Schwarz[1] **Axel Sauer**[1] **Michael Niemeyer**[1] **Yiyi Liao**[2] **Andreas Geiger**[1]

[1]University of Tübingen and Max Planck Institute for Intelligent Systems, Tübingen
[2] Zhejiang University, China

Abstract

State-of-the-art 3D-aware generative models rely on coordinate-based MLPs to parameterize 3D radiance fields. While demonstrating impressive results, querying an MLP for every sample along each ray leads to slow rendering. Therefore, existing approaches often render low-resolution feature maps and process them with an upsampling network to obtain the final image. Albeit efficient, neural rendering often entangles viewpoint and content such that changing the camera pose results in unwanted changes of geometry or appearance. Motivated by recent results in voxel-based novel view synthesis, we investigate the utility of sparse voxel grid representations for fast and 3D-consistent generative modeling in this paper. Our results demonstrate that monolithic MLPs can indeed be replaced by 3D convolutions when combining sparse voxel grids with progressive growing, free space pruning and appropriate regularization. To obtain a compact representation of the scene and allow for scaling to higher voxel resolutions, our model disentangles the foreground object (modeled in 3D) from the background (modeled in 2D). In contrast to existing approaches, our method requires only a single forward pass to generate a full 3D scene. It hence allows for efficient rendering from arbitrary viewpoints while yielding 3D consistent results with high visual fidelity. Code and models are available at https://github.com/autonomousvision/voxgraf.

1 Introduction

Generating photorealistic renderings of scenes at high resolution is a long-standing goal in computer vision and graphics. The primary paradigm is to carefully design 3D models, which are then rendered using realistic camera and illumination models. In recent years, the computer vision community has made significant headway towards reducing these design efforts by approaching content generation from a data-centric perspective. Generative Adversarial Networks (GANs) [10] have emerged as a powerful class of generative models for photorealistic high-resolution image synthesis [3, 19, 20, 22, 23, 39, 40]. One benefit of these 2D models is that they can be trained with large collections of images which are readily available. However, scaling GANs to 3D is non-trivial because 3D supervision is difficult to obtain. Recently, *3D-aware* GANs have emerged to address the gap between handcrafted 3D models and image synthesis with 2D GANs which lack 3D constraints [5, 15, 26, 32, 33, 41]. 3D-aware GANs combine 3D generators, differentiable rendering and adversarial training to synthesize novel images with explicit control over the camera pose and, potentially, other scene properties like object shape and appearance.

Early 3D-aware GANs explored voxel-based 3D representations [15, 32]. To compensate for the cubic memory growth of voxel grids, HoloGAN [32] generates features on a small 3D grid and uses a neural network to map 3D features to a 2D image. While such a *neural renderer* allows scaling to higher image resolutions, it may also entangle viewpoint and generated content [41].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

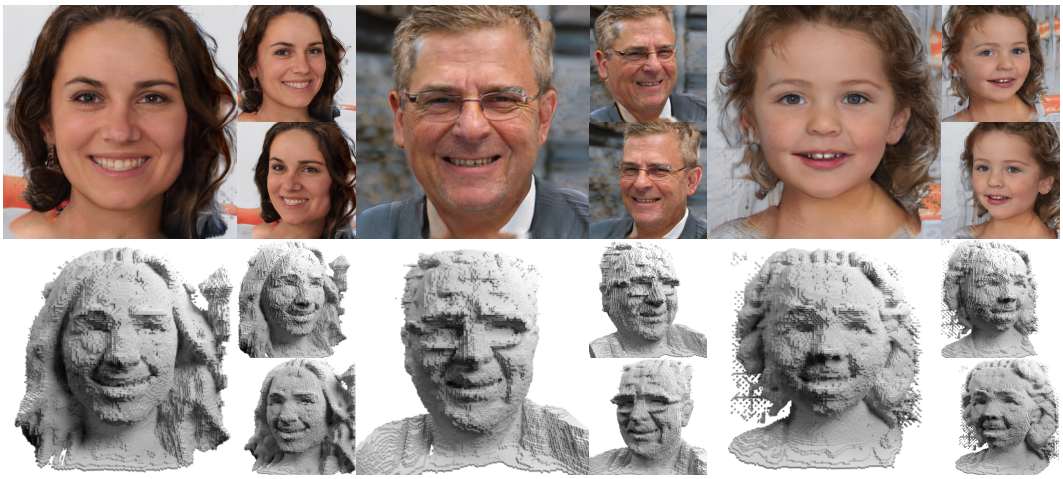


Figure 1: **3D-aware image synthesis with sparse voxel grids.** We investigate neural radiance fields represented as sparse voxel grids in the context of 3D-aware generative modeling. While training from unstructured image collections, our method allows for high quality image synthesis with explicit control over the camera viewpoint. In contrast to previous works, our method generates a 3D scene in a single forward pass allowing for more efficient and 3D-consistent rendering.

Consequently, early voxel-based approaches were either limited in image resolution or lacking 3D consistency. About the same time, Neural Radiance Fields (NeRF) [30] emerged in the context of view synthesis as a powerful alternative 3D representation. In their seminal work, Mildenhall et al. [30] represent a scene as a function of color and density, parameterized by a coordinate-based MLP. The predicted color and density values are then projected to an image with differentiable volume rendering. GRAF [42] adapts NeRF's coordinate-based representation to Generative Radiance Fields (GRAF) and proposes a 3D-aware GAN using a coordinate-based MLP and volume rendering. This propelled 3D-aware image synthesis to higher image resolutions while better preserving 3D consistency due to the physically-based and parameter-free rendering. These benefits led to the establishment of coordinate-based MLPs as new de facto standard for 3D-aware image synthesis [5, 34]. While recent 3D-aware GANs [5, 7, 11] have started to attain image fidelity and resolution similar to 2D GANs, training and inference is computationally expensive as the MLP must be queried at multiple points along each ray for volume rendering. However, querying 3D space densely is prohibitively costly. For example, rendering an image at resolution 256^2 using 48 sample points along each ray requires to query the neural network $256^2 \cdot 48 \approx 3M$ times. As a result, most recent works combine neural and volume rendering to ease computational cost at high resolutions [33]. Consequently, viewpoint and 3D content are often entangled, such that changing the camera pose might result in unwanted changes of the geometry or the appearance of the 3D scene. Further, for many downstream applications, e.g. integrating assets into physics engines, it is desirable to generate 3D content at high resolution directly. These shortcomings identify the need for a 3D representation that can be efficiently rendered at high resolution with a model that is *3D-consistent by design*.

Recently, there has been significant progress towards accelerated training of novel view synthesis models for single scenes by removing the MLP from the representation. In particular, DVGO [44] and Plenoxels [1] directly optimize a sparse voxel grid for novel-view synthesis, demonstrating that the visual fidelity attained by NeRF [30] is not primarily attributed to its MLP-based representation but rather to volumetric rendering and gradient-based optimization. In addition to impressive image quality, [1, 44] obtain large rendering speedups due to their fast density and color queries. Taking inspiration from these works, we revisit voxel-based representations for 3D-aware GANs [15, 32] in the context of volumetric rendering. To circumvent the cubic memory growth that limits early voxel-based approaches [15, 32], we explore *sparse* voxel grids for the generative settings, see Fig. 1. We observe that sparsity is key to enable scaling the 3D representation to higher resolution and to combine it with volume rendering. Specifically, we propose a 3D-aware GAN with a sparse voxel grid generator at its core. As a result, our approach inherits fast rendering and trilinear interpolation while being 3D-consistent by design, separating it from other recent 3D-aware GANs [4, 7, 11] which require a forward pass for every point along each camera ray of every view. Another difference to

existing 3D-aware GANs is that sparsity is a built-in feature of our representation, mitigating the need for exploiting sophisticated strategies to sample points along camera rays as in [5, 37, 47]. Our final model achieves image fidelity similar to recent 3D-aware GANs leveraging neural rendering while generating high-resolution geometry and improving 3D consistency. During inference, our model only requires a single forward pass which takes up most of the inference time. Once the 3D scene is generated, images can be rendered within milliseconds while existing approaches require another forward pass which is two orders of magnitude slower. We refer to our model as *VoxGRAF*.

2 Related Work

2D GANs. Rapid progress on Generative Adversarial Networks [10] now enables photorealistic synthesis up to megapixel resolution [3, 19, 21–23, 29, 40]. While the disentangled style-space of StyleGANs [21–23] allows for control over the viewpoint of the generated images to some extent [14, 27, 43, 52], gaining precise 3D-consistent control is still non-trivial due to its lack of physical interpretation and operation in 2D. In contrast, in this work we aim for explicit control over the camera pose by incorporating a 3D representation into the generator.

3D-Aware GANs. The first 3D-aware GANs, i.e. GANs that incorporate a 3D representation into the generator model, were voxel-based approaches. Dense grid-based approaches [15, 54] are limited to a lower grid resolution due to their cubic memory growth. Other works combine lower-resolution grids with neural rendering [26, 32] which scale to higher resolutions, but the generated images lack 3D consistency [42]. Recently, GRAF [41] and π-GAN combine volume rendering and coordinate-based representations [30] allowing to scale 3D-aware GANs with physically inspired rendering to high resolutions. However, dense ray marching remains computationally expensive and limits image fidelity. GIRAFFE [34] therefore proposes a hybrid rendering approach. They render a low-resolution feature map with ray marching and use a neural renderer to decode it into a high-resolution image. Due to efficiency, this approach has been widely adopted in subsequent works [4, 11, 18, 36, 48, 51, 53]. While some approaches try to counteract introduced inconsistencies, e.g. via dual discrimination [4] or a reconstruction loss [11], we instead propose a model that is *3D-consistent by design* and can generate the 3D object at high-resolution.

As an alternative to hybrid rendering, GOF [47], ShadeGAN [37] and GRAM [7] focus on reducing the number of query points for volume rendering. While aforementioned methods aim for reducing the number of sample points as querying a large MLP is computationally expensive, we instead use a sparse voxel grid as 3D representation. This allows us to speed up rendering without reducing the sample size as feature querying via trilinear interpolation is fast and can be efficiently implemented via custom CUDA kernels.

Sparse 3D Representations. As NeRF requires an optimization time in the order of multiple days per scene, a series of follow-up works [28, 38, 45, 50] propose techniques to speed up this process. The recent works Plenoxels [1] and DVGO [44] demonstrate that sparse voxel grid representations can achieve even faster convergence and higher rendering speed. In addition to efficient rendering, sparse voxel grids enable fast trilinear interpolation when queried beyond their grid resolution. Building on this representation, our approach inherits these benefits. We remark that very recently Instant-NGP [31] achieves even faster rendering by combining small MLPs with a multi-resolution hash table. Exploring this representation might be an interesting avenue for future extensions of our work. Note that all of these works focus on novel-view-synthesis for single scenes and require multi-view image supervision. Instead, we propose a generative model that trains with raw image collections and that can generate multiple novel instances at inference.

3 Method

We first provide the necessary background by summarizing the currently dominating paradigm for designing 3D-aware GANs which combines an MLP scene representation and volume rendering as introduced in GRAF [42]. Next, we introduce our sparse voxel-based scene representation which boosts rendering speed while retaining 3D-consistency by design. We refer to our model as VoxGRAF.

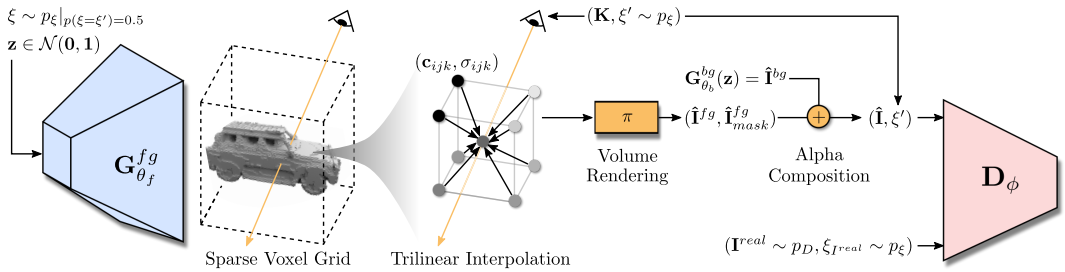


Figure 2: **VoxGRAF.** Conditioned on a camera pose ξ, the foreground generator $\mathbf{G}^{fg}_{\theta_f}$ maps a latent code $\mathbf{z}$ to color values $\mathbf{c} \in \mathbb{R}^{3 \times R_G \times R_G \times R_G}$ and densities $\sigma \in \mathbb{R}^{1 \times R_G \times R_G \times R_G}$ on a sparse voxel grid of resolution R_G. Given camera intrinsics $\mathbf{K}$ and a camera pose ξ', a foreground image is obtained using differentiable volume rendering [30]. The values at the sampling points along the camera rays are computed by trilinearly interpolating the voxel grid [1]. The background is generated by a 2D GAN $\mathbf{G}^{bg}_{\theta_b}$ and combined with the foreground using alpha composition. The discriminator D_ϕ compares the generated image $\hat{\mathbf{I}}$ to the real image $\mathbf{I}^{real}$.

3.1 3D-aware GANs with Coordinate-Based Scene Representations

Recent 3D-aware GANs typically consist of a learned coordinate-based MLP as 3D generator, a deterministic volume rendering step potentially combined with a learned neural renderer in the 2D image domain, and a learned 2D discriminator. Let G^{MLP}_ψ denote the 3D generator parameterized by a coordinate-based MLP with learnable parameters ψ. The 3D generator predicts a radiance field defined by color $\mathbf{c} \in [0, 1]$ and density values $\sigma \in \mathbb{R}^+$. The radiance field is defined at any 3D point $\mathbf{x} \in \mathbb{R}^3$ and for any viewing direction $\mathbf{d} \in \mathbb{S}^2$. To model different 3D scenes, the generator is additionally conditioned on an M-dimensional latent variable $\mathbf{z} \in \mathcal{N}(\mathbf{0}, \mathbf{1})$. The inputs $\mathbf{x}$ and $\mathbf{d}$ are projected to higher-dimensional features $\gamma(\mathbf{x}) \in L_\mathbf{x}$ and $\gamma(\mathbf{d}) \in L_\mathbf{d}$ with a fixed positional encoding to overcome the MLP's smoothness bias and model high-frequency content $\gamma(\cdot)$ [30, 46]. Formally,

$$G^{MLP}_\psi : \mathbb{R}^{L_\mathbf{x}} \times \mathbb{R}^{L_\mathbf{d}} \times \mathbb{R}^M \to \mathbb{R}^3 \times \mathbb{R}^+ \qquad (\gamma(\mathbf{x}), \gamma(\mathbf{d}), \mathbf{z}) \mapsto (\mathbf{c}, \sigma) \tag{1}$$

In this paper, we challenge this paradigm and investigate a sparse 3D CNN instead of a coordinate-based MLP as generator, as described in the next section.

The radiance field is rendered by approximating the intractable volumetric projection integral via numerical integration. First, the generator is queried at N sampling points along each camera ray r yielding colors and densities $\{(\mathbf{c}^i_r, \sigma^i_r)\}_{i=1}^N$. For each camera ray r, these points are projected to an RGB color value $\mathbf{c}_r$ and optionally an alpha mask $\mathbf{a}_r$ using alpha composition

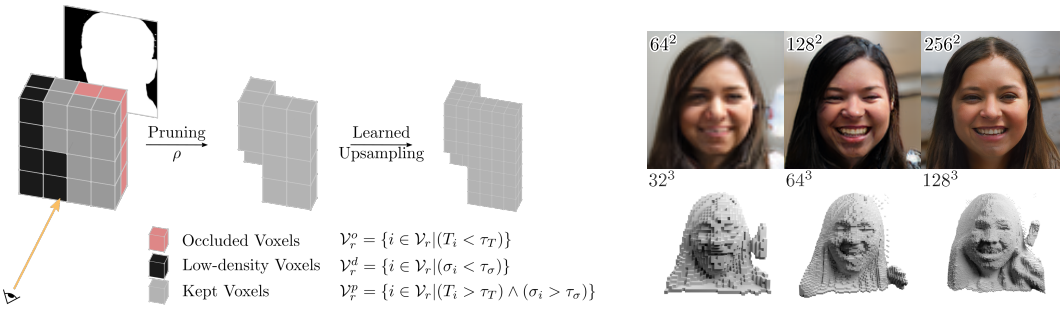
$$\pi : (\mathbb{R}^3 \times \mathbb{R}^+)^N \to \mathbb{R}^3 \qquad \{(\mathbf{c}^i_r, \sigma^i_r)\} \mapsto \mathbf{c}_r$$

$$\mathbf{c}_r = \sum_{i=1}^N T^i_r \, \alpha^i_r \, \mathbf{c}^i_r \qquad \mathbf{a}_r = \sum_{i=1}^N T^i_r \, \alpha^i_r \qquad T^i_r = \prod_{j=1}^{i-1} \left(1 - \alpha^j_r\right) \qquad \alpha^i_r = 1 - \exp\left(-\sigma^i_r \delta^i_r\right) \tag{2}$$

where T^i_r and α^i_r denote the transmittance and alpha value of sample point i along ray r and $\delta^i_r = \left\|\mathbf{x}^{i+1}_r - \mathbf{x}^i_r\right\|_2$ is the distance between neighboring sample points. As volume rendering has proven a powerful tool for high-fidelity reconstruction, VoxGRAF retains this rendering mechanism but reduces its computational cost by leveraging sparse scene representations.

3.2 VoxGRAF: Generating Radiance Fields on Sparse Voxel Grids

Our goal is to design a 3D-aware GAN based on a sparse scene representation that allows for efficient rendering. Fig. 2 shows an overview over our approach. In contrast to recent works [4, 7, 47], we do not use a coordinate-based MLP to parameterize the radiance field. Instead, inspired by recent work on novel view synthesis [1, 44], we generate values on a sparse voxel grid using a 3D convolutional neural network. Therefore, our generator requires only a single forward pass to generate a 3D scene. To disentangle 3D content from the background we combine a 3D foreground generator $\mathbf{G}^{fg}_{\theta_f}$ with a 2D background generator $\mathbf{G}^{bg}_{\theta_b}$. $\mathbf{G}^{fg}_{\theta_f}$ takes a camera matrix $\mathbf{K}$, camera pose ξ and a latent code $\mathbf{z}$



(a) Pruning (b) Progressive Growing

Figure 3: **Pruning and Progressive Growing.** A sparse representation is key for scaling voxel grids to high resolution. To sparsify the generated voxel grids, we combine density-based pruning (a) and progressive growing (b) while training our model. See text for details.

as input and predicts colors $\mathbf{c}$ and density values σ on a sparse voxel grid. For unstructured images, camera matrix $\mathbf{K}$ and pose ξ can be determined e.g. with an off-the-shelf pose detector as done in [5]. Volume rendering yields a foreground image and an alpha mask. The background generator maps the latent code $\mathbf{z}$ to a background image which is then combined with the foreground image using alpha composition. We train our model in an adversarial setting using a 2D discriminator on the full image.

Foreground Generator. Our foreground generator builds on the popular StyleGAN2 architecture [22] which achieves high fidelity in the 2D image domain. Conditioned on a camera pose $\xi \in \mathbb{R}^P$, it maps a latent code $\mathbf{z}$ to color values $\mathbf{c}$ and density values σ on a sparse voxel grid

$$\mathbf{G}_{\theta_f}^{fg}: \quad \mathbb{R}^M \times \mathbb{R}^P \rightarrow \mathbb{R}^{3\times R_G \times R_G \times R_G} \times \mathbb{R}^{1\times R_G \times R_G \times R_G} \quad (\mathbf{z}, \xi) \mapsto (\mathbf{c}, \sigma) \tag{3}$$

where θ_f and R_G denote the learnable parameters and the resolution of the voxel grid, respectively. Note that in contrast to most existing coordinate-based 3D-aware GANs, see Eq. (1), we do not condition the 3D generator on the view direction (per-ray). Instead, we follow [4] and condition it on the pose $\boldsymbol{\xi}$ (per-image) to model directional dependencies. For rendering, we compute $(\mathbf{c}_r^i, \sigma_r^i)$ via trilinear interpolation of densities and colors stored at the nearest eight vertices [1]. We use the same rendering formulation outlined in 3.1 and leverage custom CUDA kernels[1] for efficiency.

To generate voxel grids instead of images, we replace all 2D operations of the StyleGAN2 generator with their 3D equivalent, e.g., 3D modulated convolutions and 3D upsampling. At resolutions beyond 32^3, we investigate sparse convolutions [13] instead of dense ones as they can be more computationally efficient. We compare the computational efficiency of sparse convolutions[2] to dense convolutions for which we zero out the values of pruned voxels. For our architecture, using sparse convolutions reduces the memory consumption but due to the computational overhead for managing coordinates increases the runtime, see Table 1. To sparsify the representation, we combine progressive growing [19] with pruning [1] as illustrated in Fig. 3a. Specifically, we start with training a dense model at resolution 32^3. After sufficient training, we add the next layer of convolutions and prune its inputs based on the rendered view. Intuitively, the next layer should only operate on voxels visible in the rendered view to yield a sparse representation. Consequently, we prune voxels that are either occluded or have a low density. Following Eq. (2), the rendered alpha value is $\mathbf{a}_r = \sum_{i=1}^{N} T_r^i \, \alpha_r^i$. Accordingly, occluded voxels (low transmittance T_i) and empty voxels (low density σ_i) do not contribute to the final image. The pruning operator ρ discards all voxels along a camera ray r for which transmittance T^i or density σ_i is smaller than threshold τ_T or τ_σ, respectively. Let $\mathcal{V}_r$ denote the set of voxels intersecting with ray $\mathbf{r}$, then

$$\rho : \mathcal{V}_r \mapsto \mathcal{V}_r^p \qquad \mathcal{V}_r^p = \{i \in \mathcal{V}_r | \, (T_i > \tau_T) \wedge (\sigma_i > \tau_\sigma)\} \tag{4}$$

where $\mathcal{V}_r^p$ is the set of the retained voxels. After pruning, we upsample the kept voxels via sparse transposed convolutions. After the newly-added layer is sufficiently trained, we repeat this process for the next added stage. Fig. 3b shows images and voxel grids at different resolutions. We implement this on-the-fly pruning operation using efficient custom CUDA kernels.

[1] We build on the kernels from https://github.com/sxyu/svox2.git

[2] We use the Minkowski Engine library https://github.com/NVIDIA/MinkowskiEngine.git

In agreement with [4], we observe that it is crucial to account for view dependent effects or pose-correlated attributes in the training data, like eyes always looking into the camera. We take two measures to increase the flexibility of our model: Following [4], $\mathbf{G}_{\theta_f}^{fg}$ is conditioned on the pose ξ which corresponds to the rendering pose ξ' in 50% of the cases and is randomly chosen otherwise. Formally, $\xi \sim p_\xi|_{p(\xi=\xi')=0.5}$. At inference, the pose-conditioning is fixed to retain 3D consistency. However, we find that pose conditioning alone is not always sufficient to account for strong correlations in the data. Depending on the dataset, we optionally refine the rendered image with a shallow 2D CNN with 2 hidden layers of dimension 16 and kernel size 3. While this refinement is powerful enough to model dataset biases, it is considerably less flexible than the neural rendering used in [5, 11, 33]: Our 2D CNN operates on the rendered image instead of rendered features at smaller resolution. Operating on 3 channels at full resolution allows to keep its capacity at a minimum and, due to not using any upsampling operation, results in a local receptive field.

Background Generator. We consider datasets with a single object per image. Modeling only the object in 3D saves computation and is advantageous for potential downstream tasks, e.g., integrating generated assets into new environments Hence, we model the object in 3D and generate the background of the image with a 2D GAN. Specifically, we use the StyleGAN2 generator [22] with reduced channel size as modeling the background requires less capacity than generating the full image:

$$\mathbf{G}_{\theta_b}^{bg}: \quad \mathbb{R}^M \rightarrow \mathbb{R}^{3\times R_I \times R_I} \quad \mathbf{z} \mapsto \hat{\mathbf{I}}^{bg} \tag{5}$$

We choose the same latent code $\mathbf{z}$ for foreground and background to allow for modeling correlations like lighting between the two. Note that, unlike the foreground generator, the background generator is not conditioned on the camera pose. As the background remains fixed when changing the camera pose, the generator is encouraged to model pose-dependent content with the foreground generator leading to disentanglement (see supp. mat. for qualitative disentanglement results).

The final image is obtained using alpha composition

$$\hat{\mathbf{I}} = \hat{\mathbf{I}}_{mask}^{fg} \cdot \hat{\mathbf{I}}^{fg} + (1 - \hat{\mathbf{I}}_{mask}^{fg}) \cdot \hat{\mathbf{I}}^{bg}$$

The full generator is defined as

$$\mathbf{G}_\theta: \quad \mathbb{R}^M \times \mathbb{R}^P \times \mathbb{R}^P \times \mathbb{R}^K \rightarrow \mathbb{R}^{3\times R_I \times R_I} \quad (\mathbf{z}, \xi, \xi', \mathbf{K}) \mapsto \hat{\mathbf{I}} \tag{6}$$

Regularization. For fast rendering, it is crucial that most generated voxels are either fully opaque or empty such that early stopping and empty space skipping are effective. By regularizing the variance of the expected depth $\hat{z}_r$ along each ray r, the foreground generator is encouraged to generate a single, sharp surface:

$$\hat{z} = \sum_i T_i \alpha_i z_i \qquad Var(\hat{z}) = \frac{1}{\sum_i T_i \alpha_i} \sum_j T_j \alpha_j (z_j - \hat{z})^2 \tag{7}$$

$$\mathcal{L}_{DV} = \lambda_{DV} \max(Var(\hat{z}), \tau) \tag{8}$$

where τ is a hyperparameter that defines the thickness of the surface. We find that thresholding the loss is important to avoid an empty foreground. In addition, we find that adding the grid TV regularization $\mathcal{L}_{TV}$ from [1] and fore- and background coverage regularization $\mathcal{L}_{cvg}^{fg}$ and $\mathcal{L}_{cvg}^{bg}$ from [49] further stabilizes training (see sup. mat. for details). The full regularization term of our generator is

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{DV} + \mathcal{L}_{TV} + \mathcal{L}_{cvg}^{fg} + \mathcal{L}_{cvg}^{bg} \tag{9}$$

Discriminator. We use the StyleGAN2 discriminator and condition it on the camera pose as proposed in [4]. Similarly, we find that conditioning guides the generator to learn correct 3D priors and a canonical representation. Since rendering our sparse representation is fast, our discriminator is able to operate on the full image and does not need to consider image patches as done in GRAF [42].

3.3 Training

Given images $\mathbf{I}$ from the data distribution $p_\mathcal{D}$ with known camera extrinsics $\boldsymbol{\xi}_\mathbf{I}$ and intrinsics $\mathbf{K}_\mathbf{I}$ and latent codes $\mathbf{z} \in \mathcal{N}(\mathbf{0}, \mathbf{1})$, we train our model using a GAN objective with R1-regularization [29]

$$V(\theta, \phi) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{1}),\xi,\xi'\sim p_\xi} \left[f(-D_\phi \left(G_\theta(\mathbf{z}, \boldsymbol{\xi}, \boldsymbol{\xi}', \mathbf{K}), \boldsymbol{\xi}' \right)) \right] \tag{10}$$

$$+ \ \mathbb{E}_{\mathbf{I}\sim p_\mathcal{D}} \left[f(D_\phi(\mathbf{I}, \boldsymbol{\xi}_\mathbf{I})) \ - \ \lambda \|\nabla D_\phi(\mathbf{I}, \boldsymbol{\xi}_\mathbf{I})\|^2 \right] \tag{11}$$

where $f(t) = -\log(1 + \exp(-t))$ and λ controls the strength of the R1-regularizer. G_θ and D_ϕ are trained with alternating gradient descent combining the GAN objective with the regularization terms:

$$\min_\theta \max_\phi \qquad V(\theta, \phi) + \mathcal{L}_{reg}(\theta) \tag{12}$$

In practice, we optimize the generator with a non-saturating variant of Eq. (12) [10]. We train our approach with Adam [24] using a batch size of 64 at grid resolution $R_G = 32$, 64 and 32 at $R_G = 128$. We use a learning rate of 0.0025 for the generator and 0.002 for the discriminator. For faster training, we first grow R_I from 32 to 128 while keeping R_G at 32. Then, we alternately increase the grid resolution and the image resolution until the dataset resolution and $R_G = 128$ are reached. For synthetic datasets, i.e. Carla [42], we do not add any refinement layers. Depending on the dataset, we train our models for 3 to 7 days on 8 Tesla V100 GPUs. Details on the network architectures can be found in the supplemental material.

4 Results

Datasets. We validate our approach on standard benchmark datasets for 3D-aware image synthesis. The synthetic Carla dataset [9, 41] contains 10k images and camera poses of 18 car models with randomly sampled colors. FFHQ [22] comprises 70k aligned face images. AFHQv2 Cats [6] consists of 4834 cat faces. Following [5], we estimate camera poses for both datasets with off-the-shelf pose estimators [8, 25] and augment all datasets with horizontal flips. Due to the limited number of images in AFHQv2 and Carla, we use adaptive discriminator augmentation [20] for these datasets.

Evaluation Metrics. We measure image fidelity by calculating the Fréchet Inception Distance (*FID*) [16] between 20k generated images and the full dataset. For all runtime comparisons, we report times on a single Tesla V100 GPU with a batch size of 1.

4.1 Ablation Study

We investigate the sparsity of the generated voxel grids and validate the importance of the depth variance loss, see Eq. (8). Sparsity is evaluated by fusing the pruned voxel grids from 16 equally spaced camera views and reporting the number of empty voxels divided by the total number of voxels R_G^3. Table 1 shows results averaged over 100 instances. The depth variance loss increases sparsity from 74% to 95%, lowering the memory consumption. With the sparser representation, the times needed for scene generation $t_{G^{fg+bg}}$ and rendering t_π are reduced significantly. We further compare an implementation with dense instead of sparse convolutions where we zero out the values of pruned voxels. While this increases memory consumption, it reduces the runtime due to the computational overhead of managing coordinates for sparse convolutions. We prioritize faster training over memory and hence train our models with dense convolutions where we set values of pruned voxels to zero.

	Sparsity [%]	Memory [GB]	$t_{G^{fg+bg}}$ [ms]	t_π [ms]
w/o $\mathcal{L}_{\mathcal{DV}}$	74	1.4	229	7
w $\mathcal{L}_{\mathcal{DV}}$	95	0.9	198	4
w $\mathcal{L}_{\mathcal{DV}}{}^\dagger$	95	1.1	58	4

Table 1: **Regularization.** We compare sparsity and rendering time with and without depth variance regularization $\mathcal{L}_{\mathcal{DV}}$ on FFHQ with $R_G = 128$ and $R_I = 128$. † denotes an architecture with dense convolutions where values of pruned voxels are set to zero.

4.2 Baseline Comparison

Baselines. We group the baselines into two categories: (i) methods that render low-resolution features and use 2D-upsampling, i.e. a neural renderer, to obtain the final image, e.g. StyleNeRF [11], and (ii) methods that render the 3D representation directly at the final image resolution, e.g. π-GAN [5]. VoxGRAF falls into the second group. We use the official code release for all methods. Further, we reference numbers reported by concurrent works GRAM [7] and EG3D [4].

Qualitative Results. Fig. 4 shows samples from multiple views for StyleNeRF, GRAM and VoxGRAF on FFHQ at resolution 256^2. StyleNeRF can generate additional faces or strands of hair across views, as indicated by the red frames in Fig. 4. These inconsistencies under viewpoint changes

Figure 4: **Qualitative Comparison on Multi-View Consistency**. We compare multi-view consistency of state-of-the-art baselines and our method. While StyleNeRF's powerful neural renderer can introduce multi-view inconsistencies (appearing faces, moving strands of hair), GRAM's manifold representation becomes visible in layered artifacts for steeper viewing angles. In contrast, our method leads to more multi-view consistent results.

	FFHQ [22] $R_I = 256^2$	AFHQ [6] $R_I = 256^2$	Carla [42] $R_I = 128^2$
GIRAFFE [34]	31.5	16.1	–
VolumeGAN [48]	9.1	–	7.9
StyleNeRF [11]	8.0	–	–
EG3D [4]	4.8	3.9	–
GRAF [42]	71	121	41
π-GAN [5]	85	47	29.2
GOF [47]	69.2	54.1	29.3
GRAM [7]	17.9	18.5	26.3
VoxGRAF	9.6	9.6	6.7

Table 2: **Quantitative Comparison.** We compare against state-of-the-art methods with a neural rendering pipeline (first block) and without one (second block). We report FID [16] between 20k generated images and the full dataset.

	$R_I = 128^2$	$R_I = 256^2$
GIRAFFE [33]	–	5
StyleNeRF [11]	–	49
EG3D* [4]	–	27
GRAF [42]	219	878
π-GAN [5]	154	608
GOF [47]	199	742
GRAM [7]	136	418
VoxGRAF	58 + 3	58 + 6

Table 3: **Rendering times**. We report time in ms per image. Note that our method allows for separating scene generation (first number) and rendering (second number) which is useful for real-time rendering applications. *EG3D is evaluated on a faster GPU (RTX 3090 GPU) compared to the others (Tesla V100 GPU).

are introduced by StyleNeRF's powerful neural renderer. GRAM achieves more consistent results, but its plane representation creates stripe artifacts for large viewpoint ranges. Due to the shallow 2D CNN, VoxGRAF can model the dataset bias of eyes looking into the camera but otherwise achieves high 3D-consistency even under large viewing angles. Additional samples of our method and the corresponding sparse voxel grids for all datasets are provided in Fig. 1 and Fig. 5.

Quantitative Results. Table 2 reports FID on all datasets. As expected, methods that use neural rendering with upsampling, i.e., StyleNeRF and EG3D, perform best in terms of image fidelity. This is expected as a neural renderer can add flexibility. But, as shown in Fig. 4, for StyleNeRF it reduces 3D-consistency. Among methods without a neural renderer, VoxGRAF significantly improves over π-GAN and GOF and surpasses the current state-of-the-art approach GRAM.

Figure 5: **Qualitative Results for our Method.** We show generated images at resolution 256^2 for FFHQ [22] and AFHQ [6] and samples at resolution 128^2 for Carla [42].

Runtime Comparison. Lastly, we compare the rendering times at inference for methods with and methods without a neural renderer. In contrast to all baselines, VoxGRAF requires only a single forward pass to generate the scene, which can then be rendered from different viewpoints efficiently. Note that at inference, voxels are pruned solely based on their density to amortize the rendering costs per scene. We find that this does not visibly affect the rendered images. We report the times for generating the scene and rendering one view separately in Table 3. One of the earliest works, GIRAFFE, is the fastest among all approaches as it renders a low-resolution feature volume and uses comparably small neural networks. StyleNeRF significantly increases the neural renderer's size to improve image fidelity, which comes at the cost of speed compared to GIRAFFE. Yet, StyleNeRF is the fastest approach for generating a single image among the best-performing methods. However, a potential application of 3D-aware GANs is generating novel views of a single instance in real-time. In this setting, at resolution 256^2, VoxGRAF generates novel views at 167 FPS, whereas StyleNeRF runs at 20 FPS as its rendering costs are not amortized per scene.

5 Limitations and Discussion

In this work, we investigate sparse voxel grids as representation for 3D-aware image synthesis. We find that the key to generating sparse voxel grids is to combine progressive growing, pruning, and regularization to encourage a sharp surface that can be rendered efficiently. Our approach outperforms all methods that do not employ a neural renderer. Instead of discarding neural rendering entirely, we find it advantageous to utilize a shallow CNN for refinement. This CNN can model dataset bias but is significantly weaker than standard neural rendering approaches that upsample low resolution feature maps. Our approach can reduce the gap to models that build heavily on neural rendering, yet a trade-off between 3D-consistency and image fidelity remains. Whether a certain amount of neural rendering is inherently needed to reach best performance is an important direction for future research. Lastly, the speed of our method depends on the sparsity of the modeled scene. Therefore, rendering times will likely increase on more complex datasets than those commonly used in literature.

Acknowledgments and Disclosure of Funding

We acknowledge the financial support by the BMWi in the project KI Delta Learning (project number 19A19013O), the support from the BMBF through the Tuebingen AI Center (FKZ:01IS18039A), and the support of the DFG under Germany's Excellence Strategy (EXC number 2064/1 - Project number 390727645). Andreas Geiger and Michael Niemeyer were supported by the ERC Starting Grant LEGO-3D (850533). We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Katja Schwarz and Michael Niemeyer. This work was supported by an NVIDIA research gift. We thank Christian Reiser for the helpful discussions and suggestions. Lastly, we would like to thank Nicolas Guenther for his general support.

References

[1] Alex Yu and Sara Fridovich-Keil, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 4, 5, 6, 14, 15

[2] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD gans. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. 19

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019. 1, 3

[4] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 4, 5, 6, 7, 8, 14, 15, 16, 18, 19

[5] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. Pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 3, 5, 6, 7, 8, 15, 16

[6] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 7, 8, 9, 15, 16, 19, 21

[7] Y. Deng, J. Yang, J. Xiang, and X. Tong. Gram: Generative radiance manifolds for 3d-aware image generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 4, 7, 8, 15, 16, 19

[8] Y. Deng, J. Yang, S. Xu, D. Chen, Y. Jia, and X. Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019. 7

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017. 7

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 1, 3, 7

[11] J. Gu, L. Liu, P. Wang, and C. Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. *Proc. of the International Conf. on Learning Representations (ICLR)*, 2022. 2, 3, 6, 7, 8, 15, 16, 19

[12] J. Gwak, C. B. Choy, A. Garg, M. Chandraker, and S. Savarese. Weakly supervised generative adversarial networks for 3d reconstruction. *arXiv.org*, 1705.10904, 2017. 14

[13] J. Gwak, C. B. Choy, and S. Savarese. Generative sparse detection networks for 3d single-shot object detection. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 5

[14] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris. Ganspace: Discovering interpretable GAN controls. *arXiv.org*, 2020. 3

[15] P. Henzler, N. J. Mitra, , and T. Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2, 3

[16] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 7, 8, 16

[17] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole. Zero-shot text-guided object generation with dream fields. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 14

[18] K. Jo, G. Shim, S. Jung, S. Yang, and J. Choo. Cg-nerf: Conditional generative neural radiance fields. *arXiv.org*, 2021. 3

[19] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. 1, 3, 5, 15

[20] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 7, 15

[21] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. Alias-free generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3, 14

[22] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3, 5, 6, 7, 8, 9, 14, 15, 16, 19, 20

[23] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of StyleGAN. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 7

[25] T. B. Lee. Cat hipsterizer, 2018. 7

[26] Y. Liao, K. Schwarz, L. Mescheder, and A. Geiger. Towards unsupervised learning of generative models for 3d controllable image synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3

[27] H. Ling, K. Kreis, D. Li, S. W. Kim, A. Torralba, and S. Fidler. Editgan: High-precision semantic image editing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[28] L. Liu, J. Gu, K. Z. Lin, T. Chua, and C. Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[29] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? In *Proc. of the International Conf. on Machine learning (ICML)*, 2018. 3, 6, 15

[30] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2, 3, 4

[31] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. on Graphics*, 2022. 3

[32] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2, 3

[33] M. Niemeyer and A. Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 6, 8, 14, 15

[34] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 3, 8, 16

[35] M. Oechsle, L. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger. Texture fields: Learning texture representations in function space. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 14

[36] R. Or-El, X. Luo, M. Shan, E. Shechtman, J. Park, and I. Kemelmacher. Stylesdf: High-resolution 3d-consistent image and geometry generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[37] X. Pan, X. Xu, C. C. Loy, C. Theobalt, and B. Dai. A shading-guided generative implicit model for shape-accurate 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[38] C. Reiser, S. Peng, Y. Liao, and A. Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 3

[39] A. Sauer, K. Chitta, J. Müller, and A. Geiger. Projected gans converge faster. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1

[40] A. Sauer, K. Schwarz, and A. Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. *ACM Trans. on Graphics*, 2022. 1, 3

[41] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. GRAF: generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems (NIPS)*, 2020. 1, 3, 7, 15

[42] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 3, 6, 7, 8, 9, 15, 16, 19, 22

[43] Y. Shi, D. Aggarwal, and A. K. Jain. Lifting 2d stylegan for 3d-aware face generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[44] C. Sun, M. Sun, and H.-T. Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 4

[45] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[46] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 4

[47] X. Xu, X. Pan, D. Lin, and B. Dai. Generative occupancy fields for 3d surface-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3, 4, 8, 14, 15, 16

[48] Y. Xu, S. Peng, C. Yang, Y. Shen, and B. Zhou. 3d-aware image synthesis via learning structural and textural representations. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3, 8, 16

[49] Y. Xue, Y. Li, K. K. Singh, and Y. J. Lee. GIRAFFE HD: A high-resolution 3d-aware generative model. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6, 15

[50] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 3

[51] J. Zhang, E. Sangineto, H. Tang, A. Siarohin, Z. Zhong, N. Sebe, and W. Wang. 3d-aware semantic-guided generative model for human synthesis. *arXiv.org*, 2021. 3

[52] Y. Zhang, W. Chen, H. Ling, J. Gao, Y. Zhang, A. Torralba, and S. Fidler. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[53] P. Zhou, L. Xie, B. Ni, and Q. Tian. CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis. *arXiv.org*, 2021. 3

[54] J.-Y. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. B. Tenenbaum, and W. T. Freeman. Visual object networks: Image generation with disentangled 3D representations. *Advances in Neural Information Processing Systems (NIPS)*, 2018. 3

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See supplemental material.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will provide code upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Multiple runs are computationally infeasible for our models.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We include total training time and the type of GPU we used.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Implementation

Foreground Generator The foreground generator builds on the StyleGAN2 generator [22] replacing 2D operations with their 3D equivalent as described in the main paper. For faster training, we consider the layers of StyleGAN2 instead of their alias-free version proposed in StyleGAN3 [21]. The mapping network has 2 layers with 64 channels. Since 3D convolutions have more parameters than 2D convolutions with the same channel size, we reduce the channel base[3] from 32768 for StyleGAN2 to 4000. To facilitate progressive growing we choose an architecture with skip connections which adds an upsampled version of the output grid of the previous layer to the input of the next layer. The skip architecture is equivalent to the 2D variant proposed in [22].

Following [4], we condition the generator on a camera pose. Specifically, we condition the generator on a rotation matrix and a translation vector, yielding a 12-dimensional vector as input to the conditioning.

The foreground generator predicts color and density values on a sparse voxel grid. Following Plenoxels [1], the generator outputs the coefficients of spherical harmonics. We choose spherical harmonics of degree 0, i.e., a single coefficient for each color channel. For a sharp surface and efficient rendering, the foreground generator needs to predict high values for the density. We facilitate generating high values by multiplying the density output of the network with a factor of 30. A similar idea was proposed in [1] where the learning rate for the density is set to a higher value than the learning rate for the color.

Sparse Convolutions We investigate sparse convolutions [12] for the foreground generator but find that the computational overhead of managing coordinates increases runtime for our architecture. We therefore use dense convolutions and zero out values in the feature maps for pruned voxels. We also compare the difference for both implementations on performance. In general, we observe similar training behavior for both implementations. However, the faster dense implementation allows us to train the model for 60M iterations compared to 30M for the model with sparse convolutions. This improves FID from 14.4 for the sparse implementation to 9.0 for the dense implementation on FFHQ 256.

Background Generator We use the StyleGAN2 [22] generator with a 2-layer mapping network with 64 channels, and a synthesis network with channel base 2048 and a maximum of 64 channels per layer.

2D Refinement Layers Depending on the dataset, we optionally refine the rendered image with a shallow 2D CNN with 2 hidden layers of dimension 16 and kernel size 3. To avoid texture sticking under viewpoint changes we use alias-free layers [21] with critical sampling.

Regularization Without regularization, volume rendering is prone to result in semi-opaque voxels and floating artifacts and struggles to accurately represent sharp surfaces [17, 33, 35, 47]. Therefore, we regularize both the variance of the depth, as described in the main paper, and the total variation of the predicted density. For the depth variance loss $\mathcal{L}_{DV}$, we set $\tau = (1.5\delta_0)^2$ where δ_0 is the size of one voxel and $\lambda_{DV} = 0.01$.

Following Plenoxels [1], we regularize the total variation of the predicted density values in the set of all voxels $\mathcal{V}$ for a compact, smooth geometry

$$\mathcal{L}_{TV} = \lambda_{TV} \frac{1}{|\mathcal{V}|} \sum_{\mathbf{v} \in \mathcal{V}} \sqrt{\Delta_x^2(\sigma) + \Delta_y^2(\sigma) + \Delta_z^2(\sigma)} \tag{13}$$

with $\Delta_x^2(\sigma)$ shorthand for $(\sigma_{i,j,k} - \sigma_{i+1,j,k})^2$ and analogously for $\Delta_y^2(\sigma)$ and $\Delta_z^2(\sigma)$. For efficiency, we evaluate the loss stochastically on random contiguous segments of voxels as proposed in [1] and set $\lambda_{TV} = 10^{-5}$ in all experiments.

[3] see https://github.com/NVlabs/stylegan3.git

To avoid that the full image is generated by either background or foreground generator, we use a hinge loss on the mean mask value as proposed in GIRAFFE-HD [49]

$$\mathcal{L}_{cvg}^{fg} = \lambda_{cvg}^{fg} \max\left(0, \eta^{fg} - \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \hat{\mathbf{I}}_{mask}^{fg}[i]\right) \tag{14}$$

$$\mathcal{L}_{cvg}^{bg} = \lambda_{cvg}^{bg} \max\left(0, \eta^{bg} - \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} 1 - \hat{\mathbf{I}}_{mask}^{fg}[i]\right) \tag{15}$$

where η^{fg} and η^{bg} denote the minimum fraction that should be covered by foreground and background, respectively. To ensure that both models are used, we set $\eta^{fg} = 0.4$ for AFHQ [6] and FFHQ [22] and $\eta^{fg} = 0.1$ for Carla [42] because Carla's objects cover a much smaller fraction of the image. We use $\eta^{bg} = 0.1$ and $\lambda_{cvg}^{fg} = \lambda_{cvg}^{bg} = 0.1$ for all datasets.

Discriminator We use the StyleGAN2 [22] discriminator with conditional input as in [4]. To facilitate progressive growing we choose a skip architecture which adds a downsampled version of the input image to the input of each layer as introduced in [22].

Rendering For efficient rendering, we leverage custom CUDA kernels building on the official code release of [1]. We select equidistant sampling points for volume rendering in steps of 0.5 voxels but skip voxels with $\sigma_i < 10^{-10}$ and stop rendering early if $T_i < 10^{-7}$ as in [1].

Implementation and Training Our code base builds on the official PyTorch implementation of StyleGAN2 [22] available at https://github.com/NVlabs/stylegan3. Similar to StyleGAN2, we train with equalized learning rates for the trainable parameters and a minibatch standard deviation layer at the end of the discriminator [19] and apply an exponential moving average of the generator weights. For faster training, we use mixed-precision for both the generator and the discriminator as proposed in [20]. Unlike [22], we do not train with path regularization or style-mixing. To reduce computational cost and overall memory usage R1-regularization [29] is applied only once every 4 minibatches. We use a regularization strength of $\gamma = 1$ for all datasets. Due to the small size of AFHQ, we follow [20] and finetune a generator that is pretrained on FFHQ with $R_I = 128$ and $R_G = 32$, i.e., before the representation is pruned.

B Baselines

Qualitative Results We provide qualitative results for StyleNeRF [11] and GRAM [7] in both the main paper and the supplemental material. For StyleNeRF, we obtain samples from the pretrained FFHQ model available at https://github.com/facebookresearch/StyleNeRF.git. For GRAM, its authors kindly provided unpublished code and pretrained models which we use for evaluation. In the qualitative comparisons, i.e., Fig. 4 of the main paper and Fig. 8, we use a truncation of $\psi = 0.7$ for all methods. For GRAM and our approach, we show samples from $-40°$ to $+40°$ which roughly corresponds to 2 standard deviations of the pose distribution. We find that StyleNeRF does not necessarily adhere to the input pose. Hence, we manually define the range to be $-60°$ to $+60°$ such that the rendered images roughly align with the other methods.

Quantitative Results Table 2 of the main paper shows a quantitative comparison for all baselines and our method in terms of FID. For EG3D [4] and GIRAFFE [33], we report the numbers from [4]. For StyleNeRF [11], we take the numbers from [11]. From [11] we further reference the results on FFHQ and AFHQ for GRAF [41] and π-GAN [5] as these datasets were not considered in the original publications. On Carla, we report the results from [41] and [5], respectively. For GOF [47], we reference the numbers from [47] on Carla and train new models to obtain results on FFHQ and AFHQ using the official code release available at https://github.com/SheldonTsui/GOF_NeurIPS2021.git. For GRAM [7], we report the results from [7] for FFHQ. As AFHQ is not considered in [7], we train GRAM on AFHQ using their unpublished code. Similar to our approach, we finetune a generator that was pre-trained on FFHQ. We remark that across all reported values for FID the number of generated image varies where most methods report values considering either 20k or 50k generated images. An overview is provided in Table 4 which is discussed in more detail at the end of Section C.

Figure 6: **Background disentanglement.** We show generated images at resolution 256^2 for FFHQ [22] and AFHQ [6] with truncation $\psi = 0.7$.

	FFHQ [22] $R_I = 256^2$	AFHQ [6] $R_I = 256^2$	Carla [42] $R_I = 128^2$
GIRAFFE [34]	31.5†	16.1†	–
VolumeGAN [48]	9.1†	–	7.9†
StyleNeRF [11]	8.0†	–	–
EG3D [4]	4.8†	3.9†	–
GRAF [42]	71†	121†	41^{*1}
π-GAN [5]	85†	47†	29.2^{*2}
GOF [47]	69.2	54.1	29.3
GRAM [7]	17.9	18.5	26.3
VoxGRAF	9.6 / 9.0†	9.6 / 9.4†	6.7 / 6.3†

Table 4: **Quantitative Comparison.** We report FID [16] on the full dataset and explicitly annotate the number of generated images for evaluation. † denotes 50k generated images, ^{*1} denotes 1k images (GRAF, Carla) and ^{*2} denotes 8k images (π-GAN, Carla). Numbers without annotation are calculated using 20k generated images.

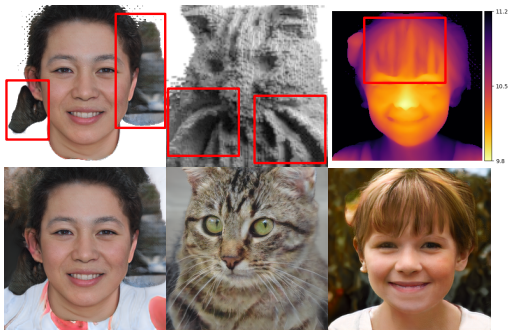


Figure 7: **Failure Cases.** Left: Some part of the background is modeled with the foreground. Middle: The whiskers are connected to the body of the cat. Right: The hair is directed inward to the head.

C Results

Background and Foreground Disentanglement. Fig. 6 illustrates foreground masks, foreground and background image, and the image after alpha composition. As the background remains fixed under viewpoint changes, the generator is encouraged to model pose-dependent content with the foreground generator. The regularization in Eq. (14) and Eq. (15) encourages the generator to use both the background and the foreground generator to synthesize the full image.

Multi-View Consistency. Corresponding to Fig. 4 of the main paper, we provide additional qualitative comparisons on multi-view consistency in Fig. 8. For StyleNeRF [11], the red boxes highlight inconsistencies, like changing eye shape (first row on the left), moving strands of hair (second row, third row on the left) and distortion of the face shape (first and third row on the right). For GRAM [7], red boxes indicate layered artifacts stemming from its manifold representation. In contrast, our method leads to more multi-view consistent results.
We further include a quantitative evaluation on consistency in Table 6. We implemented our own version of the depth and pose metric following the description in [4] as their evaluation code is not publicly available. We report the results for our approach, GRAM, StyleNeRF and EG3D for

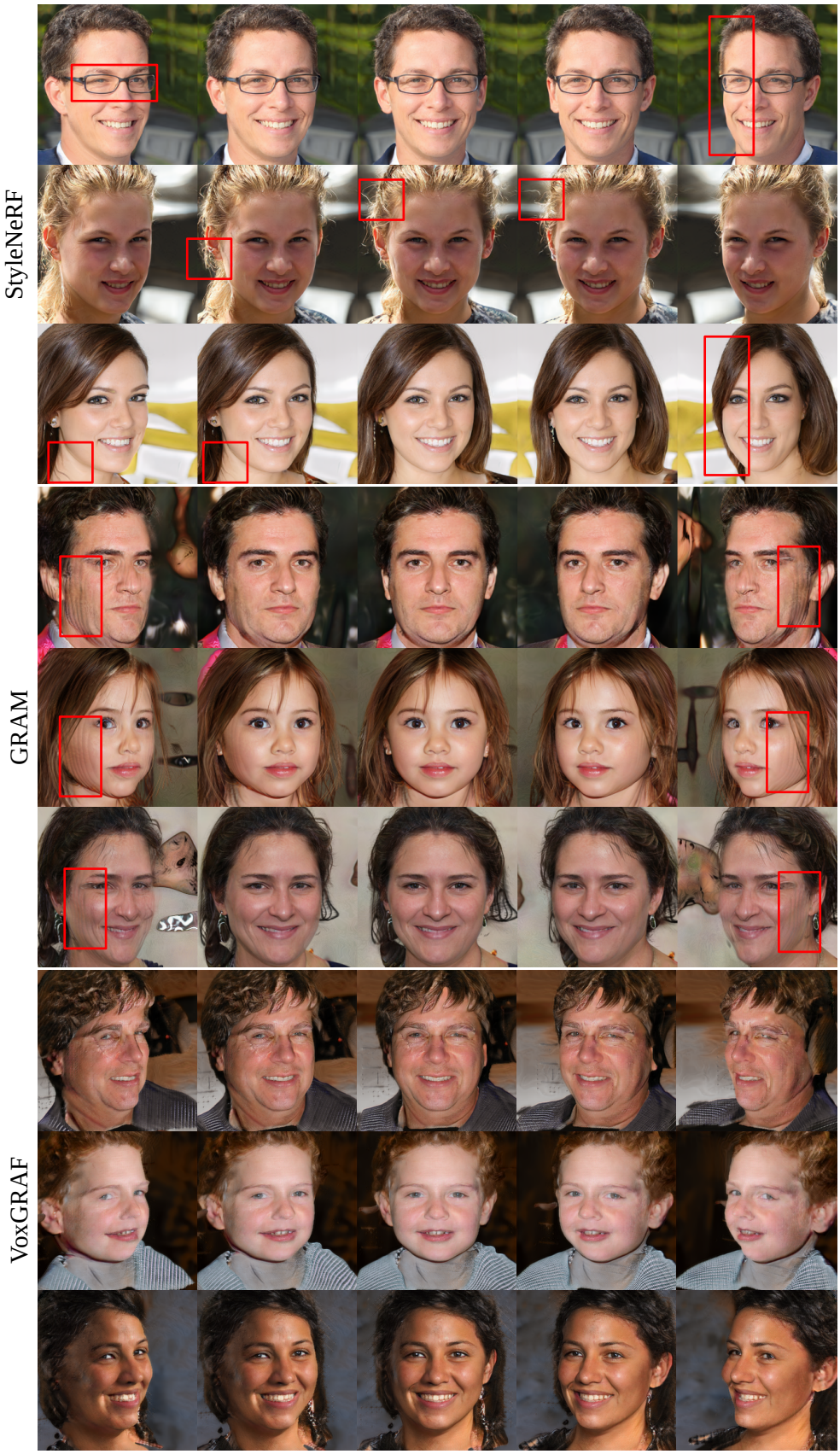


Figure 8: **Qualitative Comparison on Multi-View Consistency**. We apply truncation with $\psi = 0.7$ for all methods.

Figure 9: **Effect of Pose Conditioning**. From left to right we vary the pose used for rendering ξ' and from top to bottom we vary the pose used for conditioning ξ. All samples are generated using the same latent $\mathbf{z}$.

reference. We also report the standard deviation across the 1024 samples used for evaluation. While the results in Table 6 agree with our qualitative analysis of view consistency in Fig. 4 and the supplementary video, we find that both metrics are very sensitive to the latent code and the sampled poses, as indicated by the large standard deviations. As no established evaluation pipeline exists, our results should not be directly compared to the numbers in [4] as the implementation and pose sampling might differ.

Pose Conditioning. Fig. 9 illustrates the impact of the pose conditioning on the generated images. Pose conditioning is not only used to model slight changes, e.g. of the eyes or the smile, but can alter the general appearance of the generated instance. However, by fixing the pose conditioning during inference, view-consistent images can be generated.

Regularization. We ablate the effect of regularization in terms of end-to-end performance measured in FID. Table 5 shows that the regularizers do not significantly change FID. Nonetheless, $\mathcal{L}_{DV}$ speeds up training (see Table 1 of the main paper) and we find the remaining losses to be helpful for stabilizing training.

	$\mathcal{L}_{reg}$	w/o $\mathcal{L}_{DV}$	w/o $\mathcal{L}_{TV}$	w/o $\mathcal{L}_{cvg}^{fg}$	w/o $\mathcal{L}_{cvg}^{bg}$
FID	14.2	14.8	15.1	14.6	14.8

Table 5: **Impact of Regularization on FID.** We ablate the performance of all four regularizers on models trained on FFHQ with $R_I = 128$ and $R_G = 64$. While the regularizers do not significantly impact end-to-end performance measured in FID, we find that they can be helpful to stabilize training.

	Depth ↓	Pose ↓
StyleNeRF [11]	–	0.051 ± 0.047
GRAM [7]	0.48 ± 0.24	0.013 ± 0.013
EG3D [4]	0.29 ± 0.30	0.0018 ± 0.0031
VoxGRAF	0.33 ± 0.23	0.00045 ± 0.00079

Table 6: **View-Consistency**. We re-implement the depth and pose metric from [4]. While results agree with the qualitative evaluation in Fig. 4 of the main paper, the large standard deviations indicate that both metrics are very sensitive to the latent code and the sampled poses. Note that for StyleNeRF depth can only be rendered at resolution 32^2 and we thus omit evaluating the Depth metric for it.

Failure Cases. Fig. 7 illustrates failure cases of out method. For some samples, we observe that the background and the foreground are not disentangled properly. The left column in Fig. 7 shows an example where the foreground generates parts of the background, as indicated by the red boxes. In turn, the background models the body of the person which should be part of the foreground. Learning to disentangle background and foreground without direct supervision, e.g., instance masks, is often ambiguous which makes it a challenging task. The middle column in Fig. 7 displays a common failure case of our model on AFHQ: The whiskers of the cat are connected to its body. This is likely a consequence from the depth variance and total variation regularization we apply to obtain a single sharp surface and compact geometry. The right column in Fig. 7 shows an occasional failure on FFHQ. For some samples, we observe that the hair is directed inward to the head instead of outward. In the rendered image this effect is not visible which suggests that it likely results from ambiguity in the training data.

Uncurated Samples. We provide additional samples of our method for FFHQ [22] in Fig. 10, AFHQ [6] in Fig. 11, and Carla [42] in Fig. 12.

Quantitative Comparison. As FID is biased towards the number of images [2], we explicitly annotate the number of generated images for the results reported in Table 2 of the main paper in Table 4. Most works evaluate FID either using 20k or 50k generated images. We evaluate our method for both cases. For our method and the considered datasets, the difference between using 20k or 50k generated images is reasonably small.

D Societal Impact

This work considers the task of generating photorealistic renderings of scenes with data-driven approaches which has potential downstream applications in virtual reality, augmented reality, gaming and simulation. While many use-cases are possible, we believe that in the long run this line of research could support designers in creating renderings of 3D models more efficiently. However, generating photorealistic 3D-scenarios also bears the risk of manipulation, e.g., by creating edited imagery of real people. Further, like all data-driven approaches, our method is susceptible to biases in the training data. Such biases can, e.g., result in a lack of diversity for the generated faces and have to be addressed before using this work for any downstream applications.

Figure 10: **Uncurated Samples for FFHQ [22].** We use truncation with $\psi = 0.7$.

Figure 11: **Uncurated Samples for AFHQ [6].** We use truncation with $\psi = 0.7$.

Figure 12: **Uncurated Samples for Carla [42].** We use truncation with $\psi = 0.7$.