This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering

Tao Lu [1,3*] Mulin Yu[1*] Linning Xu[2] Yuanbo Xiangli[4]
Limin Wang[3,1] Dahua Lin[1,2] Bo Dai[1]✉
[1]Shanghai Artificial Intelligence Laboratory, [2]The Chinese University of Hong Kong,
[3]Nanjing University, [4]Cornell University
taolu@smail.nju.edu.cn, yumulin@pjlab.org.cn, linningxu@link.cuhk.edu.hk,
yx642@cornell.edu, lmwang@nju.edu.cn, dhlin@ie.cuhk.edu.hk, daibo@pjlab.org.cn

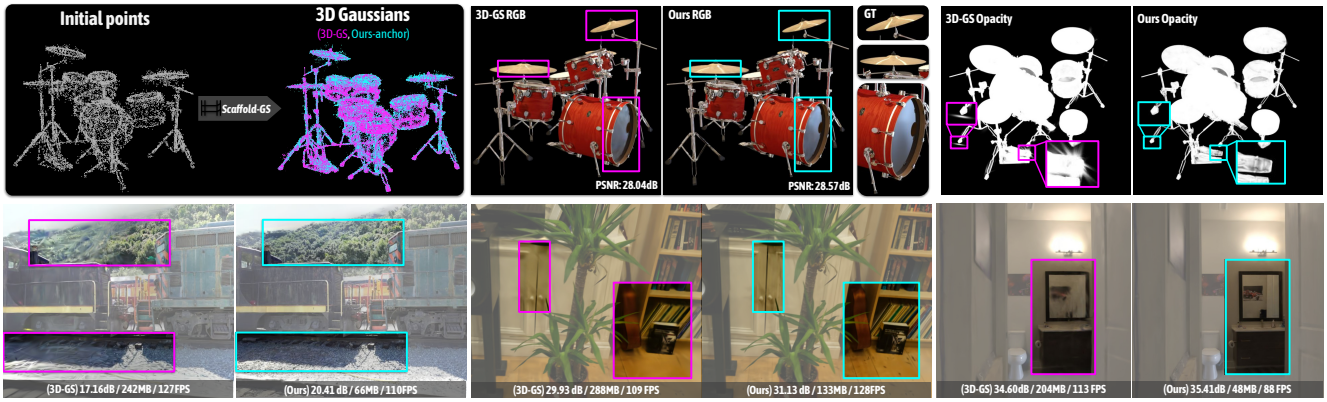

Figure 1. ***Scaffold*-GS** represents the scene using a set of 3D Gaussians structured in a dual-layered hierarchy. Anchored on a sparse grid of initial points, a modest set of neural Gaussians are spawned from each anchor to *dynamically adapt to* various viewing angles and distances. Our method achieves rendering quality and speed comparable to 3D-GS with a more compact model (last row metrics: PSNR/storage size/FPS). Across multiple datasets, *Scaffold*-GS demonstrates more robustness in large outdoor scenes and intricate indoor environments with challenging observing views *e.g.* transparency, specularity, reflection, texture-less regions and fine-scale details.

Abstract

Neural rendering methods have significantly advanced photo-realistic 3D scene rendering in various academic and industrial applications. The recent 3D Gaussian Splatting method has achieved the state-of-the-art rendering quality and speed combining the benefits of both primitive-based representations and volumetric representations. However, it often leads to heavily redundant Gaussians that try to fit every training view, neglecting the underlying scene geometry. Consequently, the resulting model becomes less robust to significant view changes, texture-less area and lighting effects. We introduce Scaffold-GS, which uses anchor points to distribute local 3D Gaussians, and predicts their attributes on-the-fly based on viewing direction and distance within the view frustum. Anchor growing and pruning strategies are developed based on the importance of neural Gaussians to reliably improve the scene coverage. We show that our method effectively reduces redundant Gaussians while delivering high-quality rendering. We also demonstrates an enhanced capability to accommodate scenes with varying levels-of-detail and view-dependent observations, without sacrificing the rendering speed. Project page: https://city-super.github.io/scaffold-gs/.

1. Introduction

Photo-realistic and real-time rendering of 3D scenes has always been a pivotal interest in both academic research and industrial domains, with applications spanning virtual reality [51], media generation [36], and large-scale scene visualization [43, 45, 49]. Traditional primitive-based representations like meshes and points [6, 26, 32, 55] are faster due to the use of rasterization techniques tailored for modern GPUs. However, they often yield low-quality renderings, exhibiting discontinuity and blurry artifacts. In contrast, volumetric representations and neural radiance fields utilize learning-based parametric models [3, 5, 30], hence

* denotes equal contribution.

can produce continuous rendering results with more details preserved. Nevertheless, they come with the cost of time-consuming stochastic sampling, leading to slower performance and potential noise.

In recent times, 3D Gaussian Splatting (3D-GS) [22] has achieved state-of-the-art rendering quality and speed. Initialized from point clouds derived from Structure from Motion (SfM) [42], this method optimizes a set of 3D Gaussians to represent the scene. It preserves the inherent continuity found in volumetric representations, whilst facilitating rapid rasterization by splatting 3D Gaussians onto 2D image planes. While this approach offers several advantages, it tends to excessively expand Gaussian balls to accommodate every training view, thereby neglecting scene structure. This results in significant redundancy and limits its scalability, particularly in the context of complex large-scale scenes. Furthermore, view-dependent effects are baked into individual Gaussian parameters with little interpolation capabilities, making it less robust to substantial view changes and lighting effects.

We present *Scaffold*-GS, a Gaussian-based approach that utilizes anchor points to establish a hierarchical and region-aware 3D scene representation. We construct a sparse grid of anchor points initiated from SfM points. Each of these anchors tethers a set of neural Gaussians with learnable offsets, whose attributes (*i.e*. opacity, color, rotation, scale) are dynamically predicted based on the anchor feature and the viewing position. Unlike the vanilla 3D-GS which allows 3D Gaussians to freely drift and split, our strategy exploits scene structure to guide and constrain the distribution of 3D Gaussians, whilst allowing them to *locally* adaptive to varying viewing angles and distances. We further develop the corresponding growing and pruning operations for anchors to enhance the scene coverage.

Through extensive experiments, we show that our method delivers rendering quality on par with or even surpassing the original 3D-GS. At inference time, we limit the prediction of neural Gaussians to anchors within the view frustum, and filter out trivial neural Gaussians based on their opacity with a filtering step (*i.e*. learnable selector). As a result, our approach can render at a similar speed (around 100 FPS at 1K resolution) as the original 3D-GS with little computational overhead. Moreover, our storage requirements are significantly reduced as we only need to store anchor points and MLP predictors for each scene.

In conclusion, our contributions are: 1) Leveraging scene structure, we initiate *anchor points* from a sparse voxel grid to guide the distribution of local 3D Gaussians, forming a hierarchical and region-aware scene representation; 2) Within the view frustum, we predict *neural* Gaussians from each anchor *on-the-fly* to accommodate diverse viewing directions and distances, resulting in more robust novel view synthesis; 3) We develop a *more reliable* anchor growing and pruning strategy utilizing the predicted neural Gaussians for better scene coverage.

2. Related work

MLP-based Neural Fields and Rendering. Early neural fields typically adopt a multi-layer perceptron (MLP) as the global approximator of 3D scene geometry and appearance. They directly use spatial coordinates (and viewing direction) as input to the MLP and predict point-wise attribute, *e.g*. signed distance to scene surface (SDF) [33, 34, 46, 54], or density and color of that point [2, 30, 49]. Because of its volumetric nature and inductive bias of MLPs, this stream of methods achieves the SOTA performance in novel view synthesis. The major challenge of this scene representation is that the MLP need to be evaluated on a large number of sampled points along each camera ray. Consequently, rendering becomes extremely slow, with limited scalability towards complex and large-scale scenes. Despite several works have been proposed to accelerate or mitigate the intensive volumetric ray-marching, *e.g*. using proposal network [4], baking technique [11, 19], and surface rendering [41]. They either incorporated more MLPs or traded rendering quality for speed.

Grid-based Neural Fields and Rendering. This type of scene representations are usually based on a dense uniform grid of voxels. They have been greatly used in 3D shape and geometry modeling [12, 15, 21, 29, 35, 44, 57]. Some recent methods have also focused on faster training and inference of radiance field by exploiting spatial data structure to store scene features, which were interpolated and queried by sampled points during ray-marching. For instance, Plenoxel [13] adopted a sparse voxel grid to interpolate a continuous density field, and represented view-dependent visual effects with Spherical Harmonics. The idea of tensor factorization has been studied in multiple works [9, 10, 50, 52] to further reduce data redundancy and speed-up rendering. K-planes [14] used neural planes to parameterize a 3D scene, optionally with an additional temporal plane to accommodate dynamics. Several generative works [8, 40] also capitalized on triplane structure to model a 3D latent space for better geometry consistency. Instant-NGP [31] used a hash grid and achieved drastically faster feature query, enabling real-time rendering of neural radiance field. Although these approaches can produce high-quality results and are more efficient than global MLP representation, they still need to query many samples to render a pixel, and struggle to represent empty space effectively.

Point-based Neural Fields and Rendering. Point-based representations utilize the geometric primitive (*i.e*. point clouds) for scene rendering. A typical procedure is to ras-

terize an unstructured set of points using a fixed size, and exploits specialized modules on GPU and graphics APIs for rendering [7, 37, 38]. In spite of its fast speed and flexibility to solve topological changes, they usually suffer from holes and outliers that lead to artifacts in rendering. To alleviate the discontinuity issue, differentiable point-based rendering has been extensively studied to model objects geometry [16, 20, 27, 48, 55]. In particular, [48, 55] used differentiable surface splatting that treats point primitives as discs, ellipsoids or surfels that are larger than a pixel. [1, 24] augmented points with neural features and rendered using 2D CNNs. As a comparison, Point-NeRF [53] achieved high-quality novel view synthesis utilizing 3D volume rendering, along with region growing and point pruning during optimization. However, they resorted to volumetric ray-marching, hence hindered display rate. Notably, the recent work 3D-GS [22] employed anisotropic 3D Gaussians initialized from structure from motion (SfM) to represent 3D scenes, where a 3D Gaussian was optimized as a volume and projected to 2D to be rasterized as a primitive. Since it integrated pixel color using α-blender, 3D-GS produced high-quality results with fine-scale detail, and rendered at real-time frame rate.

3. Methods

The original 3D-GS [22] optimizes Gaussians to reconstruct every training view, with heuristic splitting and pruning operations but in general neglects the underlying scene structure. This often leads to highly redundant Gaussians and makes the model less robust to novel viewing angles and distances. To address this issue, we propose a hierarchical 3D Gaussian scene representation that respects the scene geometric structure, with *anchor points* initialized from SfM to encode local scene information and spawn local *neural Gaussians*. The physical properties of neural Gaussians are decoded from the learned anchor features in a view-dependent manner *on-the-fly*. Fig. 2 illustrates our framework. We start with a brief background of 3D-GS then unfold our proposed method in details. Sec. 3.2.1 introduces how to initialize the scene with a regular sparse grid of anchor points from the irregular SfM point clouds. Sec. 3.2.2 explains how we predict neural Gaussians properties based on anchor points and view-dependent information. To make our method more robust to the noisy initialization, Sec. 3.3 introduces a neural Gaussian based "growing" and "pruning" operations to refine the anchor points. Sec. 3.4 elaborates training details.

3.1. Preliminaries

3D-GS [22] represents the scene with a set of anisotropic 3D Gaussians that inherit the differential properties of volumetric representation while be efficiently rendered via a tile-based rasterization.

Starting from a set of Structure-from-Motion (SfM) points, each point is designated as the position (mean) μ of a 3D Gaussian:

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

where x is an arbitrary position within the 3D scene and Σ denotes the covariance matrix of the 3D Gaussian. Σ is formulated using a scaling matrix S and rotation matrix R to maintain its positive semi-definite:

$$\Sigma = RSS^T R^T. \tag{2}$$

In addition to color c modeled by Spherical harmonics, each 3D Gaussian is associated with an opacity α which is multiplied by $G(x)$ during the blending process.

Distinct from conventional volumetric representations, 3D-GS efficiently renders the scene via tile-based rasterization instead of resource-intensive ray-marching. The 3D Gaussian $G(x)$ are first transformed to 2D Gaussians $G'(x)$ on the image plane following the projection process as described in [58]. Then a tile-based rasterizer is designed to efficiently sort the 2D Gaussians and employ α-blending:

$$C(x') = \sum_{i \in N} c_i \sigma_i \prod_{j=1}^{i-1} (1 - \sigma_j), \quad \sigma_i = \alpha_i G'_i(x'), \tag{3}$$

where x' is the queried pixel position and N denotes the number of sorted 2D Gaussians associated with the queried pixel. Leveraging the differentiable rasterizer, all attributes of the 3D Gaussians are learnable and directly optimized end-to-end via training view reconstruction.

3.2. *Scaffold*-GS

3.2.1 Anchor Point Initialization

Consistent with existing methods [22, 53], we use the sparse point cloud from COLMAP [39] as our initial input. We then voxelize the scene from this point cloud $\mathbf{P} \in \mathbb{R}^{M \times 3}$ as:

$$\mathbf{V} = \left\{ \left\lfloor \frac{\mathbf{P}}{\epsilon} \right\rceil \right\} \cdot \epsilon, \tag{4}$$

where $\mathbf{V} \in \mathbb{R}^{N \times 3}$ denotes voxel centers, and ϵ is the voxel size. $\lfloor \cdot \rceil$ denotes rounding operation. We then remove duplicate entries, denoted by $\{\cdot\}$ to reduce the redundancy and irregularity in $\mathbf{P}$.

The center of each voxel $v \in \mathbf{V}$ is treated as an anchor point, equipped with a local context feature $f_v \in \mathbb{R}^{32}$, a scaling factor $l_v \in \mathbb{R}^3$, and k learnable offsets $\mathbf{O}_v \in \mathbb{R}^{k \times 3}$. In a slight abuse of terminology, we will denote the anchor point as v in the following context. We further enhance f_v to be multi-resolution and view-dependent. For each anchor v, we 1) create a features bank $\{f_v, f_{v_{\downarrow_1}}, f_{v_{\downarrow_2}}\}$, where $\downarrow_n$ denotes f_v being down-sampled by 2^n factors in channel

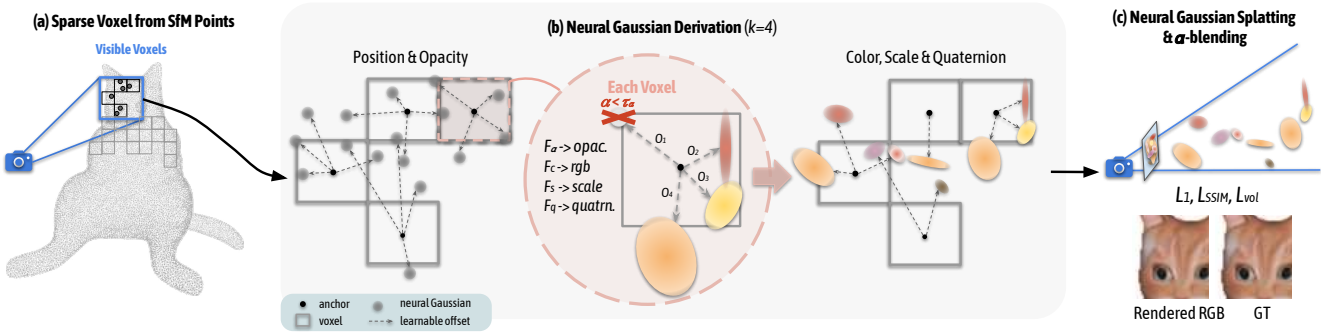

Figure 2. **Overview of *Scaffold*-GS**. (a) We start by forming a *sparse voxel grid* from SfM-derived points. An **anchor** associated with a learnable scale is placed at the center of each voxel, roughly sculpturing the scene occupancy. (b) Within a view frustum, k **neural Gaussians** are spawned from each *visible anchor* with offsets $\{\mathcal{O}_k\}$. Their attributes, *i.e.* opacity, color, scale and quaternion are then decoded from the anchor feature, relative camera-anchor viewing direction and distance using F_α, F_c, F_s, F_q respectively. (c) Note that to alleviate redundancy and improve efficiency, only non-trivial neural Gussians (*i.e.* $\alpha \geq \tau_\alpha$) are rasterized following [22]. The rendered image is supervised via reconstruction ($\mathcal{L}_1$), structural similarity ($\mathcal{L}_{SSIM}$) and a volume regularization ($\mathcal{L}_{vol}$).

dimension; 2) blend the feature bank with view-dependent weights to form an integrated anchor feature $\hat{f}_v$. Specifically, Given a camera at position $\mathbf{x}_c$ and an anchor at position $\mathbf{x}_v$, we calculate their relative distance and viewing direction with:

$$\delta_{vc} = \|\mathbf{x}_v - \mathbf{x}_c\|_2, \vec{\mathbf{d}}_{vc} = \frac{\mathbf{x}_v - \mathbf{x}_c}{\|\mathbf{x}_v - \mathbf{x}_c\|_2}, \tag{5}$$

then weighted sum the feature bank with weights predicted from a tiny MLP F_w:

$$\{w, w_1, w_2\} = \text{Softmax}(F_w(\delta_{vc}, \vec{\mathbf{d}}_{vc})), \tag{6}$$

$$\hat{f}_v = w \cdot f_v + w_1 \cdot f_{v_{\downarrow_1}} + w_2 \cdot f_{v_{\downarrow_2}}. \tag{7}$$

3.2.2 Neural Gaussian Derivation

In this section, we elaborate on how our approach derives neural Gaussians from anchor points. Unless specified otherwise, F_* represents a particular MLP throughout the section. Moreover, we introduce two efficient pre-filtering strategies to reduce MLP overhead.

We parameterize a neural Gaussian with its position $\mu \in \mathbb{R}^3$, opacity $\alpha \in \mathbb{R}$, covariance-related quaternion $q \in \mathbb{R}^4$ and scaling $s \in \mathbb{R}^3$, and color $c \in \mathbb{R}^3$. As shown in Fig. 2(b), for each visible anchor point within the viewing frustum, we spawn k neural Gaussians and predict their attributes. Specifically, given an anchor point located at $\mathbf{x}_v$, the positions of its neural Gaussians are calculated as:

$$\{\mu_0, ..., \mu_{k-1}\} = \mathbf{x}_v + \{\mathcal{O}_0, \ldots, \mathcal{O}_{k-1}\} \cdot l_v, \tag{8}$$

where $\{\mathcal{O}_0, \mathcal{O}_1, ..., \mathcal{O}_{k-1}\} \in \mathbb{R}^{k\times 3}$ are the learnable offsets and l_v is the scaling factor associated with that anchor, as described in Sec. 3.2.1. The attributes of k neural Gaussians are directly decoded from the anchor feature $\hat{f}_v$, the relative viewing distance δ_{vc} and direction $\vec{\mathbf{d}}_{vc}$ between the camera and the anchor point (Eq. 5) through individual MLPs, denoted as F_α, F_c, F_q and F_s. Note that attributes are decoded in *one-pass*. For example, opacity values of neural Gaussians spawned from an anchor point are given by:

$$\{\alpha_0, ..., \alpha_{k-1}\} = F_\alpha(\hat{f}_v, \delta_{vc}, \vec{\mathbf{d}}_{vc}), \tag{9}$$

their colors $\{c_i\}$, quaternions $\{q_i\}$ and scales $\{s_i\}$ are similarly derived. Implementation details are in supplementary.

Note that the prediction of neural Gaussian attributes are *on-the-fly*, meaning that only anchors visible within the frustum are activated to spawn neural Gaussians. To make the rasterization more efficient, we only keep neural Gaussians whose opacity values are larger than a predefined threshold τ_α. This substantially cuts down the computational load and helps our method maintain a high rendering speed on-par with the original 3D-GS.

3.3. Anchor Points Refinement

Growing Operation. Since neural Gaussians are closely tied to their anchor points which are initialized from SfM points, their modeling power is limited to a local region, as has been pointed out in [22, 53]. This poses challenges to the initial placement of anchor points, especially in textureless and less observed areas. We therefore propose an error-based anchor growing policy that grows new anchors where neural Gaussians find *significant*. To determine a *significant* area, we first spatially quantize the neural Gaussians by constructing voxels of size ϵ_g. For each voxel, we compute the averaged gradients of the included neural Gaussians over N training iterations, denoted as ∇_g. Then, voxels with $\nabla_g > \tau_g$ are deemed as *significant*, where τ_g is a pre-defined threshold; and a new anchor point is thereby deployed at the center of that voxel if there was no an-

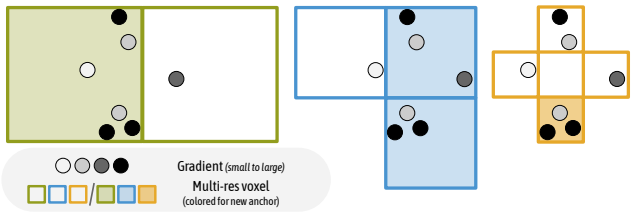

Figure 3. **Growing operation.** Dots represent neural Gaussians, with grey shades indicating varied accumulated gradients. From left to right, we spatially quantize neural Gaussians into multi-resolution voxels ($m \in \{1, 2, 3\}$) of size $\{\epsilon_g^{(m)}\}$ to capture scene details of different granularity (illustrated by voxels in three colors). For each voxel, we compute averaged gradients and generate a new anchor if it surpasses a threshold $\{\tau_g^{(m)}\}$ (indicated by colored fills).

chor point established. Fig. 3 illustrates this growing operation. In practice, we quantize the space into multi-resolution voxel grid to allow new anchors to be added at different granularity, where

$$\epsilon_g^{(m)} = \epsilon_g / 4^{m-1}, \quad \tau_g^{(m)} = \tau_g * 2^{m-1}, \tag{10}$$

where m denotes the level of quantization. To further regulate the addition of new anchors, we apply a random elimination to these candidates. This cautious approach to adding points effectively curbs the rapid expansion of anchors. Details in supplementary.

Pruning Operation To eliminate *trivial* anchors, we accumulate the opacity values of their associated neural Gaussians over N training iterations. If an anchor fails to produce neural Gaussians with a satisfactory level of opacity, we then remove it from the scene.

Observation Threshold To enhance the robustness of the Growing and Pruning operations for long image sequences, we implement a minimum observation threshold for anchor refinement control. We define the anchor update interval as N, allowing only anchors been visited more than $N \times r_g$ and $N \times r_p$ times to undergo growing and pruning, respectively.

3.4. Losses Design

We optimize the learnable parameters and MLPs with respect to the $\mathcal{L}_1$ loss over rendered pixel colors, with SSIM term [47] $\mathcal{L}_{SSIM}$ and volume regularization [28] $\mathcal{L}_{vol}$. The total supervision is given by:

$$\mathcal{L} = \mathcal{L}_1 + \lambda_{\text{SSIM}} \mathcal{L}_{\text{SSIM}} + \lambda_{\text{vol}} \mathcal{L}_{\text{vol}}, \tag{11}$$

where the volume regularization $\mathcal{L}_{\text{vol}}$ is:

$$\mathcal{L}_{\text{vol}} = \sum_{i=1}^{N_{\text{ng}}} \text{Prod}(s_i). \tag{12}$$

Here, N_{ng} denotes the number of neural Gaussians in the scene and $\text{Prod}(\cdot)$ is the product of the values of a vector, *e.g.*, in our case the scale s_i of each neural Gaussian. The volume regularization term encourages the neural Gaussians to be small with minimal overlapping.

4. Experiments

4.1. Experimental Setup

Dataset and Metrics We conducted a comprehensive evaluation across 27 scenes from publicly available datasets. Specifically, we tested our approach on all available scenes tested in the 3D-GS [22], including nine scenes from Mip-NeRF360 [4], two scenes from Tanks&Temples [23], two scenes from DeepBlending [18] and synthetic Blender dataset [30]. We additionally evaluated on datasets with contents captured at multiple LODs to demonstrate our advantages in view-adaptive rendering. Six scenes from BungeeNeRF [49] and two scenes from VR-NeRF [51] are selected. The former provides multi-scale outdoor observations and the latter captures intricate indoor environments. Apart from the commonly used metrics (PSNR, SSIM [47], and LPIPS [56]), we additionally report the storage size (MB) and the rendering speed (FPS) for model compactness and performance efficiency. We provide the averaged metrics over all scenes of each dataset in the main paper and leave the full quantitative results on each scene in the supplementary.

Baseline and Implementation. 3D-GS [22] is selected as our main baseline for its established SOTA performance in novel view synthesis. Both 3D-GS and our method were trained for 30k iterations. We also record the results of Mip-NeRF360 [4], iNGP [31] and Plenoxels [13] as in [22].

For our method, we set $k = 10$ for all experiments. All the MLPs employed in our approach are 2-layer MLPs with ReLU activation; the dimensions of the hidden units are all 32. For anchor points refinement, we average gradients over $N = 100$ iterations, and by default use $\tau_g = 64\epsilon$. And we set the observation threshold as $r_g = 0.4$, $r_p = 0.8$. On intricate scenes and the ones with dominant texture-less regions, we use $\tau_g = 16\epsilon$. An anchor is pruned if the accumulated opacity of its neural Gaussians is less than 0.5 at each round of refinement. The two loss weights λ_{SSIM} and λ_{vol} are set to 0.2 and 0.001 in our experiments. Please check the supplementary material for more details.

4.2. Results Analysis

Our evaluation was conducted on diverse datasets, ranging from synthetic object-level scenes, indoor and outdoor environments, to large-scale urban scenes and landscapes. A variety of improvements can be observed especially on challenging cases, such as texture-less area, insufficient obser-

Table 1. **Quantitative comparison to previous methods on real-world datasets.** Competing metrics are extracted from respective papers.

Dataset	Mip-NeRF360			Tanks&Temples			Deep Blending		
Method | Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3D-GS [22]	27.21	0.815	0.214	23.14	0.841	0.183	29.41	0.903	0.243
Mip-NeRF360 [4]	27.69	0.792	0.237	22.22	0.759	0.257	29.40	0.901	0.245
iNGP [31]	25.59	0.699	0.331	21.72	0.723	0.330	23.62	0.797	0.423
Plenoxels [13]	23.08	0.626	0.463	21.08	0.719	0.379	23.06	0.795	0.510
Ours	27.72	0.811	0.228	24.04	0.853	0.172	30.43	0.910	0.250

Figure 4. **Qualitative comparison of *Scaffold*-GS and 3D-GS [22] across diverse datasets [4, 17, 23, 51].** Patches that highlight the visual differences are emphasized with arrows and green & yellow insets for clearer visibility. Our approach consistently outperforms 3D-GS on these scenes, with evident advantages in challenging scenarios, *e.g.* thin geometry and fine-scale details (MIP360-ROOM(a), MIP360-COUNTER(a)), texture-less regions (DB-DRJOHNSON, DB-PLAYROOM), light effects (MIP360-COUNTER(b), DB-DRJOHNSON), insufficient observations (TANDT-TRAIN, VR-KITCHEN). It can also be observed (*e.g.* VR-APARTMENT) that our model is superior in representing contents at varying scales and viewing distances.

Table 2. **Performance comparison.** Rendering FPS and storage size are reported. Storage size reduction ratio is indicated by (↓). Rendering speed of both methods are measured on our machine.

Dataset	Mip-NeRF360		Tanks&Temples		Deep Blending	
	FPS	Mem (MB)	FPS	Mem (MB)	FPS	Mem (MB)
3D-GS	97	721	**123**	411	109	676
Ours	**102**	**171** (4.2× ↓)	110	**87** (4.7× ↓)	**139**	**66** (10.2× ↓)

vations, fine-scale details and view-dependent light effects. See Fig. 1 and Fig. 4 for examples.

Comparisons. Quality assessment on real-world datasets are presented in Tab. 1. Baselines' metrics align with those reported in the 3D-GS study. It can be noticed that our approach achieves comparable results with the SOTA algorithms on Mip-NeRF360 dataset, and surpassed the SOTA on Tanks&Temples and DeepBlending, which captures more challenging environments with the presence of *e.g.* changing lighting, texture-less regions and reflections. In terms of efficiency, we evaluated rendering speed and storage size of our method and 3D-GS, as shown in Tab. 2. Our method achieved real-time rendering while using less storage, indicating that our model is more com-

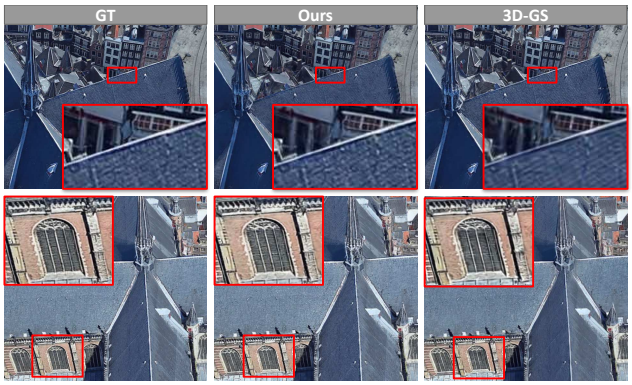

Figure 5. **Comparison on multi-scale scenes (w/ zoom-in cases).** Rendering at an unseen closer scale on the AMSTERDAM scene from BungeeNeRF. Our method smoothly extrapolates to new viewing distances using refined neural Gaussian properties, remedying the needle-like artifacts of original 3D-GS caused by fixed Gaussian scaling values.

pact than 3D-GS without sacrificing rendering quality and speed. Additionally, akin to prior grid-based methods, our approach converged faster than 3D-GS. See supplementary material for more analysis.

We also examined our method on the synthetic Blender dataset, which provides an exhaustive set of views capturing objects at 360°. Following 3D-GS, we use random points to initialize the anchors. The PSNR score and storage size comparisons presented in Tab. 3. Fig. 1 also demonstrate that our method can achieve better visual quality with more reliable geometry and texture details.

Multi-scale Scene Contents. We examined our model's capability in handling multi-scale scene details on the BungeeNeRF and VR-NeRF datasets. As shown in Tab. 3, our method achieved superior quality whilst using fewer storage size. As illustrated in Fig. 4 and Fig. 5, our method was superior in accommodating varying levels of detail in the scene. In contrast, images rendered from 3D-GS often suffered from noticeable blurry and needle-shaped artifacts. This is likely because that 3D-GS tends to overfit multi-scale training views, creating excessive Gaussians that work for each observing distance. However, it can easily lead to ambiguity and uncertainty when synthesizing novel views, since it lacks the ability to reason about viewing angle and distance. On contrary, our method efficiently encoded local structures into compact neural features, enhancing both rendering quality and convergence speed. Details are provided in the supplementary material.

Feature Analysis. We further perform an analysis of the learnable anchor features and the selector mechanism. As depicted in Fig. 6, the clustered pattern suggests that the compact anchor feature spaces adeptly capture regions with similar visual attributes and geometries, as evidenced by their proximity in the encoded feature space.

Table 3. **Qualitative comparison.** Our method is able to handle large-scale scenes (*e.g.* BUNGEENERF) with light-weight representation. Our method shows consistent compactness and effectiveness in complex lighting conditions and synthetic scenes.

Dataset	BungeeNeRF		VR-NeRF		Synthetic Blender	
	PSNR	Mem (MB)	PSNR	Mem (MB)	PSNR	Mem (MB)
3D-GS	24.89	1606	28.94	263	33.32	53
Ours	**27.01**	**203** (7.9× ↓)	**29.24**	**69** (3.8× ↓)	**33.68**	**14** (3.8× ↓)

Figure 6. **Anchor feature clustering.** We cluster anchor features (DB-PLAYROOM) into 3 clusters using K-means [25] and visualize the result. The clustered features show clues of scene contents, *e.g.* the banister, stroller, desk and monitor can be clearly identified. Anchors on the wall and floor are also respectively grouped together. This shows that our approach improves the interpretability of 3D-GS model, and has the potential to be scaled-up on much larger scenes exploiting reusable features.

Table 4. **Effects of filtering**. FILTER 1: view frustum culling; FILTER 2: the opacity-based selection. The filtering method has no notable impact on fidelity, but greatly affects inference speed.

Scene	DB-PLAYROOM		DB-DRJOHNSON	
	PSNR	FPS	PSNR	FPS
NO FILTERS	30.4	84	29.7	79
FILTER 1	30.3	118	29.6	100
FILTER 2	30.6	109	29.7	104
FULL	31.07	150	29.79	129

View Adaptability. To support that our neural Gaussians are view-adaptive, we explore how the values of attributes change when the same Gaussian is observed from different positions. Fig. 8 demonstrates a view-dependent distribution of attributes intensity, which maintains a degree of local continuity. This accounts for the superior *view adaptability*

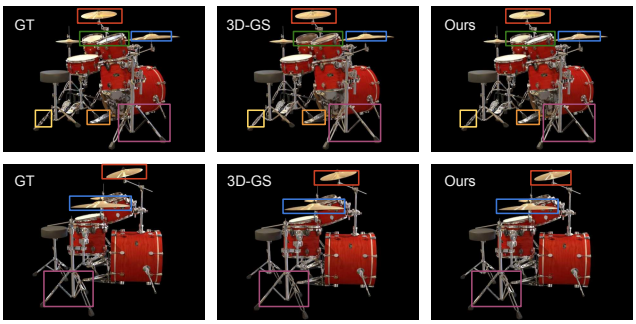

Figure 7. Our cross-view performance in specular reflection, leveraging regularized structures and view adaptability (Better viewed in zoom-in).

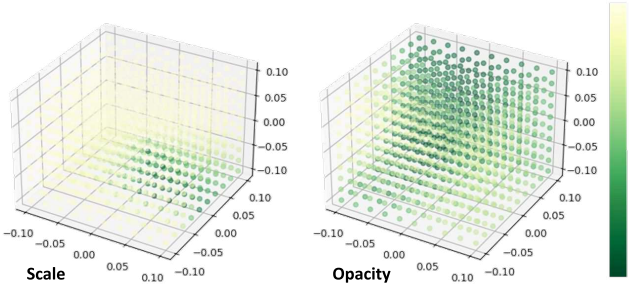

Figure 8. **View-adaptive neural Gaussian attributes.** Decoded attributes of a *single* neural Gaussian observed at different positions. Each point corresponds to a viewpoint in space. The color of the point denotes the intensity of attributes decoded for this view (left: $F_s \rightarrow s_i$; right: $F_\alpha \rightarrow \alpha_i$). This pattern indicates that attributes of a neural Gaussian adapt to viewpoint changing, while exhibiting a certain degree of local continuity.

of our method compared to the static attributes of 3D-GS, as well as its enhanced generalization to novel views. Cross-view comparisons in Fig. 7 shows that the view adaptability produces better performance in specular reflection.

4.3. Ablation Studies

Efficacy of Filtering Strategies. We evaluated our filtering strategies (Sec. 3.2.2), which we found crucial for speeding up our method. As Tab. 4 shows, while these strategies had no notable effect on fidelity, they significantly enhanced inference speed.

Efficacy of Anchor Points Refinement Policy. We evaluated our growing and pruning operations described in Sec. 3.3. Tab. 5 shows the results of disabling each operation in isolation and maintaining the rest of the method. We found that the addition operation is crucial for accurately reconstructing details and texture-less areas, while the pruning operation plays an important role in eliminating trivial Gaussians and maintaining the efficiency of our approach.

Table 5. **Anchor refinement**. The growing operation is essential for fidelity since it improves the poor initialization. The pruning operation controls the increasing of storage size and optimizes the quality of remained anchors.

Scene	DB-PLAYROOM		DB-DRJOHNSON	
	PSNR	Mem (MB)	PSNR	Mem (MB)
NONE	28.45	24	28.81	12
W/ PRUNING	29.12	23	28.51	12
W/ GROWING	30.54	71	29.75	76
FULL	31.07	61	29.79	50

4.4. Discussions and Limitations

Through our experiments, we found that the initial points play a crucial role for high-fidelity results. Initializing our framework from SfM point clouds is a swift and viable solution, considering these point clouds usually arise as a byproduct of image calibration processes. However, this approach may be suboptimal for scenarios dominated by large texture-less regions. Despite our anchor point refinement strategy can remedy this issue to some extent, it still suffers from extremely sparse points. We expect that our algorithm will progressively improve as the field advances, yielding more accurate results. Further details are discussed in the supplementary material.

5. Conclusion

In this work, we introduce *Scaffold*-GS, a novel 3D neural scene representation for efficient view-adaptive rendering. The core of *Scaffold*-GS lies in its structural arrangement of 3D Gaussians guided by anchor points from SfM, whose attributes are on-the-fly decoded from view-dependent MLPs. We show that our approach leverages a much more compact set of Gaussians to achieve comparable or even better results than the SOTA algorithms. The advantage of our *view-adaptive* neural Gaussians is particularly evident in challenging cases where 3D-GS usually fails. We further show that our anchor points encode local features in a meaningful way that exhibits semantic patterns to some degree, suggesting its potential applicability in a range of versatile tasks such as large-scale modeling, manipulation and interpretation in the future.

Acknowledgements

This work is funded in part by the National Key R&D Program of China (2022ZD0160201), Shanghai Artifical Intelligence Laboratory, and the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission (ITC)'s InnoHK.

References

[1] Kara-Ali Aliev, Dmitry Ulyanov, and Victor S. Lempitsky. Neural point-based graphics. In *European Conference on Computer Vision*, 2019. 3

[2] 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. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844, 2021. 2

[3] 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 (ICCV)*, pages 5855–5864, 2021. 1

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 2, 5, 6

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19697–19705, 2023. 1

[6] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's gpus. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 17–141. IEEE, 2005. 1

[7] Mario Botsch, Alexander Sorkine-Hornung, Matthias Zwicker, and Leif P. Kobbelt. High-quality surface splatting on today's gpus. *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 17–141, 2005. 3

[8] Eric Chan, Connor Z. Lin, Matthew Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, S. Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16102–16112, 2021. 2

[9] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *ArXiv*, abs/2203.09517, 2022. 2

[10] Anpei Chen, Zexiang Xu, Xinyue Wei, Siyu Tang, Hao Su, and Andreas Geiger. Factor fields: A unified framework for neural fields and beyond. *ArXiv*, abs/2302.01226, 2023. 2

[11] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *The Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[12] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. *ArXiv*, abs/1604.00449, 2016. 2

[13] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2, 5, 6

[14] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12479–12488, 2023. 2

[15] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas A. Funkhouser. Local deep implicit functions for 3d shape. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4856–4865, 2019. 2

[16] Markus Gross and Hanspeter Pfister. *Point-based graphics*. Elsevier, 2011. 3

[17] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. 37(6):257:1–257:15, 2018. 6

[18] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. 5

[19] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul E. Debevec. Baking neural radiance fields for real-time view synthesis. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, 2021. 2

[20] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 3

[21] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *ArXiv*, abs/1708.05375, 2017. 2

[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), 2023. 2, 3, 4, 5, 6

[23] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 5, 6

[24] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. *Computer Graphics Forum*, 40, 2021. 3

[25] K Krishna and M Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, 1999. 7

[26] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1440–1449, 2021. 1

[27] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 3

[28] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (ToG)*, 40(4):1–13, 2021. 5

[29] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4455–4465, 2018. 2

[30] 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. 1, 2, 5

[31] 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. 2, 5, 6

[32] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8280–8290, 2022. 1

[33] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5569–5579, 2021. 2

[34] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 2

[35] Songyou Peng, Michael Niemeyer, Lars M. Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *ArXiv*, abs/2003.04618, 2020. 2

[36] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023. 1

[37] Liu Ren, Hanspeter Pfister, and Matthias Zwicker. Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering. *Computer Graphics Forum*, 21, 2002. 3

[38] Miguel Sainz and Renato Pajarola. Point-based rendering techniques. *Computers & Graphics*, 28(6):869–879, 2004. 3

[39] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[40] Jessica Shue, Eric Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20875–20886, 2022. 2

[41] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Frédo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Neural Information Processing Systems*, 2021. 2

[42] Noah Snavely, Steven M. Seitz, and Richard Szeliski. *Photo Tourism: Exploring Photo Collections in 3D*. Association for Computing Machinery, New York, NY, USA, 1 edition, 2023. 2

[43] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. 1

[44] Shubham Tulsiani, Tinghui Zhou, Alyosha A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 209–217, 2017. 2

[45] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12922–12931, 2022. 1

[46] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 2

[47] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 5

[48] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7465–7475, 2019. 3

[49] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *The European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 5

[50] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Bo Dai, and Dahua Lin. Assetfield: Assets mining and reconfiguration in ground feature plane representation. *ArXiv*, abs/2303.13953, 2023. 2

[51] Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, Dahua Lin, Michael Zollhöfer, and Christian Richardt. VR-NeRF: High-fidelity virtualized walkable spaces. In *SIGGRAPH Asia Conference Proceedings*, 2023. 1, 5, 6

[52] Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8296–8306, 2023. 2

[53] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5438–5448, 2022. 3, 4

[54] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 2

[55] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019. 1, 3

[56] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[57] Xi Zhao, Ruizhen Hu, Haisong Liu, Taku Komura, and Xinyu Yang. Localization and completion for 3d object interactions. *IEEE Transactions on Visualization and Computer Graphics*, 26(8):2634–2644, 2019. 2

[58] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 3