西北工业大学

数字图像处理–论文翻译

原论文标题：

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

冯梓瑜

数学与统计学院

计算机科学与技术

2024 年 11 月

学号：2021303984

Northwestern Polytechnical University

用于视图自适应渲染的结构化三维高斯图

图1中，Scaffold-GS使用一组具有连续层次结构的三维高斯结构来表示场景。在固定了初始点的稀疏网格上，从每个锚点衍生出一组合适的神经高斯函数，以动态地适应不同的视角和距离。本方法通过一个更紧凑的模型实现了与3D高斯相当的渲染质量和速度（最后面的行单位：PSNR/storage size/FPS）。在多个数据集上，Scaffold-GS在大型户外场景和复杂的室内环境中表现出更强的鲁棒性，如透明度、镜性、无纹理区域和精细尺度细节。

摘要

神经渲染方法在各种学术和工业应用中显著提高了逼真的三维场景渲染技术。最近的三维高斯分割方法结合了基于原语的表示和体积表示的优点，实现了最先进的渲染质量和速度。然而，它经常导致大量冗余的高斯分布，试图适应每个训练视图，而忽略了底层的现场几何测量。因此，生成的模型对显著的视图变化、无纹理区域和灯光效果没有鲁棒性。接下来我们将介绍Scaffold-GS，它使用一个锚点来分布局部3D高斯函数，并根据视野内的观察方向和距离来确定他们的属性。锚点生长和修建策略是基于神经高斯分布的权重，以可靠地提高场景覆盖率。实验表明，我们的方法在提供高质量渲染的同时有效地再现了高斯分布。我们还展示了它的一种更强的能力，它能够适应具有不同细节层次和视点依赖性观察的场景，而不会牺牲渲染速度。

1. 介绍

3D场景的逼真和实时渲染一直是学术研究和工业领域的核心兴趣，其应用范围涵盖了虚拟现实[51]、媒体生成[36]以及大规模场景可视化[43, 45, 49]。传统的基于原始元素的表示方法，如网格和点[6, 26, 32, 55]，由于使用了为现代GPU量身定制的光栅化技术，因此渲染速度更快。然而，它们通常会产生低质量的渲染效果，表现出不连续性和模糊伪影。相比之下，体素表示和神经辐射场利用基于学习的参数模型[3, 5, 30]，因此能够产生连续的渲染结果，保留更多的细节。尽管如此，它们需要耗时的随机采样，导致性能下降和潜在的噪声。

最近，3D高斯点云投射（简称3D-GS）[22] 在渲染质量和速度上都达到了最先进的水平。这种方法从由运动恢复结构（简称SfM）[42] 派生的点云初始化，通过优化一组3D高斯来表示场景。它保留了体素表示中固有的连续性，同时通过将3D高斯投射到2D图像平面上，促进了快速的光栅化处理。

虽然这种方法提供了几个优势，但它倾向于过度扩展高斯球以适应每一个训练视图，从而忽略了场景结构。这导致了显著的冗余，并限制了其可扩展性，特别是在复杂的大规模场景

的背景下。此外，视图依赖性效果被固化到个别高斯参数中，几乎没有插值能力，使其对大幅度视图变化和光照效果的鲁棒性较低。

我们提出了 Scaffold-GS，这是一种基于高斯的方法，利用锚点建立分层和区域感知的三维场景表示。我们从 SfM 点出发，构建了一个稀疏的锚点网格。 每个锚点都绑定一组具有可学习偏离集的神经高斯，其属性（即不透明度、颜色、旋转、比例）可根据锚点特征和观察位置进行动态预测。 与允许三维高斯自由漂移和分裂的普通三维高斯不同，我们的策略是利用场景结构来引导和限制三维高斯的分布，同时允许它们局部适应不同的观察角度和距离。 我们进一步开发了相应的锚点生长和修剪操作，以增强场景覆盖范围。

在推理时，我们将神经高斯的预测范围限制在视锥内的锚点，并通过过滤步骤（即可学习的选择）根据神经高斯的不透明度过滤掉琐碎的神经高斯。因此，我们的方法能以与原始 3D-GS 相似的速度（1K 分辨率下约 100 FPS）进行渲染，而计算开销很小。 此外，由于我们只需存储每个场景的锚点和 MLP 预测器，因此存储需求大大降低。

总之，我们的贡献在于 1) 利用场景结构，我们从稀疏的体素网格中启动锚点，引导局部三维高斯的分布，形成分层和区域感知的场景表示；2) 在视场中，我们即时预测每个锚点的神经高斯，以适应不同的视角和距离，从而实现更稳健的新视图合成；3) 我们开发了一种更可靠的锚点增长和剪枝策略，利用预测的神经高斯实现更好的场景覆盖。

2. 相关工作

基于 MLP 的神经场和渲染。

早期的神经场通常采用多层感知器（MLP）作为三维场景几何和外观的全局近似器。 它们直接使用空间坐标（和观察方向）作为 MLP 的输入，并预测点的位置，如与场景表面的签名距离（SDF）[33,34, 46, 54]，或该点的密度和颜色[2, 30, 49]。 由于其体积特性和 MLP 的归纳偏差，这一系列方法在新颖的视图合成中实现了 SOTA 性能。这种场景再现的主要挑战在于，MLP 需要在沿每条摄像机光线的大量采样点上进行评估。 因此，渲染速度变得极其缓慢，对复杂和大规模场景的可扩展性有限。尽管已经有几项研究提出了加速或减轻密集体积光线行进的方法，例如使用提案网络[4]、烘焙技术[11,19]和曲面渲染[41]。 它们要么加入了更多的 MLP，要么以渲染质量换取速度。

基于网格的神经场和渲染。

这类场景表示通常基于密集均匀的体素网格。 它们被广泛应用于三维形状和几何建模[12、15、21、29、35、44、57]。 最近的一些方法也侧重于通过利用空间数据结构来存储场景特征，并在光线行进过程中对采样点进行插值和查询，从而加快辐射场的训练和对比度。在实际应用中，Plenoxel[13] 采用稀疏体素网格对连续密度场进行内插，并用球形谐波表示视角相关的视觉效果。 为了进一步减少数据冗余并加快渲染速度，多部作品[9,10,50,52]对张量

因子化的理念进行了研究。 K 平面[14]使用神经平面对三维场景进行参数化，还可选择使用额外的临时平面来适应动态变化。 一些生成性作品[8,40]也利用三平面结构对三维潜空间进行建模，以获得更好的几何一致性。 Instant- NGP [31] 使用哈希网格，大大加快了特征查询速度，实现了神经辐射场的实时渲染。 虽然这些方法可以生成高质量的结果，而且比全局 MLP 表示更高效，但它们仍然需要查询很多样本才能渲染一个像素，而且难以有效地表示空白空间。

基于点的神经场和渲染。

基于点的表示法利用几何基元（即点云）进行场景渲染。 典型的程序是使用固定大小对非结构化的点集进行渲染，并利用 GPU 和图形应用程序接口上的专门模块进行渲染 [7, 37, 38]。尽管这种方法在解决拓扑变化方面速度快、灵活性强，但通常会出现漏洞和异常值，导致渲染效果不佳。为了解决不连续性问题，人们对基于可变点的渲染技术进行了广泛研究，以建立物体的几何模型 [16, 20, 27, 48, 55]。 特别是，[48, 55]使用了可微分曲面拼接技术，将点基元视为大于像素的圆盘、椭圆或曲面。文献[1,24]使用神经特征增强点，并使用二维 CNN 进行渲染。 作为比较，Point-NeRF [53] 利用三维体绘制以及区域生长和优化过程中的点修剪，实现了高质量的新型视图合成。不过，他们采用了体积光线行进技术，因此影响了显示率。值得注意的是，最近的研究成果 3D-GS [22]采用了从运动结构（SfM）初始化的各向异性三维高斯来表示三维场景。 由于 3D-GS 使用 α 融合器整合了像素颜色，因此能生成具有细微尺度细节的高质量结果，并能以实时帧速率进行渲染。

3. 实现方法

最初的 3D-GS 模型[22]通过启发式拆分和剪枝操作，优化高斯来重新构建每个训练视图，但总体上忽略了底层场景结构。 这通常会导致高斯高度冗余，使模型对新的视角和距离的鲁棒性降低。 为了解决这个问题，我们提出了一种尊重场景几何结构的分层三维高斯场景表示法，通过从 SfM 初始化锚点来编码局部场景信息并生成局部神经高斯。神经高斯的物理特性是通过学习到的锚点特征以视图相关的方式即时解码的。 图 2 展示了我们的工作框架。我们首先简要介绍 3D-GS 的背景，然后详细介绍我们提出的方法。 第 3.2.1 节介绍了如何使用不规则 SfM 点云中的规则稀疏网格初始化场景。3.2.2 节解释了我们如何根据锚点和视图相关信息预测神经高斯特性。为了使我们的方法对噪声初始化更具鲁棒性，第 3.3 节介绍了基于神经高斯的 “生长 ”和 “剪枝 ”操作，以完善锚点。 第 3.4 节详细说明了训练细节。

3.1 前期准备工作

3D-GS [22]用一组各向异性的三维高斯来表示场景，这些高斯继承了体积表示法的差异特性，并通过基于瓦片的光栅化技术进行高效渲染。

从一组运动结构（SfM）点开始，每个点都被指定为三维高斯的位置（平均值）μ：

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

其中，x 是三维场景中的任意位置，Σ 表示三维高斯的协方差矩阵。 Σ 采用缩放矩阵 S 和旋转矩阵 R 来保持其正半定式：

$$\Sigma = RSS^TR^T, \quad (2)$$

除了用球形谐波建模的颜色 c 外，每个三维高斯还与不透明度 α 相关联，在混合过程中，不透明度会与 G(x) 相乘。

有别于传统的体积表示法，3D-GS 通过基于瓦片的光栅化来有效渲染场景，而不是耗费大量资源的光线行进。 三维高斯 G(x) 首先按照 [58] 中描述的投影过程在图像平面上转换为二维高斯 G′ (x)。 然后设计一个基于瓦片的光栅器，对二维高斯进行有效分类，并采用 α 混合：

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

其中，x′ 是查询到的像素位置，N 表示与查询到的像素相关的已排序二维高斯的数量。利用可微分光栅器，三维高斯的所有属性都是可学习的，并通过训练视图重构直接进行端到端优化。

3.2 Scaffold-GS

3.2.1 锚点初始化

与现有方法 [22,53] 一致，我们使用 COLMAP [39] 中的稀疏点云作为初始输入。 然后，我们根据该点云 P∈ RM ×3 对场景进行体素化处理：

$$\mathbf{V} = \left\{ \left\lfloor \frac{\mathbf{P}}{\epsilon} \right\rfloor \right\} \cdot \epsilon, \quad (4)$$

其中 V∈RN ×3 表示体素中心，ϵ 是体素大小。 然后，我们删除以 {-} 表示的重复条目，以减少 P 中的冗余和不规则性。

每个体素v∈v的中心被视为一个锚点，配备了一个局部上下文特征fv∈R32、一个比例因子1v∈R3和k个可学习偏移量Ov∈Rk×3。在对术语的滥用中，我们将在以下上下文中表示锚点。我们进一步增强了fvo是多分辨率和依赖于视图的。对于每个锚点v，我们1)创建模型库{fv ，fv↓1，fv↓2}，其中↓表示fv被2n个因子降采样；2) 将
特征库与视图权重相融合，形成综合锚点特征 f^v. 具体来说，给定位置 xc 的摄像机和位置 xv 的锚点，我们用以下方法计算它们的相对距离和观察方向相对距离和观察方向：

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

然后将特征库与微型 MLP Fw 预测的权重加权求和：

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

$$\hat{\mathbf{f}}_v = w \cdot f_v + w_1 \cdot f_{v_{i_1}} + w_2 \cdot f_{v_{i_2}}, \quad (7)$$

3.2.2 神经高斯推导

在本节中，我们将详细阐述我们的方法是如何从锚点推导出神经高斯的。除非另有说明，F∗ 在本节中代表一个特定的 MLP。此外，我们还介绍了两种高效的预过滤策略，以减少 MLP 的开销。

我们用其position$\mu \in R3$、opacity$\alpha \in R$、协方差相关的四元数$q \in R4$和缩放$\in R3$和颜色$c \in R3$来参数化一个神经高斯分布。如图2(b)所示，对于视锥内的每个可见锚点，我们会生成 k 个神经高斯，并预测它们的属性。具体来说，给定位于x_v的锚点，其神经高斯位置计算为：

$$\{\mu_0, ..., \mu_{k-1}\} = \mathbf{x}_v + \{O_0, ..., O_{k-1}\} \cdot l_v, \quad (8)$$

其中$\{O0, O1, O1Ok-1\} \in Rk\times3$是学习偏移量，$l_v$是与锚相关的比例因子，如 Sec.3.2.1所述。k 个神经高斯的属性通过单个 MLPs（分别表示为 Fα、Fc、Fq 和 Fs）直接从锚点特征 $\hat{f}_v$、摄像机与锚点之间的相对观察距离 δ_{vc} 和方向 $\vec{d}_{vc}$ 解码（公式 5）。请注意，属性是一次性解码的。例如，从锚点生成的神经高斯的不透明度值为：

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

它们的颜色 {ci }、四元数 {qi } 和比例 {si } 也是类似推导出来的。实现细节见补充资料。

请注意，神经高斯属性的预测是即时进行的，也就是说，只有球面内可见的锚点才会被激活，从而产生神经高斯。为了提高光栅化效率，我们只保留不透明度值大于预定义阈值 τ_α 的神经高斯。这大大减少了计算负荷，帮助我们的方法保持了与原始 3D-GS 相当的高渲染速度。

3.3 锚点细化

生长操作。

由于神经高斯与由 SfM 点初始化的锚点密切相关，因此其建模能力仅限于局部区域，这一点在文献[22, 53]中已经指出。这给锚点的初始放置带来了挑战，尤其是在无纹理和观察较

少的区域。因此，我们提出了一种基于误差的锚点增长策略，在神经高斯认为重要的地方增长新锚点。为了确定重要区域，我们首先通过构建大小为 ϵ_g 的体素对神经高斯进行空间量化。对于每个体素，我们计算 N 次训练迭代中包含的神经高斯的平均梯度，记为 ∇_g。然后，$\nabla_g > \tau_g$ 的体素被视为重要体素，其中 τ_g 是预先定义的阈值；如果没有建立锚点，则在该体素的中心部署一个新锚点。图 3 展示了这种增长操作。在实践中，我们将空间量化为多分辨率体素网格，以便在不同粒度上添加新的锚点，其中：

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

其中 m 表示量化程度。为了进一步规范新锚点的添加，我们对这些候选锚点进行随机剔除。这种谨慎添加点的方法有效地抑制了锚点的快速扩张。

剪枝操作。

为了剔除琐碎的锚点，我们会在 N 次训练迭代中累积其相关神经高斯的不透明度值。如果一个锚点无法产生令人满意的神经高斯不透明度，我们就会将其从场景中删除。

3.4 损失设计

我们利用 SSIM 项 [47] LSSIM 和体积正则化 [28] Lvol，针对渲染像素颜色的 L1 损失优化可学习参数和 MLP：

$$L = L_1 + \lambda_{SSIM} L_{SSIM} + \lambda_{vol} L_{vol}, \qquad (11)$$

其中，体积正则化 Lvol 为：

$$L_{vol} = \sum_{i1}^{g} Prod(s_i). \qquad (12)$$

这里，Nng 表示场景中神经高斯的数量，Prod(-) 是一个向量值的乘积，例如，在我们的例子中，每个神经高斯的比例 si。体积正则化项可以使神经高斯最小化，重叠最小化。

4. 实验部分

4.1 实验环境设置

数据集和衡量标准。

我们对公开数据集中的27个场景进行了全面评估。具体来说，我们在 3D-GS [22] 中测试的所有可用场景上测试了我们的方法，包括来自 Mip-NeRF360 [4] 的七个场景、来自 Tanks&Temples [23] 的两个场景、来自 DeepBlending [18] 的两个场景和合成 Blender 数据集 [30]。此外，我们还对以多个 LOD 捕捉内容的数据集进行了评估，以展示我们在视图自适应渲染方面的优势。我们选择了来自 BungeeNeRF [49] 的六个场景和来自 VR-NeRF [51]

的两个场景。前者提供了多尺度室外观测数据，后者捕捉了错综复杂的室内环境。除了常用指标（PSNR、SSIM [47] 和 LPIPS [56]）外，我们还报告了存储大小（MB）和渲染速度（FPS），以简化模型并提高性能效率。我们在正文中提供了每个数据集所有场景的平均指标，并将每个场景的完整定量结果留在了补充材料中。

基准和实施。

3D-GS [22] 被选为我们的主要基准，因为它在新颖视图合成中具有公认的 SOTA 性能。3D-GS 和我们的方法都经过了30k次迭代训练。我们还记录了 MipNeRF360 [4]、iNGP [31] 和 Plenoxels [13] 的结果，与 [22] 相同。

对于我们的方法，我们在所有实验中都设置了 k = 10。我们方法中使用的所有 MLP 都是具有 ReLU 激活功能的 2 层 MLP；隐藏单元的维数均为 32。对于锚点细化，我们在 N = 100 次迭代中平均梯度，默认使用 $\tau_g = 64\epsilon$。在复杂场景和有主要无纹理区域的场景中，我们使用 $\tau_g = 16\epsilon$。如果在每一轮细化中，锚点神经高斯的累积不透明度小于 0.5，则该锚点将被剪枝。在我们的实验中，两个损失权重 λ_{SSIM} 和 λ_{vol} 分别设置为 0.2 和 0.001。更多详情请查阅补充材料。

4.2 结果分析

我们在各种数据集上进行了评估，从合成物体级场景、室内和室外环境，到大型城市场景和景观。特别是在一些具有挑战性的情况下，如无纹理区域、观测不足、精细尺度细节和视角依赖的光照效果等，我们可以观察到各种改进。示例见图 1 和图 4。

对比。

为了评估我们方法的质量，我们在真实世界数据集上与 3D-GS [22]、Mip-NeRF360 [4]、iNGP [31] 和 Plenoxels [13]进行了比较。定性结果见表 1。1. Mip-NeRF360、iNGP 和 Plenoxels 的质量指标与 3D-GS 研究中报告的质量指标一致。可以看出，我们的方法在 Mip-NeRF360 数据集上取得了与 SOTA 算法相当的结果，而在 Tanks&Temples 和 DeepBlending 数据集上则超过了 SOTA 算法，后者捕捉的是更具挑战性的环境，例如存在光照变化、无纹理区域和反射的环境。在效率方面，我们评估了我们的方法和 3D-GS 的渲染速度和存储大小，如表 2 所示。2. 我们的方法实现了实时渲染，同时使用了更少的存储空间，这表明我们的模型比 3D-GS 更为紧凑，而不会牺牲渲染质量和速度。此外，与之前基于网格的方法类似，我们的方法比 3D-GS 收敛得更快。更多分析，请参见补充材料。

我们还在合成的 Blender 数据集上检验了我们的方法，该数据集提供了一组详尽的视图，可捕捉 360 ◦ 范围内的物体。在该数据集中，我们无法轻易获得一组良好的初始 SfM 点，因此我们从 100k 个网格点开始，通过锚点细化操作来学习增长和修剪点。经过 30k 次迭代后，我们将剩余的点作为初始化锚点，并重新运行我们的框架。与 3DGS 相比，PSNR 分数和存储

空间大小见表 3。3. 图 1 也表明，我们的方法可以获得更好的视觉质量，几何和纹理细节更加可靠。

多尺度场景内容。

我们在 BungeeNeRF 和 VR-NeRF 数据集上检验了我们的模型处理多尺度场景细节的能力。如表 3 所示，与 3D-GS [22]相比，我们的方法在使用更少存储空间存储模型的同时，还获得了更高的质量。如图 4 和图 5 所示，我们的方法在适应场景的不同细节水平方面更胜一筹。相比之下，3D-GS 渲染的图像往往存在明显的模糊和针状伪影。这可能是因为高斯属性经过优化，过度适应了多尺度训练视图，从而产生了过多适用于每个观察距离的高斯。然而，由于缺乏对观察角度和距离的推理能力，在合成新视图时很容易导致模糊性和不确定性。相反，我们的方法能有效地将局部结构编码为紧凑的神经特征，从而提高了渲染质量和收敛速度。详情请参见补充材料。

特征分析。

我们进一步对可学习的锚点特征和选择器机制进行了分析。如图 6 所示，聚类模式表明，紧凑的锚点特征空间善于捕捉具有相似视觉属性和几何形状的区域，这一点从它们在编码特征空间中的接近程度可以看出。

视图适应性。

为了证明我们的神经高斯具有视图适应性，我们探索了当从不同位置观察同一个高斯时，属性值的变化情况。图 7 展示了不同观察位置下属性强度的不同分布，同时保持一定程度的局部连续性。这说明与 3D-GS 的静态属性相比，我们的方法具有更强的视角适应性，同时还增强了对新视角的普适性。

不透明度的选择过程。

我们从两个方面考察了神经高斯的不透明度解码和基于不透明度的选择过程（图 2(b)）。首先，在没有锚点细化模块的情况下，我们使用神经高斯的解码不透明度值进行过滤，从随机点云中提取几何图形。图 8 显示，残留的神经高斯有效地从随机点中重建了推土机模型的粗略结构，突出了其在主要基于渲染的监督下进行隐式几何建模的能力。我们发现这在概念上与 [4] 中使用的提案网络类似，作为几何代理估计器，从而实现高效采样。

其次，我们在方法中采用了不同的 k 值。图 9 显示，无论 k 值大小，最终激活的神经高斯数量在训练过程中都趋于相似。这表明 Scaffold-GS 更倾向于选择足以代表场景的非冗余高斯集合。

4.3 消融研究

过滤策略的有效性。

我们评估了我们的过滤策略（第 3.2.2 节），我们发现这些策略对加快我们的方法至关重要。如表 4 所示，虽然这些策略对保真度没有显著影响，但却大大提高了推理速度。然而，这些策略有可能会掩盖相关的神经高斯，我们希望在未来的工作中解决这个问题。

锚点完善政策的有效性。

我们对第 3.3 节所述的生长和修剪操作进行了评估。表 5 显示了单独禁用每种操作并保留方法其余部分的结果。我们发现加法操作对于准确重建细节和无纹理区域至关重要，而剪枝操作在消除琐碎高斯和保持我们方法的效率方面发挥着重要作用。

4.4 讨论和限制

通过实验，我们发现初始点对高保真结果起着至关重要的作用。考虑到这些点云通常是图像校准过程中产生的副产品，从 SfM 点云初始化我们的框架是一个快速可行的解决方案。然而，这种方法对于以大面积无纹理区域为主的场景可能不是最佳选择。尽管我们的锚点细化策略可以在一定程度上解决这个问题，但它仍然受到极度稀疏点的影响。我们希望，随着该领域的发展，我们的算法将逐步得到改进，从而产生更精确的结果。更多细节将在补充材料中讨论。

5. 总结

在这项工作中，我们介绍了 Scaffold-GS，一种用于高效视图自适应渲染的新型三维神经场景表示法。Scaffold-GS 的核心在于其由 SfM 锚点引导的三维高斯结构安排，其属性由视图相关 MLP 实时解码。我们的研究表明，我们的方法利用了更为紧凑的高斯集，取得了与 SOTA 算法相当甚至更好的效果。我们的视图自适应神经高斯的优势在 3D-GS 通常失效的挑战性案例中尤为明显。我们还进一步证明，我们的锚点编码局部特征的方式在一定程度上表现出语义模式，这表明它在未来的大规模建模、操作和解释等多功能任务中具有潜在的适用性。

用于视图自适应渲染的结构化三维高斯图

补充材料

6. 概述

本补充文件的结构如下 (1) 在第一部分中，我们阐述了 Scaffold-GS 的实现细节，包括锚点特征增强（第 3.2.1 节）、MLP 结构（第 3.2.2 节）和锚点细化策略（第 3.3 节）；(2) 第二部分介绍我们的数据集准备步骤。然后，我们将根据训练观察结果展示更多实验结果和分析。

7. 实施细节

特征库。

为了增强视角适应性，我们通过视角编码来更新锚点特征。在计算出摄像机和锚点的相对距离 δvc 和观察方向 $\vec{}$ dvc后，我们预测一个权重向量 w∈ R3 如下：

$$(w, w_1, w_2) = \text{Softmax}(F_w(\delta_{vc}, \vec{\mathbf{d}}_{vc})), \qquad (13)$$

其中，Fw 是作为视图编码函数的微小 MLP。然后，我们通过合成一个包含不同分辨率信息的特征库，将视图方向信息编码为锚点特征 fv，具体如下：

$$\hat{\mathbf{f}}_v = w \cdot f_v + w_1 \cdot f_{v_{\downarrow 1}} + w_2 \cdot f_{v_{\downarrow 2}}, \qquad (14)$$

在实践中，我们通过切片和重复来实现特征库，如图 10 所示。我们发现，这种切片和混合操作提高了 Scaffold-GS 捕捉不同场景粒度的能力。特征库权重的分布如图 11 所示。

用MLP 作为特征解码器。

核心 MLP 包括不透明度 MLP Fα、颜色 MLP Fc 以及协方差 MLP Fs 和 Fq。如图 12 所示，所有这些 F* 均以 LINEAR → RELU → LINEAR 方式实现，隐藏维度为 32。每个分支的输出由一个头部层激活。

对于不透明度，输出由 Tanh 激活，其中值 0 是选择有效样本的自然阈值，最终有效值可覆盖 [0,1) 的整个范围。

对于颜色，我们使用西格莫德函数激活输出：

$$\{c_0, \ldots, c_{k-1}\} = \text{Sigmoid}(F_c), \qquad (15)$$

它将颜色限制在 (0,1) 的范围内。

对于旋转，我们沿用 3D-GS [22]，并对其进行归一化激活，以获得有效的四元数。

在缩放方面，我们根据 MLP 的输出调整每个锚的基本缩放 sv，如下所示：

$$\{s_0, \ldots, s_{k-1}\} = \text{Sigmoid}(F_s) \cdot s_v, \qquad (16)$$

体素尺寸。

体素大小ϵ 设定了最精细的锚点分辨率。我们采用两种策略 1) 使用所有初始点近邻距离的中值： ϵ 与点云密度相适应，可产生更密集的锚点，提高渲染质量，但可能会带来更多的计算开销；2) 将ϵ 手动设置为 0.005 或 0.01：这在大多数情况下都很有效，但可能会导致无纹理区域的细节缺失。在实验中，我们发现这两种策略都能充分适应各种场景的复杂性。

锚点细化。

正如主论文中简要讨论的那样，体素化过程表明我们的方法可能会对初始 SfM 结果表现出敏感性。我们在图 13 中说明了锚点细化过程的效果，新锚点增强了场景细节，填补了大面积无纹理区域和观察较少区域的空白。

8. 实验结果

其他数据预处理。

我们使用COLMAP[39]来估计摄像机姿势，并为 VR-NeRF [51] 和 BungeeNeRF [49] 数据集生成 SfM 点。这两个数据集在以下方面都具有挑战性因为捕捉到的细节程度各不相同。对 VR-NeRF 数据集的测试使用的是 3 个摄像头的眼位子集。对于所有其他数据集，我们都采用了最初的 3D-GS [22]方法，数据来源于公共资源。

每个场景的结果。

在此，我们列出了第 4 节中评估所有方法和场景时使用的误差指标，如表 6-17 所示。6-17

训练过程分析。图 14 展示了训练视图和测试视图在训练过程中 PSNR 的变化。与 3D-GS 相比，我们的方法具有更快的收敛速度、更强的鲁棒性和更好的泛化能力，这一点可以从训练 PSNR 的快速增长和测试 PSNR 的提高得到证明。具体而言，对于 BungeeNeRF 中的阿姆斯特丹和蓬皮杜场景，我们使用三个较粗比例的图像对其进行了训练，并在一个新的较细比例下对其进行了评估。3D-GS 的训练 PSNR 较高，但测试 PSNR 较低，这表明它在训练尺度上有过拟合的倾向。

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

Tao Lu [1,3]* Mulin Yu[1]* Linning Xu[2] YuanboXiangli[4]
Limin Wang[1,3] 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

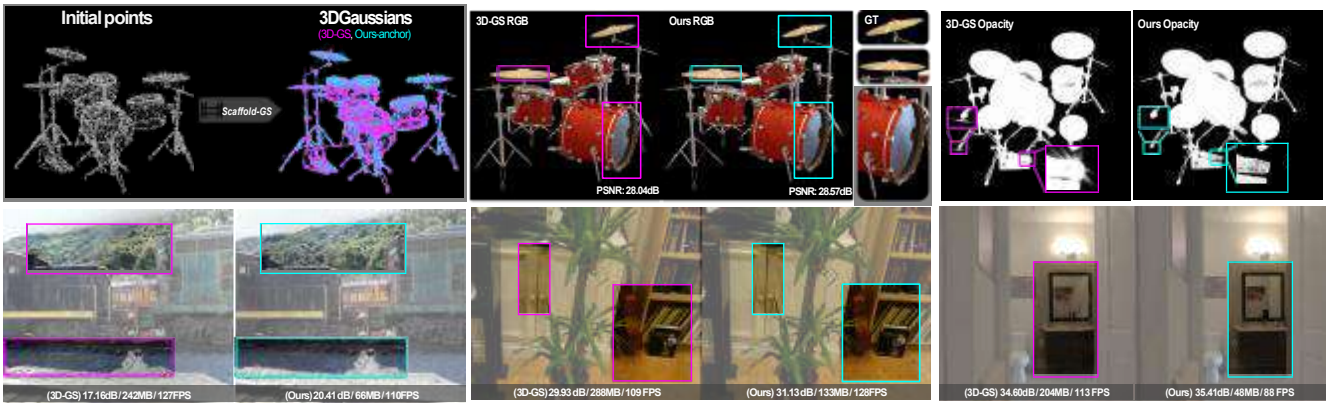

Figure 1. ***Scaffold*-GS** represents the scene using a set of 3D Gaussians structured in a dual-layered hierarchy. Anchored on a sparse grid of initial points, a modest set of neural Gaussians are spawned from each anchor to *dynamically adapt to* various viewing angles and distances. Our method achieves 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/.

* denotes equal contribution.

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 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 rasterize 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 flexibil-

ity 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-timeframe rate.

3. Methods

The original 3D-GS [22] optimizes Gaussians toreconstruct 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:

$$\Sigma = RSS^TR^T, \tag{2}$$

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 rasterizeris 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:

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

where $\mathbf{V} \in \mathbb{R}^{N\times 3}$ denotes voxel centers, and ϵ is the voxel size. 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; 2) blend the feature bank with view-dependent weights to form an integrated anchor feature $\hat{f}_v$. Specifically, Given a camera at

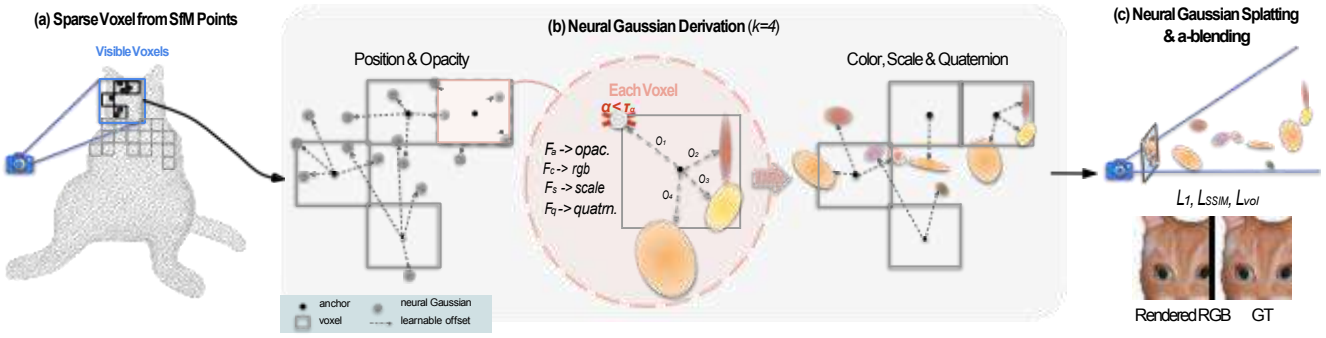

Figure 2. **Overview of *Scaffold*-GS**. (a) We start by forming a *sparse voxel grid* from SfM-derived points. An **anchor** associated with a learnable scale is placed at the center of each voxel, roughly sculpturing the scene occupancy. (b) Within a view frustum, k **neural Gaussians** are spawned from each *visible anchor* with offsets $\{O_k\}$. 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 (L_1),structural similarity (L_{SSIM}) and a volume regularization (L_{vol}).

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:

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

$$\hat{\mathbf{f}}_v = w \cdot f_v + w_1 \cdot f_{v_{\downarrow 1}} + w_2 \cdot f_{v_{\downarrow 2}}, \tag{7}$$

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 viewingfrustum, 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:

$$\{\mu_0, ..., \mu_{k-1}\} = \mathbf{x}_v + \{O_0, ..., O_{k-1}\} \cdot l_v, \tag{8}$$

where $\{O_0, O_1, ..., O_{k-1}\} \in \mathbb{R}^{k\times 3}$ 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 $\hat{\mathbf{f}}_v$, 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:

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

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 $\nabla_g > \tau_g$ is 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 anchor pointestablished. Fig. 3illustrates this growing operation. In practice, we quantize the space into multi-resolution voxel grid to al-

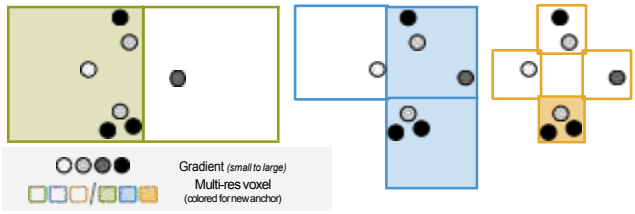

Figure 3. **Growing operation.** We develop an anchor growing policy guided by the gradients of the neural Gaussians. From left to right, we spatially quantize neural Gaussians into multi-resolution voxels ($m \in \{1, 2, 3\}$) of size $\{\epsilon_g^{(m)}\}$. New anchors are added to voxels with aggregated gradients larger than $\{\tau_g^{(m)}\}$.

low 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.

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.

3.4. Losses Design

We optimize the learnable parameters and MLPs with respect to the L_1 loss over rendered pixel colors, with SSIM term [47] L_{SSIM} and volume regularization [28] L_{vol}. The total supervision is given by:

$$L = L_1 + \lambda_{SSIM} L_{SSIM} + \lambda_{vol} L_{vol}, \tag{11}$$

where the volume regularization L_{vol} is:

$$L_{vol} = \sum_{i1}^{N_{ng}} Prod(s_i). \tag{12}$$

Here, N_{ng} denotes the number of neural Gaussians in the scene and $Prod(\cdot)$ 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 seven 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 $\tau_g = 64\epsilon$. 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 observations, fine-scale details and view-dependent light effects. See Fig. 1 and Fig. 4for examples.

Comparisons. In assessing the quality of our approach, we compared with 3D-GS [22], Mip-NeRF360 [4], iNGP [31] and Plenoxels [13] on real-world datasets. Qualitative results are presented in Tab. 1. The quality metrics for Mip-NeRF360, iNGP and Plenoxels 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

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

Dataset / Method | Metrics	Mip-NeRF360 PSNR ↑	Mip-NeRF360 SSIM ↑	Mip-NeRF360 LPIPS ↓	Tanks&Temples PSNR ↑	Tanks&Temples SSIM ↑	Tanks&Temples LPIPS ↓	Deep Blending PSNR ↑	Deep Blending SSIM ↑	Deep Blending LPIPS ↓
3D-GS [22]	28.69	0.870	0.182	23.14	0.841	0.183	29.41	0.903	0.243
Mip-NeRF360 [4]	29.23	0.844	0.207	22.22	0.759	0.257	29.40	0.901	0.245
iNPG [31]	26.43	0.725	0.339	21.72	0.723	0.330	23.62	0.797	0.423
Plenoxels [13]	23.62	0.670	0.443	21.08	0.719	0.379	23.06	0.795	0.510
Ours	28.84	0.848	0.220	23.96	0.853	0.177	30.21	0.906	0.254

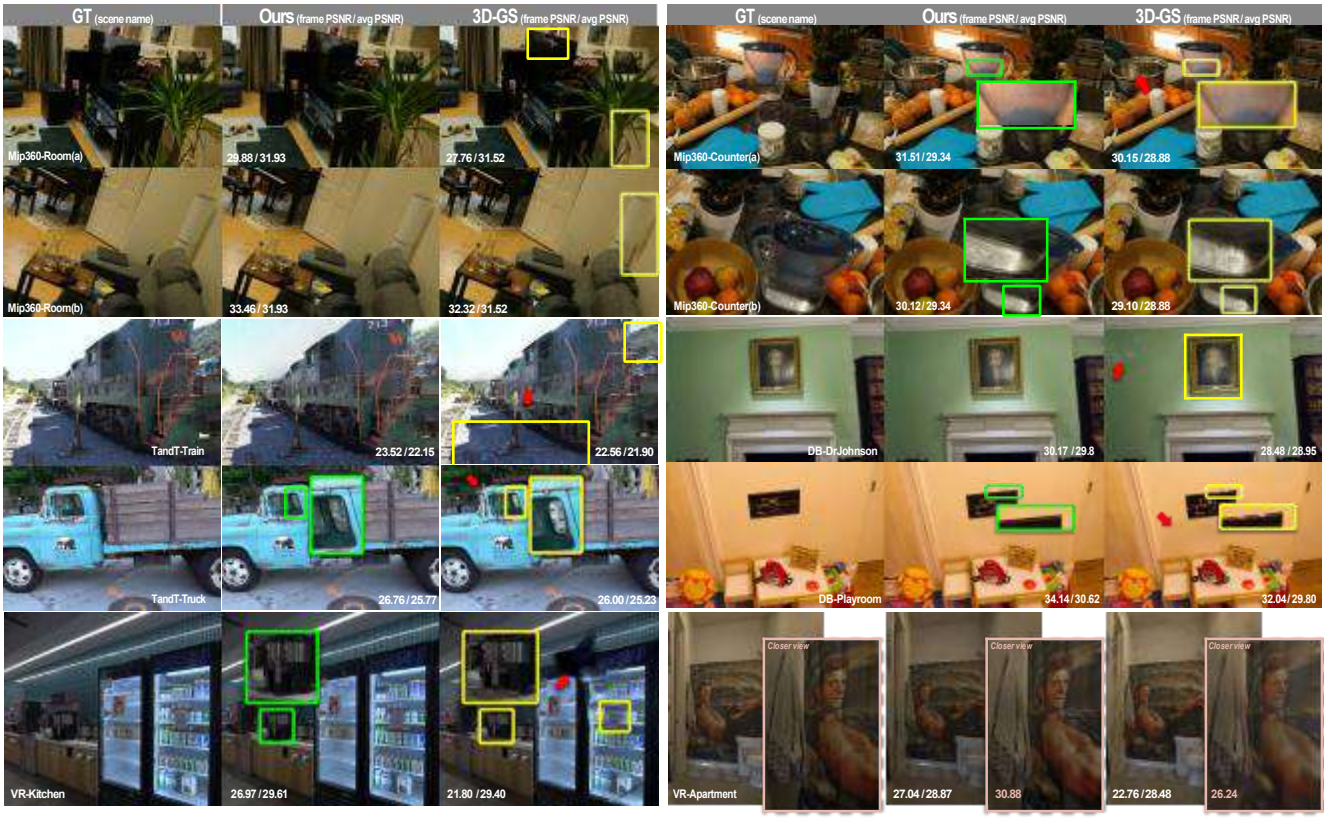

Figure 4. **Qualitative comparison of *Scaffold*-GS and 3D-GS [22] across diverse datasets [4, 17, 23, 51].** Patches that highlight the visual differences are emphasized with arrows and green & yellow insets for clearer visibility. Our approach constantly outperforms 3D-GS on these scenes, with evident 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 FPS	Mip-NeRF360 Mem (MB)	Tanks& Temples FPS	Tanks& Temples Mem (MB)	Deep Blending FPS	Deep Blending Mem (MB)
3D-GS	97	693	**123**	411	109	676
Ours	**102**	**156** (4.4× ↓)	110	**87** (4.7× ↓)	**139**	**66** (10.2× ↓)

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 compact 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° . A good set of initial SfM points is not readily available in this dataset, thus we start from 100k grid points and learn to grow and prune points with our anchor refinement operations. After 30k iterations, we used the re-

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× ↓)

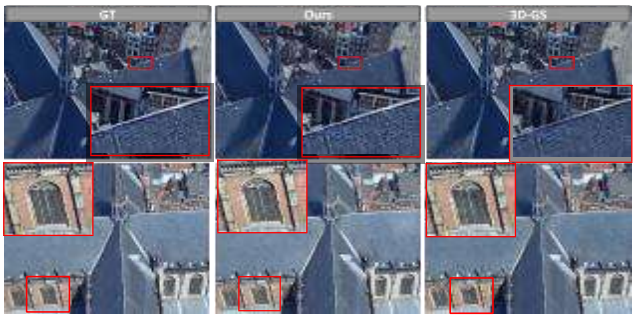

Figure 5. **Comparison on multi-scale scenes (w/ zoom-in cases).** We showcase the rendering outcomes at an unseen closer scale on the AMSTERDAM scene from BungeeNeRF. Our method smoothly extrapolates to new viewing distances using refined neural Gaussian properties, remedying the needle-like artifacts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

mained points as initialized anchors and re-run our framework. The PSNR score and storage size compared with 3D-GS are presented in Tab. 3. Fig. 1also demonstrates 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 to store the model compared to 3D-GS [22]. 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 Gaussian attributes are optimized 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.

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.

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.

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. 7 demonstrates a varying distribution of attributes intensity at different viewing positions, while maintaining a degree of local continuity. This accounts for the superior *view adaptability* of our method compared to the static attributes of 3D-GS, as well as its enhanced generalizability to novel views.

Selection Process by Opacity. We examine the decoded opacity from neural Gaussians and our opacity-based selection process (Fig. 2(b)) from two aspects. First, without the anchor point refinement module, we filter neural Gaussian using their decoded opacity values to extract geometry from a random point cloud. Fig. 8demonstrates that the remained neural Gaussians effectively reconstruct the coarse structure of the bulldozer model from random points, highlighting its capability for implicit geometry modeling under mainly rendering-based supervision. We found this is conceptually similar to the proposal network utilized in [4], serving as the geometry proxy estimator for efficient sampling.

Second, we apply different k values in our method.

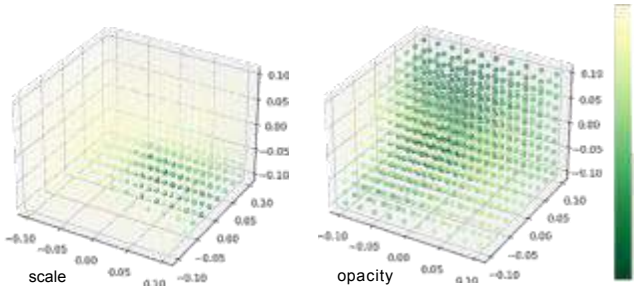

Figure 7. **View-adaptive neural Gaussian attributes.** We visualize the decoded attributes of a *single* neural Gaussian observed at different positions. Each point corresponds to a viewpoint in space. The color of the point denotes the intensity of attributes decoded for this view (left: $F_s \rightarrow s_i$; 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.

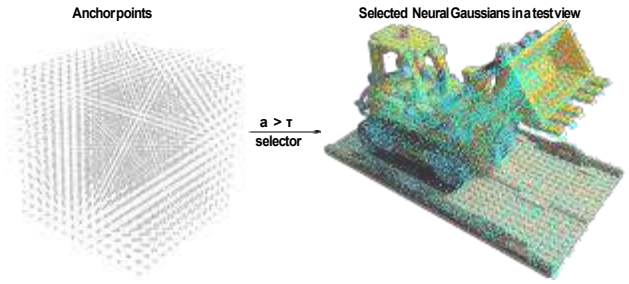

Figure 8. **Geometry culling via selector.** (Left) Anchor points from randomly initialized points; (Right) Activated neural Gaussians derived from each anchor under the current view. In synthetic Blender scenes, with all 3D Gaussians visible in the viewing frustum, our opacity filtering functions similar to a geometry proxy estimator, excluding unoccupied regions before rasterization.

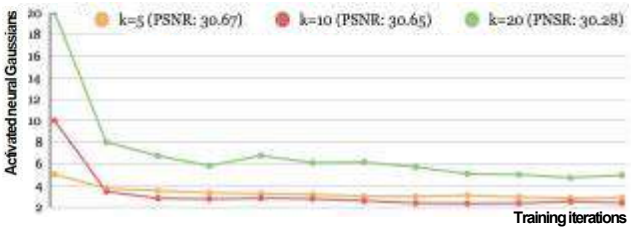

Figure 9. **Learning with different k values.** Despite varying initial k values under different hyper-parameter settings, they converge to activate a similar number of neural Gaussians with comparable rendering fidelity.

Fig. 9 shows that regardless of the k value, the finalnumber of activated neural Gaussians converges to a similar amount through the training, indicating *Scaffold*-GS's preference to select a collection of non-redundant Gaussians that are sufficient to represent the scene.

Table 4. **Effects of filtering**. FILTER 1 refers to selecting anchors by view frustum and FILTER 2 refers to the opacity-based selection process. 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	30.62	150	29.8	129

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	30.62	63	29.80	68

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. However, there was a risk of masking pertinent neural Gaussians, which we aim to address in future works.

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.

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 maybe 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.

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. 2,5,6,7,3

[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. 2

[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. 2,5,6,3

[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. 6,3

[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. 5,2,3

[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 Hane, 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. 1,3,4,5,6,7,2

[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. 5,6,2,3

[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, 2,5,3

[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. 2,5,6,3

[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. 2

[38] Miguel Sainz and Renato Pajarola. Point-based rendering techniques. *Computers & Graphics*, 28(6):869–879, 2004. 2

[39] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3,2

[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, SabeekPradhan, Ben Mildenhall, PratulP Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVFConference 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 *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXII*, pages 106–122. Springer, 2022. 1,2,5,3

[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. 1,5,6,2,3

[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 3dobject 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

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

Supplementary Material

6. Overview

This supplementary is organized as follows: (1) In the first section, we elaborate implementation details of our *Scaffold*-GS, including anchor point feature enhancement (Sec.3.2.1), structure of MLPs (Sec.3.2.2) and anchor point refinement strategies (Sec.3.3); (2) The second part describes our dataset preparation steps. We then show additional experimental results and analysis based on our training observations.

7. Implementation details.

Feature Bank. To enhance the view-adaptability, we update the anchor feature through a view-dependent encoding. Following calculating the relative distance δ_{vc} and viewing direction $\vec{\mathbf{d}}_{vc}$ of a camera and an anchor, we predict a weight vector $w \in R^3$ as follows:

$$(w, w_1, w_2) = \text{Softmax}(F_w(\delta_{vc}, \vec{\mathbf{d}}_{vc})), \quad (13)$$

where F_w is a tiny MLP that serves as a view encoding function. We then encode the view direction information to the anchor feature f_v by compositing a feature bank containing information with different resolutions as follows:

$$\hat{\mathbf{f}}_v = w \cdot f_v + w_1 \cdot f_{v\downarrow_1} + w_2 \cdot f_{v\downarrow_2}, \quad (14)$$

In practice, we implement the feature bank via slicing and repeating, as illustrated in Fig. 10. We found this slicing and mixture operation improves *Scaffold*-GS's ability to capture different scene granularity. The distribution of feature bank's weights is illustrated in Fig. 11.

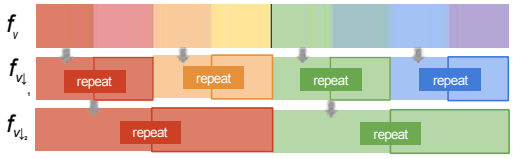

Figure 10. **Generation of Feature Bank.** We expand the anchor feature f into a set of *multi-resolution* features $\{f_v, f_{v\downarrow_1}, f_{v\downarrow_2}\}$ via slicing and repeating. This operation improves *Scaffold*-GS's ability to capture different scene granularity.

MLPs as feature decoders. The core MLPs include the opacity MLP F_α, the color MLP F_c and the covariance MLP F_s and F_q. All of these F_* are implemented in a LINEAR → RELU → LINEAR style with the hidden dimension of 32, as illustrated in Fig. 12. Each branch's output is activated with a head layer.

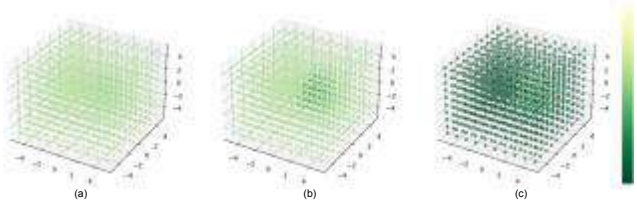

Figure 11. **View-based feature bank's weight distribution.** (a), (b) and (c) denote the predicted weights $\{w_2, w_1, w\}$ for $\{f_{v\downarrow_2}, f_{v\downarrow_1}, f_v\}$ from a group of uniformally distributed viewpoints. Light color denotes larger weights. For this anchor, finer features are more activated at center view positions. The patterns exhibit the ability to capture different scene granularities based on view direction and distance.

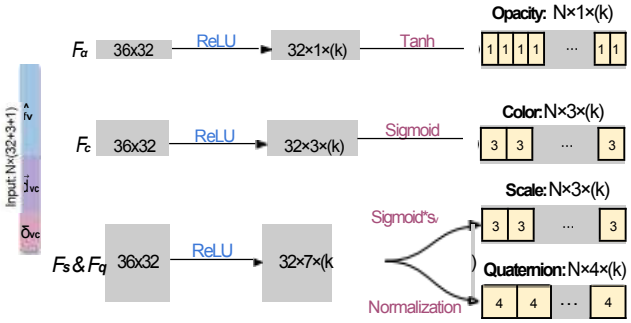

Figure 12. **MLP Structures.** For each anchor point, we use small MLPs (F_α, F_c, F_s,F_q) to predict attributes (opacity, color, scale and quaternion) of k neural Gaussians. The input to MLPs are anchor feature $\hat{\mathbf{f}}_v$, relative viewing direction $\vec{\mathbf{d}}_{vc}$ and distance δ_{vc} between the camera and anchor point.

- For *opacity*, the output is activated by Tanh, where value 0 serves as a natural threshold for selecting valid samples and the final valid values can cover the full range of [0,1).
- For *color*, we activate the output with Sigmoid function:

$$\{c_0, \ldots, c_{k-1}\} = \text{Sigmoid}(F_c), \quad (15)$$

which constrains the color into a range of (0,1).
- For *rotation*, we follow 3D-GS [22] and activate it with a normalization to obtain a valid quaternion.
- For *scaling*, we adjust the base scaling s_v of each anchor with the MLP output as follows:

$$\{s_0, \ldots, s_{k-1}\} = \text{Sigmoid}(F_s) \cdot s_v, \quad (16)$$

Voxel Size. The voxel size ϵ sets the finest anchor resolution. We employ two strategies: 1) Use the median of the nearest-neighbor distances among all initial points: ϵ

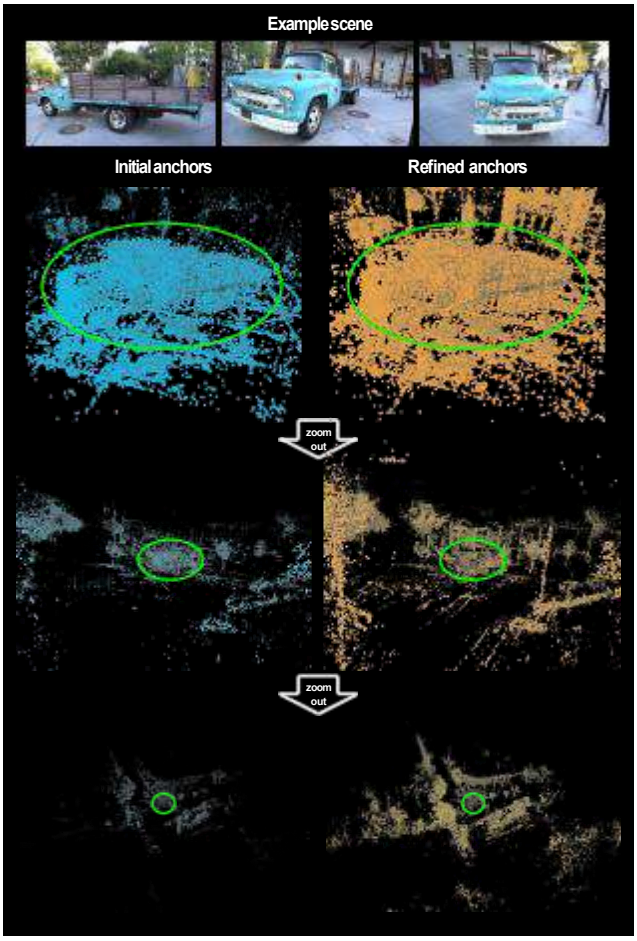

Figure 13. **Anchor Refinement.** We visualize the initial and refined anchor points on the truck scene [23]. The truck is highlighted by the circle. Note that the refined points effectively covers surrounding regions and fine-scale structures, leaning to more complete and detailed scene renderings.

is adapted to point cloud density, yielding denser anchors with enhanced rendering quality but might introduce more computational overhead; 2) Set ϵ manually to either 0.005 or 0.01: this is effective in most scenarios but might lead to missing details in texture-less regions. We found these two strategies adequately accommodate various scene complexities in our experiments.

Anchor Refinement. As briefly discussed in the main paper, the voxelization process suggests that our method may behave sensitive to initial SfM results. We illustrate the effect of the anchor refinement process in Fig. 13, where new anchors enhance scene details and fill gaps in large texture-less regions and less observed areas.

Table 6. **SSIM scores for Mip-NeRF360 [4] scenes.**

Method | Scenes	bicycle	garden	stump	room	counter	kitchen	bonsai
3D-GS [22]	**0.771**	**0.868**	0.775	0.914	0.905	0.922	0.938
Mip-NeRF360 [4]	0.685	0.813	0.744	0.913	0.894	0.920	0.941
iNPG [31]	0.491	0.649	0.574	0.855	0.798	0.818	0.890
Plenoxels [13]	0.496	0.6063	0.523	0.8417	0.759	0.648	0.814
Ours	0.705	0.842	**0.784**	**0.925**	**0.914**	**0.928**	**0.946**

Table 7. **PSNR scores for Mip-NeRF360 [4] scenes.**

Method | Scenes	bicycle	garden	stump	room	counter	kitchen	bonsai
3D-GS [22]	**25.25**	**27.41**	**26.55**	30.63	28.70	30.32	31.98
Mip-NeRF360 [4]	24.37	26.98	26.40	31.63	**29.55**	**32.23**	**33.46**
iNPG [31]	22.19	24.60	23.63	29.27	26.44	28.55	30.34
Plenoxels [13]	21.91	23.49	20.66	27.59	23.62	23.42	24.67
Ours	24.50	27.17	26.27	**31.93**	29.34	31.30	32.70

Table 8. **LPIPS scores for Mip-NeRF360 [4] scenes.**

Method | Scenes	bicycle	garden	stump	room	counter	kitchen	bonsai
3D-GS [22]	**0.205**	**0.103**	**0.210**	0.220	0.204	0.129	0.205
Mip-NeRF360 [4]	0.301	0.170	0.261	0.211	0.204	0.127	**0.176**
iNPG [31]	0.487	0.312	0.450	0.301	0.342	0.254	0.227
Plenoxels [13]	0.506	0.3864	0.503	0.4186	0.441	0.447	0.398
Ours	0.306	0.146	0.284	**0.202**	**0.191**	**0.126**	0.185

Table 9. **Storage size (MB) for Mip-NeRF360 [4] scenes.**

Method | Scenes	bicycle	garden	stump	room	counter	kitchen	bonsai
3D-GS [22]	1291	1268	1034	327	261	414	281
Ours	**248**	**271**	**493**	**133**	**194**	**173**	**258**

Table 10. **SSIM scores for Tanks&Temples [23] and Deep Blending [18] scenes.**

Method | Scenes	Truck	Train	Dr Johnson	Playroom
3D-GS [22]	0.879	0.802	0.899	**0.906**
Mip-NeRF360 [4]	0.857	0.660	0.901	0.900
iNPG [31]	0.779	0.666	0.839	0.754
Plenoxels [13]	0.774	0.663	0.787	0.802
Ours	**0.883**	**0.822**	**0.907**	0.904

8. Experiments and Results

Additional Data Preprocessing. We used COLMAP [39] to estimate camera poses and generate SfM points for VR-NeRF [51] and BungeeNeRF [49] datasets. Both two datasets are challenging in terms of varying levels of details presented in the captures. The VR-NeRF dataset was tested using its eye-level subset with 3 cameras. For all other datasets, we adhered to the original 3D-GS [22] method, sourcing them from public resources.

Per-scene Results. Here we list the error metrics used in our evaluation in Sec.4 across all considered methods and scenes, as shown in Tab. 6-17.

Training Process Analysis. Figure 14illustrates the variations in PSNR during the training process for both training and testing views. Our method demonstrates quicker

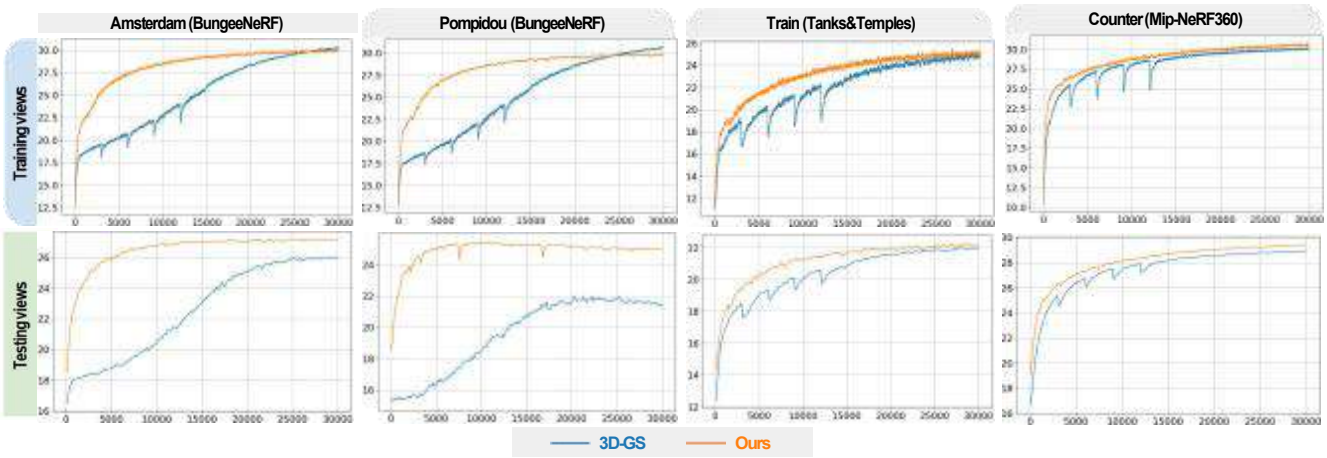

Figure 14. **PSNR curve of *Scaffold*-GS and 3D-GS [22] across diverse datasets [4,17,49].** We illustrate the variations in PSNR during the training process for both training and testing views. The orange curve represents *Scaffold*-GS, while the blue curve corresponds to 3D-GS. Our method not only achieves rapid convergence but also exhibits superior performance, marked by a significant rise in training PSNR and consistently higher testing PSNR, in contrast to 3D-GS.

Table 11. **PSNR scores for Tanks&Temples [23] and Deep Blending [18] scenes.**

Method	Scenes	Truck	Train	Dr Johnson	Playroom
3D-GS [22]		25.19	21.10	28.77	30.04
Mip-NeRF360 [4]		24.91	19.52	29.14	29.66
iNPG [31]		23.26	20.17	27.75	19.48
Plenoxels [13]		23.22	18.93	23.14	22.98
Ours		**25.77**	**22.15**	**29.80**	**30.62**

Table 12. **LPIPS scores for Tanks&Temples [23] and Deep Blending [18] scenes.**

Method	Scenes	Truck	Train	Dr Johnson	Playroom
3D-GS [22]		0.148	0.218	0.244	**0.241**
Mip-NeRF360 [4]		0.159	0.354	**0.237**	0.252
iNPG [31]		0.274	0.386	0.381	0.465
Plenoxels [13]		0.335	0.422	0.521	0.499
Ours		**0.147**	**0.206**	0.250	0.258

Table 13. **Storage size (MB) for Tanks&Temples [23] and Deep Blending [18] scenes.**

Method	Scenes	Truck	Train	Dr Johnson	Playroom
3D-GS [22]		578	240	715	515
Ours		**107**	**66**	**69**	**63**

Table 14. **PSNR scores for Synthetic Blender [30] scenes.**

Method	Scenes	Mic	Chair	Ship	Materials	Lego	Drums	Ficus	Hotdog
3D-GS [22]		35.36	**35.83**	30.80	30.00	**35.78**	26.15	34.87	37.72
Ours		**37.25**	35.28	**31.17**	**30.65**	35.69	**26.44**	**35.21**	**37.73**

convergence, enhanced robustness, and better generalization compared to 3D-GS, as evidenced by the rapid increase in training PSNR and higher testing PSNR. Specifically, for

Table 15. **Storage size (MB) for Synthetic Blender [30] scenes.**

Method	Scenes	Mic	Chair	Ship	Materials	Lego	Drums	Ficus	Hotdog
3D-GS [22]		50	116	63	35	78	93	59	44
Ours		**12**	**13**	**16**	**18**	**13**	**35**	**11**	**8**

Table 16. **PSNR scores for BungeeNeRF [49] and VR-NeRF [51] scenes.**

Method	Scenes	Amsterdam	Bilbao	Pompidou	Quebec	Rome	Hollywood	Apartment	Kitchen
3D-GS [22]		25.74	26.35	21.20	28.79	23.54	23.25	28.48	29.40
Ours		**27.10**	**27.66**	**25.34**	**30.51**	**26.50**	**24.97**	**28.87**	**29.61**

Table 17. **Storage size (MB) for BungeeNeRF [49] and VR-NeRF [51] scenes.**

Method	Scenes	Amsterdam	Bilbao	Pompidou	Quebec	Rome	Hollywood	Apartment	Kitchen
3D-GS [22]		1453	1337	2129	1438	1626	1642	202	323
Ours		**243**	**197**	**230**	**166**	**200**	**182**	**48**	**90**

the Amsterdam and Pompidou scenes in BungeeNeRF, we trained them with images at *three coarser scales* and evaluated them at a *novel finer scale*. The fact that 3D-GS achieved higher training PSNR but lower testing PSNR indicates its tendency to overfit at training scales.