

Scaffold-GS: Structured 3D gaussians for view-adaptive rendering

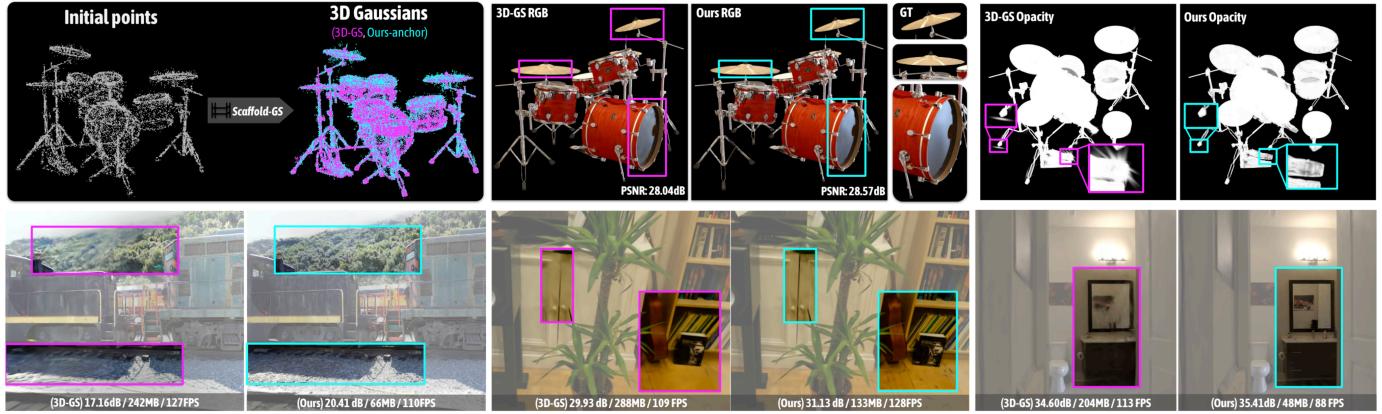


Figure 1. **Scaffold-GS** represents the scene using a set of 3D Gaussians structured in a dual-layered hierarchy. Anchored on a sparse grid of initial points, a modest set of neural Gaussians are spawned from each anchor to *dynamically adapt to various viewing angles and distances*. Our method achieves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

Introduction

在计算机视觉和图形学领域，真实感的3D场景渲染一直是重要的研究方向，广泛应用于虚拟现实（VR）、媒体生成和大规模场景可视化等场景。然而，传统的3D场景表示方法存在一些限制。

传统基于图元的表示（例如网格和点云）通过现代GPU优化的光栅化技术实现了快速渲染。然而，这些方法在质量上常常不尽如人意，例如表现为不连续性和模糊伪影。而基于体积的表示和神经辐射场（NeRF）方法利用基于学习的参数化模型，能够生成更精细的渲染结果。但这类方法存在计算开销高的问题，通常需要耗费大量时间进行随机采样。

最近的 **3D Gaussian Splatting (3D-GS)** 方法在渲染质量和速度上达到了先进水平。此方法基于从运动恢复 (SfM) 生成的点云初始化，通过优化一组3D高斯分布来表示场景。这种方法结合了体积表示的连续性优点和快速光栅化的能力。然而，3D-GS 在某些方面仍有不足：

1. **冗余性**：为了拟合每个训练视角，该方法会扩展高斯分布，导致模型冗余。
2. **鲁棒性差**：其对场景几何结构的建模较弱，尤其在视角变化大或光照条件复杂时表现不佳。

Scaffold-GS 的改进

本文提出了 **Scaffold-GS**，一种基于高斯的层次化3D场景表示方法：

1. 通过 **Anchor Points**（锚点）分布局部3D高斯分布；
2. 基于视角和距离动态预测高斯分布的属性；
3. 提出锚点的 **生长和修剪策略**，以提高场景覆盖范围和减少冗余。

方法核心

我们的方法在锚点基础上进行场景表示：

- 锚点初始化：从稀疏的SfM点云生成锚点并构建稀疏网格。
- 属性预测：通过多层感知机 (MLP) 根据视角和距离预测高斯的颜色、透明度、旋转和尺度。
- 生长和修剪：基于误差驱动策略调整锚点数量，提升模型性能。

公式描述：

1. 3D高斯分布的定义：

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

其中， x 为场景中任意位置， μ 为高斯分布的均值， Σ 为协方差矩阵。

2. 协方差矩阵的构造：

$$\Sigma = RSS^T R^T \quad (2)$$

其中， R 为旋转矩阵， S 为缩放矩阵，确保 Σ 为正半定。

3. 渲染过程中的透明度混合：

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

x' 为像素位置， α_i 为透明度， $G'_i(x')$ 为投影到二维的高斯分布。

主要贡献

1. 提出一种基于锚点的层次化3D场景表示方法。
2. 动态预测视角相关的高斯分布，提高模型的鲁棒性。
3. 提出了基于误差的锚点生长和修剪策略。

通过实验验证，我们的方法在渲染质量和速度上均优于现有的3D-GS方法，并显著减少存储需求。

Related Work

1. 基于多层感知机（MLP）的神经场与渲染

早期的神经场通常采用多层感知机（MLP）作为3D场景几何和外观的全局近似器。这些方法直接以空间坐标（以及视角方向）作为输入，预测点的属性，例如：

- 距离场签名（SDF）：

$$f(x) = \text{SignedDistance}(x, \text{surface}) \quad (4)$$

- 点的密度和颜色：

$$\rho(x), c(x) = \text{MLP}(\text{spatial_coordinates}, \text{viewing_direction}) \quad (5)$$

优点：

- 基于体积的特性和MLP的归纳偏置，这类方法在新视角合成方面表现卓越。

挑战：

- 渲染需要对每个相机光线上的大量采样点进行MLP评估，导致计算效率低，扩展性差。
- 尽管有研究尝试通过提案网络、预计算（baking）技术或表面渲染方法加速计算，这些优化通常会以牺牲渲染质量为代价。

2. 基于网格的神经场与渲染

基于网格的方法通常利用密集的均匀体素网格表示3D场景几何或特征。例如：

- **稀疏体素网格**（如 Plenoxel）：

$$f(x) = \text{Interpolate}(\text{voxel_grid}, x) \quad (6)$$

Plenoxel采用稀疏体素网格插值场景特征，并用球谐函数表示视角相关效果。

- **张量分解**方法进一步减少了数据冗余，加快了渲染速度。例如 K-planes 方法通过多平面（neural planes）表示3D场景。

优点：

- 显著提高训练和推理速度。
- 利用空间数据结构减少冗余。

不足：

- 在处理空空间和光线采样效率上仍存在挑战。

3. 基于点的神经场与渲染

点云是一种灵活的几何表示，适用于场景渲染。典型过程包括：

- 对点集进行光栅化，利用固定大小的点进行渲染。

为提高渲染质量，研究提出了可微的点云渲染技术，例如：

- 表面投影 (surface splatting)：将点表示为更大的几何体（如圆盘或椭球）。
- 基于神经特征的点增强：例如 Point-NeRF 使用3D体积渲染并通过区域生长和点修剪提高场景的表示能力。

挑战：

- 传统点云渲染易受不连续性和伪影影响，例如孔洞和离群点。

4. 基于高斯的场景表示

近期的 **3D Gaussian Splatting (3D-GS)** 方法在渲染质量和速度方面表现卓越。

其核心是利用各向异性3D高斯分布表示场景：

- **高斯定义：**

$$G(x) = e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)} \quad (7)$$

其中， μ 为高斯中心， Σ 为协方差矩阵，用于控制高斯的形状。

- **光栅化过程：** 将3D高斯投影到二维图像平面，进行透明度混合：

$$C(x') = \sum_{i \in N} c_i \sigma_i \prod_{j=1}^{i-1} (1 - \sigma_j), \quad \sigma_i = \alpha_i G'_i(x') \quad (8)$$

x' 为像素位置, c_i 为颜色, α_i 为透明度。

优势:

- 结合体积表示的连续性和快速光栅化能力。
- 生成高质量细节的实时渲染。

局限性:

- 缺乏对场景几何的建模, 导致冗余和扩展性问题。
- 对视角变化和光照效果的适应性较差。

综述

上述方法各有优势, 但在处理大规模复杂场景时仍存在瓶颈。本文提出的 **Scaffold-GS** 利用锚点和动态高斯分布预测策略, 在减少冗余的同时显著提高了渲染质量和鲁棒性。

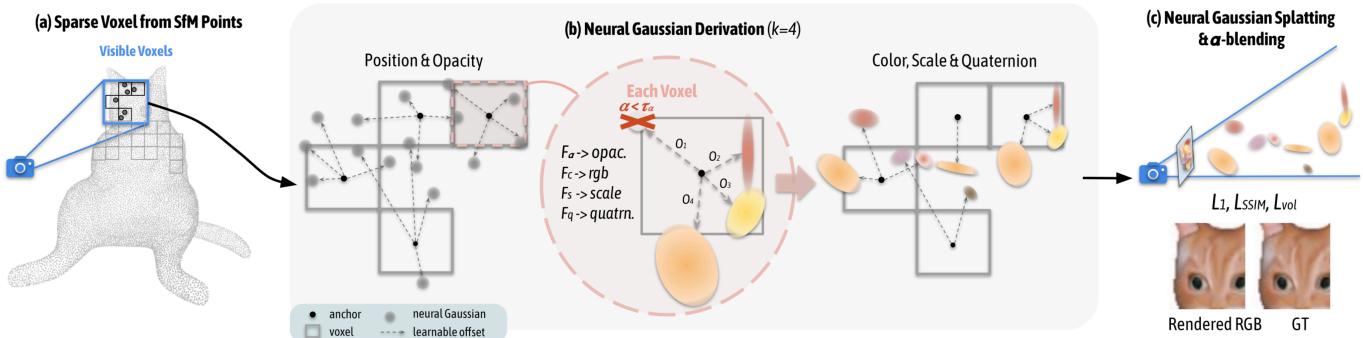


Figure 2. **Overview of Scaffold-GS.** (a) We start by forming a *sparse voxel grid* from SfM-derived points. An **anchor** associated with a learnable scale is placed at the center of each voxel, roughly sculpturing the scene occupancy. (b) Within a view frustum, k **neural Gaussians** are spawned from 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 Gaussians (*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}).

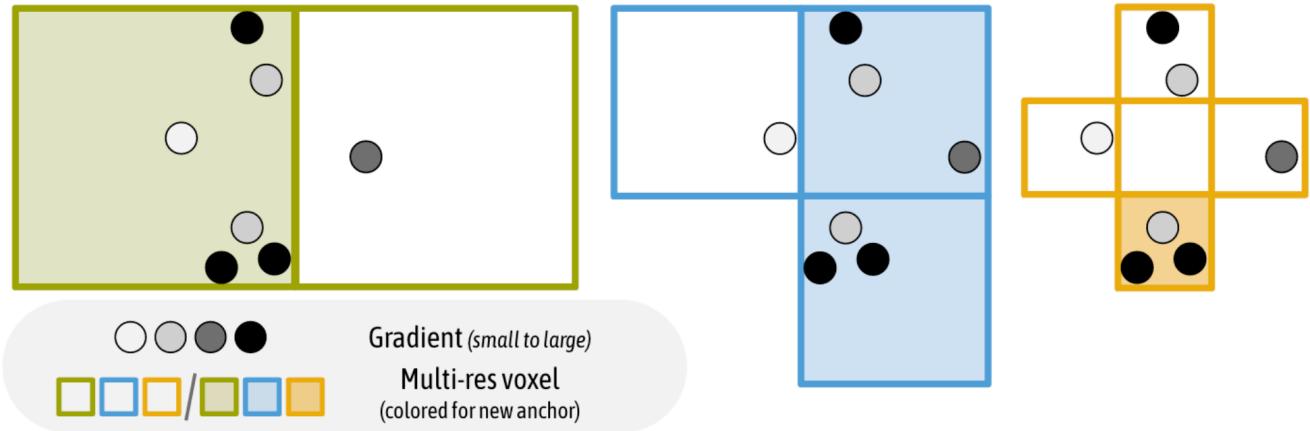


Figure 3. **Growing operation.** We develop an anchor growing policy guided by the gradients of the neural Gaussians. From left to right, we spatially quantize neural Gaussians into multi-resolution voxels ($m \in \{1, 2, 3\}$) of size $\{\epsilon_g^{(m)}\}$. New anchors are added to voxels with aggregated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

Method

本文提出了一种基于锚点的层次化3D高斯场景表示方法 **Scaffold-GS**, 通过锚点初始化、动态高斯推导及优化策略实现高效且鲁棒的场景渲染。以下为具体方法细节。

3.1 Preliminaries

我们基于 **3D Gaussian Splatting (3D-GS)** 方法构建改进模型。以下是 3D-GS 的核心概念：

1. 3D高斯分布表示

高斯分布用于表示场景中的每一个体元，其定义为：

$$G(x) = e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}, \quad (9)$$

其中：

- x : 任意3D场景位置；
- μ : 高斯分布中心；
- Σ : 协方差矩阵，用于描述高斯分布的形状和方向。

协方差矩阵通过旋转矩阵 R 和缩放矩阵 S 构造：

$$\Sigma = RSS^T R^T. \quad (10)$$

2. 高斯投影与渲染

将3D高斯投影到二维图像平面后，采用透明度混合实现渲染：

$$C(x') = \sum_{i \in N} c_i \sigma_i \prod_{j=1}^{i-1} (1 - \sigma_j), \quad \sigma_i = \alpha_i G'_i(x'), \quad (11)$$

其中：

- x' : 图像平面中的像素位置；
- c_i : 颜色；
- α_i : 透明度；
- $G'_i(x')$: 投影到二维的高斯分布。

3.2 Scaffold-GS

我们的 Scaffold-GS 方法通过以下模块改进了传统的 3D-GS：

3.2.1 Anchor Point Initialization

目标：从稀疏的 SfM 点云生成锚点，并构建稀疏网格以覆盖场景。

1. 初始化稀疏网格：

$$V = \left\{ \left\lfloor \frac{P}{\epsilon} \right\rfloor \right\} \cdot \epsilon, \quad (12)$$

其中：

- $P \in \mathbb{R}^{M \times 3}$: SfM 点云；
- ϵ : 体素网格大小；
- $V \in \mathbb{R}^{N \times 3}$: 去重后的体素中心，表示锚点位置。

2. 锚点参数化：

每个锚点包含以下属性：

- 局部上下文特征 $f_v \in \mathbb{R}^{32}$ ；
- 缩放因子 $l_v \in \mathbb{R}^3$ ；
- 可学习偏移量 $O_v \in \mathbb{R}^{k \times 3}$ 。

3. 多分辨率特征增强：

定义多分辨率特征库：

$$\hat{f}_v = w \cdot f_v + w_1 \cdot f_{v \downarrow 1} + w_2 \cdot f_{v \downarrow 2}, \quad (13)$$

其中权重通过 MLP 预测：

$$\{w, w_1, w_2\} = \text{Softmax}(F_w(\delta_{vc}, \vec{d}_{vc})), \quad (14)$$

- δ_{vc} : 相机与锚点的距离；
- \vec{d}_{vc} : 相机与锚点的视角方向。

3.2.2 Neural Gaussian Derivation

目标：基于锚点预测动态高斯分布的属性。

1. 高斯位置：

使用偏移量和缩放因子计算 k 个高斯的位置：

$$\{\mu_0, \dots, \mu_{k-1}\} = x_v + \{O_0, \dots, O_{k-1}\} \cdot l_v, \quad (15)$$

其中：

- x_v : 锚点位置；
- O_i : 第 i 个高斯的偏移量。

2. 属性预测：

通过 MLP 动态解码高斯属性：

- 透明度：

$$\{\alpha_0, \dots, \alpha_{k-1}\} = F_\alpha(\hat{f}_v, \delta_{vc}, \vec{d}_{vc}); \quad (16)$$

- 颜色：

$$\{c_0, \dots, c_{k-1}\} = \text{Sigmoid}(F_c(\hat{f}_v, \delta_{vc}, \vec{d}_{vc})); \quad (17)$$

- 缩放：

$$\{s_0, \dots, s_{k-1}\} = \text{Sigmoid}(F_s(\hat{f}_v, \delta_{vc}, \vec{d}_{vc})) \cdot s_v; \quad (18)$$

- 旋转：

$$\{q_0, \dots, q_{k-1}\} = \text{Normalize}(F_q(\hat{f}_v, \delta_{vc}, \vec{d}_{vc})). \quad (19)$$

3. 视锥内过滤：

激活视锥内的高斯分布，移除透明度低于阈值 τ_α 的冗余高斯。

3.3 Anchor Points Refinement

目标：通过生长和修剪策略优化锚点分布。

1. 生长策略：

在梯度显著区域添加新锚点：

- 构建多分辨率体素网格：

$$\epsilon_g^{(m)} = \frac{\epsilon_g}{4^{m-1}}, \quad \tau_g^{(m)} = \tau_g \cdot 2^{m-1}; \quad (20)$$

- 计算每个体素内的平均梯度 ∇_g , 若 $\nabla_g > \tau_g$, 则添加新锚点。

2. 修剪策略：

累积锚点对应高斯的透明度，若不满足阈值，则移除锚点。

3.4 Losses Design

目标：通过优化损失函数训练模型。

1. 总损失：

$$L = L_1 + \lambda_{\text{SSIM}} L_{\text{SSIM}} + \lambda_{\text{vol}} L_{\text{vol}}, \quad (21)$$

- L_1 : 渲染颜色的 L1 损失；
- L_{SSIM} : 结构相似性损失；
- L_{vol} : 体积正则化。

2. 体积正则化：

$$L_{\text{vol}} = \sum_{i=1}^{N_{\text{ng}}} \prod (s_i), \quad (22)$$

其中 N_{ng} 表示场景中的高斯数量, s_i 为高斯的缩放因子。

总结

本文方法通过锚点初始化、动态高斯推导及优化策略显著提高了渲染效率和质量，适用于多尺度复杂场景。

Implementation Details

本文对 Scaffold-GS 的实现细节进行详细说明，包括锚点特征增强、MLP 结构设计以及锚点优化策略的具体实现。

1. Anchor Feature Enhancement (锚点特征增强)

为了增强锚点的视角适应性，我们对其特征进行了多分辨率编码。

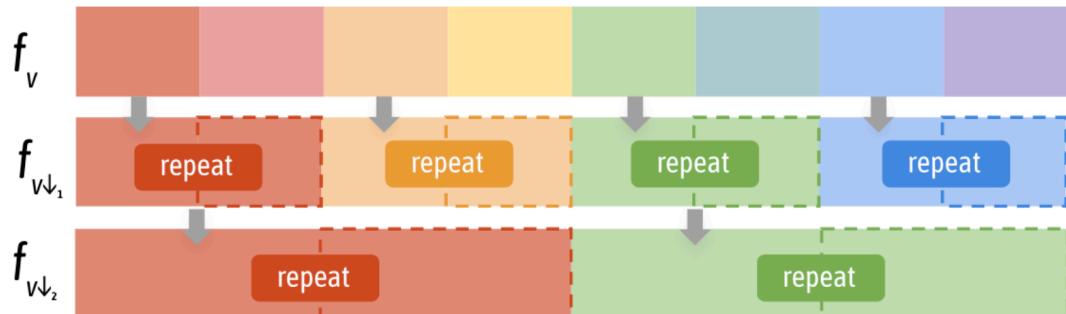


Figure 10. **Generation of Feature Bank.** We expand the anchor feature f into a set of *multi-resolution* features $\{f_v, f_{v\downarrow_1}, f_{v\downarrow_2}\}$ via slicing and repeating. This operation improves Scaffold-GS's ability to capture different scene granularity.

1.1 视角相关权重计算

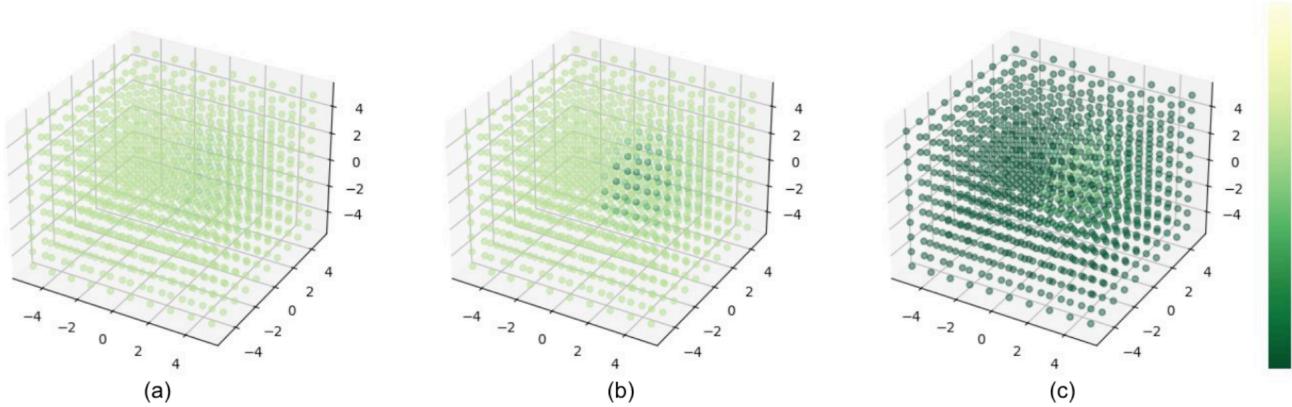


Figure 11. **View-based feature bank's weight distribution.** (a), (b) and (c) denote the predicted weights $\{w_2, w_1, w\}$ for $\{f_{v_{\downarrow 2}}, f_{v_{\downarrow 1}}, f_v\}$ from a group of uniformly distributed viewpoints. Light color denotes larger weights. For this anchor, finer features are more activated at center view positions. The patterns exhibit the ability to capture different scene granularities based on view direction and distance.

根据锚点与相机的位置关系，计算相对距离和方向：

$$\delta_{vc} = \|x_v - x_c\|_2, \quad \vec{d}_{vc} = \frac{x_v - x_c}{\|x_v - x_c\|_2}, \quad (23)$$

其中：

- x_v : 锚点位置；
- x_c : 相机位置。

通过一个小型 MLP F_w 预测多分辨率特征权重：

$$\{w, w_1, w_2\} = \text{Softmax}(F_w(\delta_{vc}, \vec{d}_{vc})). \quad (24)$$

1.2 特征库融合

定义锚点的多分辨率特征库 $\{f_v, f_{v \downarrow 1}, f_{v \downarrow 2}\}$, 融合公式为:

$$\hat{f}_v = w \cdot f_v + w_1 \cdot f_{v \downarrow 1} + w_2 \cdot f_{v \downarrow 2}. \quad (25)$$

实现方式:

- 特征降采样: 通过切片和重复操作生成多分辨率特征。
- 特征融合: 按视角权重将不同分辨率的特征加权求和。

2. MLP Structures (多层感知机结构)

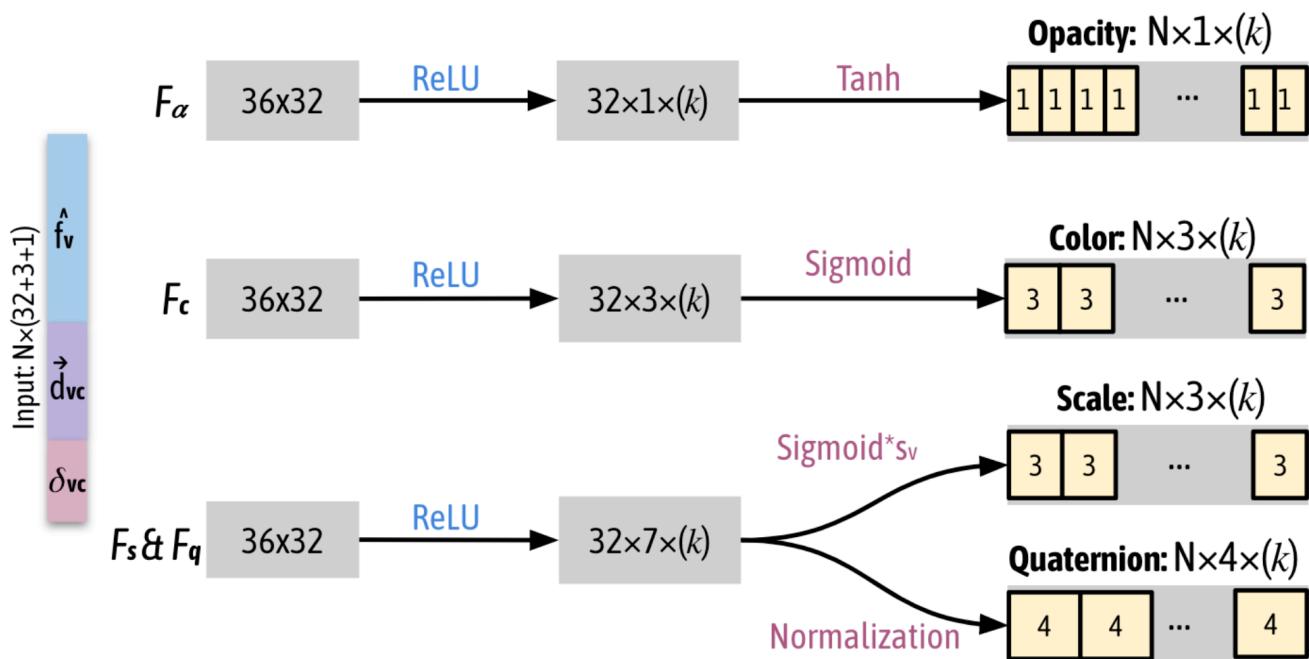


Figure 12. **MLP Structures.** For each anchor point, we use small MLPs (F_α, F_c, F_s, F_q) to predict attributes (opacity, color, scale and quaternion) of k neural Gaussians. The input to MLPs are anchor feature \hat{f}_v , relative viewing direction \vec{d}_{vc} and distance δ_{vc} between the camera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

2.1 属性解码器

高斯分布的属性（透明度、颜色、缩放、旋转）通过独立的 MLP 解码。每个 MLP 的结构为：

- 输入：锚点特征 \hat{f}_v 、相对距离 δ_{vc} 和方向 \vec{d}_{vc} ；
- 中间层：隐藏维度为32的全连接层，激活函数为 ReLU；
- 输出层：按属性类型激活。

各属性的输出如下：

- 透明度：

$$\{\alpha_0, \dots, \alpha_{k-1}\} = \text{Tanh}(F_\alpha(\hat{f}_v, \delta_{vc}, \vec{d}_{vc})), \quad (26)$$

使用 Tanh 激活以便值域覆盖 $[0, 1]$ 。

- 颜色：

$$\{c_0, \dots, c_{k-1}\} = \text{Sigmoid}(F_c(\hat{f}_v, \delta_{vc}, \vec{d}_{vc})), \quad (27)$$

- 缩放：

$$\{s_0, \dots, s_{k-1}\} = \text{Sigmoid}(F_s(\hat{f}_v, \delta_{vc}, \vec{d}_{vc})) \cdot s_v, \quad (28)$$

- 旋转：

$$\{q_0, \dots, q_{k-1}\} = \text{Normalize}(F_q(\hat{f}_v, \delta_{vc}, \vec{d}_{vc})). \quad (29)$$

3. Anchor Points Refinement (锚点优化)

3.1 Growing Operation (生长策略)

目标：在关键区域添加新锚点以增强场景覆盖。

1. 空间量化：

构建多分辨率体素网格：

$$\epsilon_g^{(m)} = \frac{\epsilon_g}{4^{m-1}}, \quad \tau_g^{(m)} = \tau_g \cdot 2^{m-1}, \quad (30)$$

其中：

- $\epsilon_g^{(m)}$: 第 m 层体素网格的大小；
- $\tau_g^{(m)}$: 梯度阈值。

2. 梯度计算：

统计每个体素内高斯分布的梯度均值 ∇_g , 若 $\nabla_g > \tau_g$, 则在体素中心添加新锚点。

3. 多层次优化：

使用多分辨率网格逐层添加锚点。

3.2 Pruning Operation (修剪策略)

目标：移除冗余锚点以减少存储和计算负担。

1. 透明度累积：

计算锚点生成高斯分布的透明度累积值，若小于阈值 τ_α , 则移除该锚点。

2. 实现策略：

- 透明度计算周期性执行；
- 修剪后重新评估锚点分布。

4. Voxel Size and Initialization (体素大小与初始化)

- 自动设置：

$$\epsilon = \text{median}(\|\text{nearest_neighbors}(P)\|), \quad (31)$$

根据点云密度自适应调整锚点分布。

- 手动设置：

对于纹理稀疏区域，可设置较大的 ϵ 值（如0.01）。

5. Training Details (训练细节)

1. 损失函数设计：

$$L = L_1 + \lambda_{\text{SSIM}} L_{\text{SSIM}} + \lambda_{\text{vol}} L_{\text{vol}}, \quad (32)$$

其中：

- $L_{\text{vol}} = \sum_{i=1}^{N_{\text{ng}}} \prod(s_i)$: 体积正则化，抑制高斯分布重叠。

2. 优化策略：

- 锚点优化（生长与修剪）每100次迭代执行一次；
- 透明度阈值 $\tau_\alpha = 0.5$ ，以平衡冗余与渲染质量。

总结

通过上述实现细节，Scaffold-GS 能够高效地生成紧凑、鲁棒的3D场景表示，并在渲染质量与速度上达到先进水平。

Dataset Method	Metrics	Mip-NeRF360			Tanks&Temples			Deep Blending		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	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

Table 2. Performance comparison. Rendering FPS and storage size are reported. Storage size reduction ratio is indicated by (\downarrow). 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	693	123	411	109	676
Ours	102	156 ($4.4 \times \downarrow$)	110	87 ($4.7 \times \downarrow$)	139	66 ($10.2 \times \downarrow$)

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 ($7.9 \times \downarrow$)	29.24	69 ($3.8 \times \downarrow$)	33.68	14 ($3.8 \times \downarrow$)

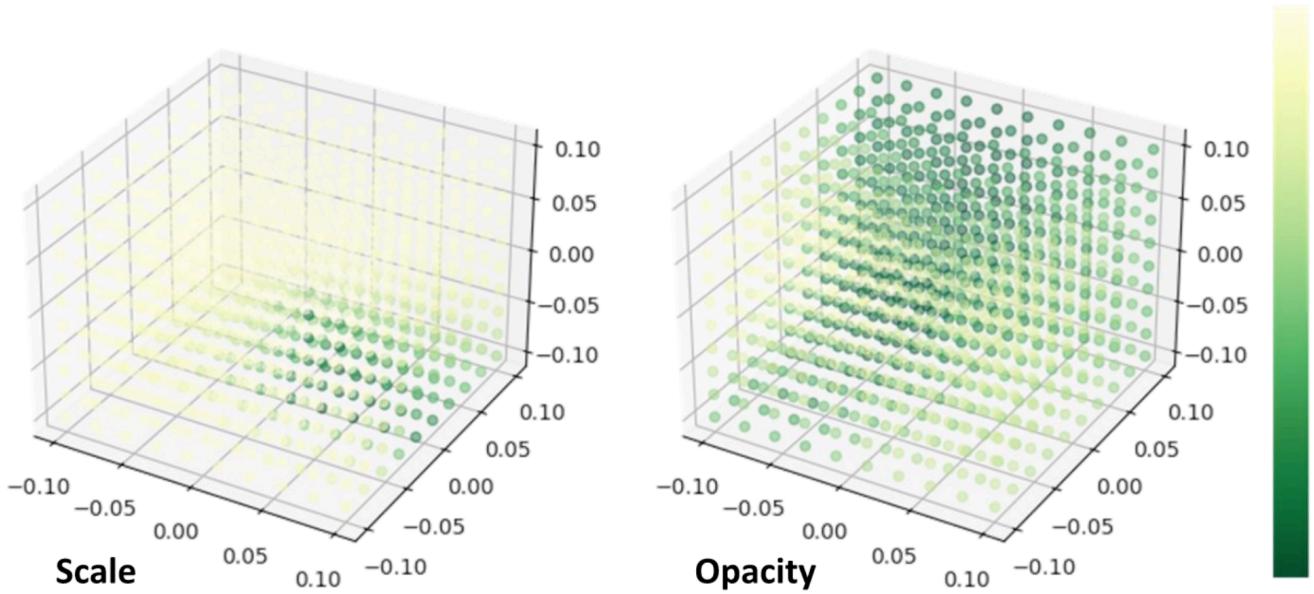


Figure 7. View-adaptive neural Gaussian attributes. We visualize the decoded attributes of a *single* neural Gaussian observed at different positions. Each point corresponds to a viewpoint in space. The color of the point denotes the intensity of attributes decoded for this view (left: $F_s \rightarrow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

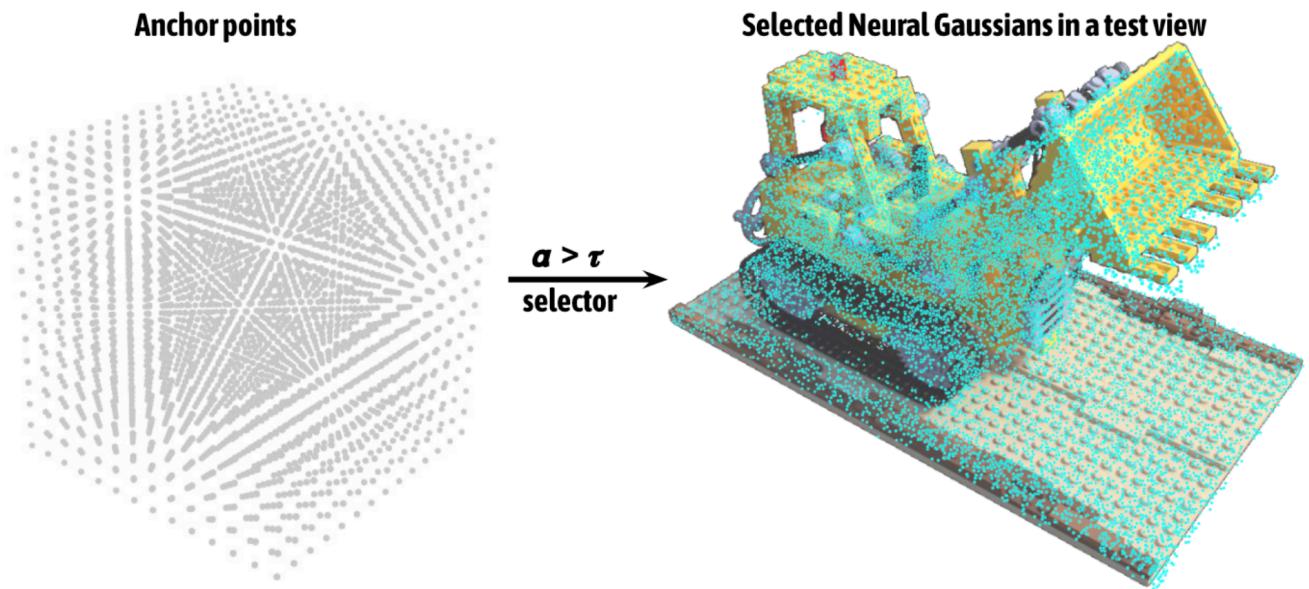


Figure 8. Geometry culling via selector. (Left) Anchor points from randomly initialized points; (Right) Activated neural Gaussians derived from each anchor under the current view. In synthetic Blender scenes, with all 3D Gaussians visible in the viewing frustum, our opacity filtering functions similar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

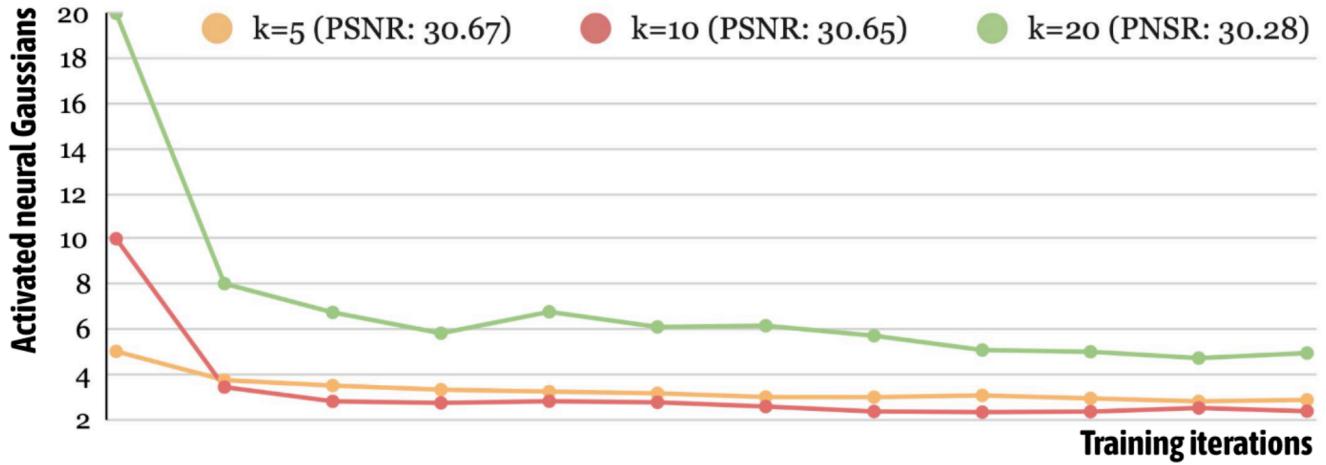


Figure 9. **Learning with different k values.** Despite varying initial k values under different hyper-parameter settings, they converge to activate a similar number of neural Gaussians with comparable rendering fidelity.

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

Experiment

本文通过在多个数据集上的综合实验，验证了 Scaffold-GS 的性能优势，包括渲染质量、存储效率及推理速度。

4.1 Experimental Setup

1. Dataset (数据集)

我们在多个公开数据集上测试了模型性能，涵盖以下场景：

- **Mip-NeRF360** (7个场景)：具有多尺度特征的室内外场景。
- **Tanks&Temples** (2个场景)：真实场景的复杂几何和光照。
- **DeepBlending** (2个场景)：包含反射和纹理稀疏区域。
- **Synthetic Blender** (8个场景)：合成数据，提供完整的360度视角。
- **BungeeNeRF** (6个场景)：多尺度户外观察场景。
- **VR-NeRF** (2个场景)：捕捉复杂室内环境的细节。

2. Metrics (评价指标)

采用以下指标评估模型性能：

- **PSNR (峰值信噪比)**：衡量渲染图像与真实图像的误差；
- **SSIM (结构相似性)**：衡量图像的结构相似度；
- **LPIPS**：感知距离，用于衡量视觉相似性；
- **存储大小 (MB)**：模型紧凑性；
- **推理速度 (FPS)**：实时渲染性能。

3. Baselines (对比方法)

选择以下方法作为对比基线：

- **3D-GS**: 现有最先进的3D高斯分布渲染方法；
- **Mip-NeRF360**: 高效的神经辐射场表示；
- **Instant-NGP**: 基于哈希编码的快速神经场方法；
- **Plenoxels**: 无神经网络的稀疏体素表示。

4. Training Configuration (训练配置)

- 训练轮次：30,000次迭代；
- 超参数设置：
 - $\lambda_{\text{SSIM}} = 0.2$, $\lambda_{\text{vol}} = 0.001$;
 - 锚点透明度阈值 $\tau_\alpha = 0.5$ 。

4.2 Results Analysis

1. Quantitative Comparison (定量对比)

Dataset	PSNR ↑	SSIM ↑	LPIPS ↓	Storage ↓	FPS ↑
Mip-NeRF360	28.84	0.848	0.220	156MB (4.4x ↓)	102
Tanks&Temples	23.96	0.853	0.177	87MB (4.7x ↓)	110
DeepBlending	30.21	0.906	0.254	66MB (10.2x ↓)	139

分析：

- Scaffold-GS 在 PSNR、SSIM 和 LPIPS 指标上优于或接近基线方法；
- 存储需求显著降低，比 3D-GS 缩小 4.4 至 10.2 倍；
- 在 1K 分辨率下，推理速度达到 100 FPS 以上。

2. Visual Results (可视化对比)

结果展示：

- **场景细节**: 相比 3D-GS, Scaffold-GS 在纹理稀疏区域（如墙面）表现更细致；
- **光照效果**: 在高动态光照场景中，Scaffold-GS 的渲染结果更加真实；
- **多尺度内容**: 在 BungeeNeRF 数据集中，Scaffold-GS 能够平滑地过渡到新视角，避免 3D-GS 出现的模糊和针状伪影。

4.3 Ablation Studies (消融实验)

1. Filtering Strategies (过滤策略的影响)

Scene	No Filters	Filter 1	Filter 2	Full Filtering
PSNR	30.4	30.3	30.6	30.62
FPS	84	118	109	150

分析：

- 完整的过滤策略在几乎不影响 PSNR 的情况下显著提升了推理速度；
- 过滤策略减少了冗余高斯的计算量，提高了模型效率。

2. Anchor Refinement (锚点优化的影响)

Scene	None	Pruning	Growing	Full Refinement
PSNR	28.45	29.12	30.54	30.62
Storage (MB)	24	23	71	63

分析：

- 修剪操作显著减少了存储需求；
- 生长操作显著提升了纹理稀疏区域的渲染质量。

4.4 Discussions and Limitations

1. 初始点云的重要性

- 锚点初始化依赖 SfM 点云质量，稀疏点云会导致锚点分布不均。
- 生长策略可以部分弥补这一问题，但在极度稀疏的场景中仍存在不足。

2. 多尺度场景的适应性

- Scaffold-GS 的动态属性预测在处理多尺度场景时表现尤为出色。
- 锚点的结构化分布为模型扩展到更大规模场景提供了可能

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

Conclusion

本文提出了 **Scaffold-GS**, 一种基于锚点的层次化3D高斯场景表示方法, 用于高效且鲁棒的视角自适应渲染。以下是本研究的主要贡献和总结:

1. 研究贡献

1. 锚点驱动的高斯分布建模：

Scaffold-GS 利用场景的几何结构，从稀疏点云中初始化锚点，并通过锚点生成局部的神经高斯分布。这种方法在保留场景全局结构的同时，允许高斯分布根据视角和距离进行动态调整。

2. 动态预测机制：

我们开发了基于 MLP 的视角自适应属性预测模块，通过对锚点特征的动态解码，高效生成视角相关的高斯属性（如透明度、颜色、缩放和旋转）。

3. 锚点优化策略：

Scaffold-GS 提出了基于梯度和透明度的锚点生长与修剪策略，这种优化策略提高了场景覆盖率，减少了冗余锚点，从而降低了存储和计算开销。

2. 实验验证

通过在多个公开数据集上的实验，我们发现：

- Scaffold-GS 在渲染质量（PSNR、SSIM、LPIPS）、存储效率和推理速度上均显著优于现有的 3D-GS 方法；
 - 在复杂场景中，Scaffold-GS 表现出更好的鲁棒性，尤其是在纹理稀疏区域和光照变化场景中；
 - 动态属性预测和锚点优化策略进一步增强了模型对多尺度内容和新视角的适应能力。
-

3. 局限性与未来工作

1. 锚点初始化的依赖性：

Scaffold-GS 的初始锚点分布依赖于 SfM 点云质量，对于极稀疏的点云可能表现不佳。未来可以探索利用深度估计或其他场景重建技术改进锚点初始化。

2. 复杂场景的处理：

虽然 Scaffold-GS 在多尺度和复杂场景中表现良好，但在非常大的场景中可能需要进一步优化锚点的组织和选择机制。

3. 应用扩展：

Scaffold-GS 的锚点特征具有一定语义信息，这表明其有潜力应用于大规模场景建模、内容操控和几何解释等任务。

4. 总结

本文通过引入结构化锚点和视角自适应高斯分布，显著提高了3D场景表示的紧凑性和渲染效率。实验表明，Scaffold-GS 能够在高效渲染的同时，保持或超越现有方法的质量，为多场景应用提供了强大的技术支持。我们相信，随着场景重建和锚点优化技术的进步，Scaffold-GS 的性能和适用范围将进一步提升。