arXiv:2512.17528v1 [cs.MM] 19 Dec 2025

Voxel-GS：采用游程编码的量化 Scaffold 高斯泼溅压缩

傅春阳[1],刘祥瑞[1],王世琪[1] 和李竹[2]

[1]香港城市大学
中国香港特别行政区
chunyang.fu@my.cityu.edu.hk
xiangrliu3-c@my.cityu.edu.hk

[2]密苏里大学堪萨斯城分校
美国堪萨斯城
shiqwang@cityu.edu.hk
zhu.li@ieee.org

摘要

大量的高斯泼溅格式点云需要高效的压缩方法。本文提出了 Voxel-GS，这是一种简单却极为有效的框架。该框架摒弃了以往工作中复杂的神经熵模型，转而仅使用轻量级码率代理和游程编码来实现具有竞争力的性能。具体而言，我们采用可微量化对 Scaffold-GS 的高斯属性进行离散化。随后，设计了一种基于拉普拉斯分布的码率代理，以施加熵约束，引导生成高保真且紧凑的重建结果。最后，利用八叉树和游程编码对这种整数型高斯点云进行无损压缩。实验验证了所提出的码率代理能够准确估计游程编码的比特率，使 Voxel-GS 得以消除冗余，并优化为更紧凑的表示。因此，与现有方法相比，我们的方法在实现显著更快的编码速度的同时，达到了极高的压缩比。代码可在 https://github.com/zb12138/VoxelGS 获取。

引言

3D 高斯泼溅（3DGS） [1]在三维建模领域展现出了非凡的能力。然而，其压缩仍需进一步研究，这已引起工业界和学术界的广泛关注。在大多数现有的压缩工作中, [2, 3, 4], 基于学习的熵模型（例如，因子化熵瓶颈 [5]）是高斯泼溅重建网络的一部分。这种对重建与压缩的联合优化直接学习多视图图像并生成比特流，同时将高斯点云作为副产品。鉴于此，MPEG 在最近召开的第152次会议上提出了高斯泼溅编码需求草案 [6] ，其中考虑了 GS压缩方案潜在标准化的两种范式:

- A-3DGS：一种“替代型”3DGS 表示，可基于真实图像/视频进行训练。该方案与当前大多数 3DGS压缩方法 [2, 3, 4]一致。它们通常将 GS网络与基于学习的熵模型结合进行联合学习，以一组多视图图像作为输入，并直接生成比特流。

该研究部分得到研究资助局一般研究基金11200323、NSFC/RGC联合研究计划项目 N CityU198/24、-香港创新及科技基金 GHP/044/21SZ 以及 PRP/036/24FX 的支持。

- I-3DGS：“INRIA” [1] 格式编码，涉及训练后的 GS 数据或“其他”基于点云的表示。由于 3D GS 点云与标准点云采用相同格式，因此扩展现有点云编码器（例如 G-PCC）以支持 3D GS 压缩或许是合理的。

尽管 A-3DGS 压缩方法能够实现较高的率失真性能，但主要面临以下问题。1）需要保存深度熵模型的参数，并为每个场景传输神经网络；然而，当 GS 场景相对较小时，分配给神经网络的比特率预算有限。2）由于重建与压缩是联合优化的，重建内容与熵模型高度相关，因此难以基于现有重建结果复用或单独优化编码器；这是整个系统中最耗时的部分。在 I-3DGS 压缩中，前置的重建网络可以指定为朴素 3DGS [1], 但这会引发一些问题。1）模型输出为浮点数据，为确保渲染保真度，需要使用大量比特进行存储，这不利于压缩。2）重建过程中没有熵约束，导致输出点云冗余且不够紧凑。

为了解决这些问题，本文探讨了一种新的前置 3DGS 网络，用于为 I-3DGS 生成 GS 点云。其目标是在重建过程中获得显式、紧凑的 GS 点云，并使用下游编码器对其进行压缩。本文提出的用于 I-3DGS 的 3DGS 网络基于最流行的方法之一 Scaffold-GS [7], ，且仅进行了非常小的改动。此外，我们假设使用游程编码作为下游编码器。游程编码是一种基础且应用广泛的熵编码方法，可使生成的点云兼容更广泛的其他下游编解码器。我们相信，本文提出的方案有助于推动高斯点云压缩的标准化。

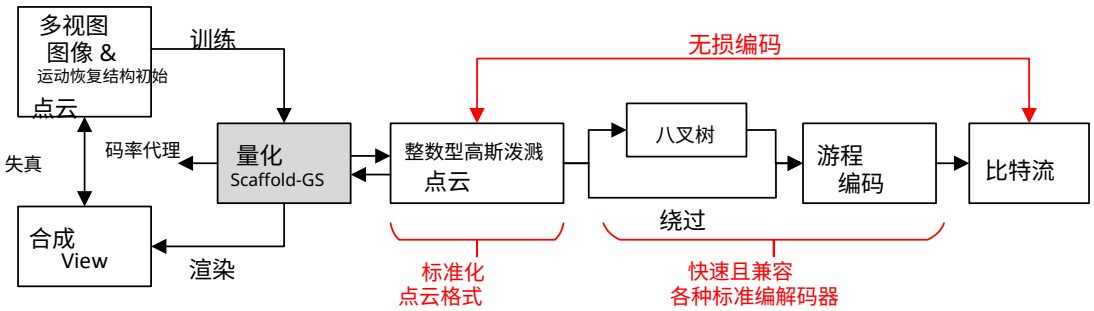
相关工作

3D 高斯泼溅中的分布本质上是稀疏且无序的。由于高斯基元之间存在冗余，可以通过掩码、梯度阈值、视角相关度量及其他重要性评估机制对其进行剪枝。此外，高斯基元的特征（如球谐函数（SH）系数、不透明度和尺度）可以通过量化、码本和熵约束进行调整，从而降低信息的不确定性。此外，Compressed3D [8] 和 Compact3DGS [9], 等早期探索性工作采用掩码、码本和量化方法对 GS 参数进行剪枝，以实现紧凑表示。Scaffold-GS [7]通过多层网络利用锚点生成高斯体，从而实现更紧凑的 GS 表示，在减少参数的同时提升保真度。LightGaussian [10]识别出场景重建中全局重要性最低的高斯体，并通过剪枝与恢复流程减少冗余，同时保持视觉质量。尽管这些紧凑表示方法与本文讨论的第一个目标（即生成紧凑表示）有些相似，但它们并未采用

显式熵约束，而这种约束在我们的方法中显著提升了 GS 表示的紧凑性。

后续方法 CompGS [4], 、ContextGS [11], 、HAC [2], 和 HAC++ [3] 逐渐转向联合重建与压缩，在 GS 网络中引入可学习的熵模型，并改进上下文感知设计，以实现更好的率失真性能。然而，这些方案主要面临两个问题：首先，需要存储熵模型参数，在小型场景中这可能造成显著开销；其次，灵活性不足，因为压缩器无法独立于重建过程使用或优化。此外，FCGS [12] 利用预训练的高斯点云离线学习熵模型，但其压缩率并不高。主要原因可能是预训练的高斯点云不够紧凑，而且压缩阶段引入的量化噪声会降低渲染质量。本文提出的方案将有助于缓解这一问题，具体而言，通过拉普拉斯熵约束生成紧凑表示，并将输出的高斯点云限制为整数，从而避免后续压缩器引入的噪声影响。HGSC [13] 和 HyBridGS [14] 是近期探讨标准 I-3DGS 压缩方法的工作；然而，它们采用 RAHT 作为点云压缩方案，这是一种有损压缩模式，会引入噪声，可能降低渲染质量。

方法



Fig图 1：所提出的高斯泼溅压缩方法概述 .

概述 Voxel-GS 的概述如图 1 所示。与大多数现有的端到端高斯泼溅压缩方法不同，Voxel-GS 将重建与压缩过程分离。经过 Voxel-GS 训练后，可获得紧凑的整数型高斯点云，并可使用八叉树和游程编码或其他潜在的编解码器进行压缩。这种结构具有以下优势：1）将重建与压缩分离，便于开展进一步研究，并改进离线高斯点云编解码器。2）对整数型特征进行无损压缩，确保压缩过程不会引入新的失真，从而避免渲染质量下降。3）输出的高斯点云可编辑，且压缩过程非常快速，因此无需经历漫长的重建过程，即可快速进行内容分发。

4）为 I-3DGS 压缩的标准化研究提供了一种方法。

体素化的 3D-GS 表示 Scaffold-GS 提供了一种方法，通过多层感知机使用每个高斯锚点生成 k 高斯基元，从而实现紧凑表示。在 Voxel-GS 中，我们采用了相同的框架，但对 Scaffold-GS 中锚点的位置和特征进行量化，以生成整数型高斯点云，使其兼容传统点云编解码器。Scaffold-GS 中锚点的位置 P 经过量化，并移除了重复锚点，以减少冗余和不规则性，这恰好满足点云编解码器（即 G-PCC）的要求。假设锚点位置的体素分辨率为 Q_P，我们修改了 Scaffold-GS 中的锚点位置函数，使其

$$P_Q = \text{STE_ROUND}(P * Q_P)/Q_P, \tag{1}$$

其中，STE ROUND 是一种采用直通估计器的舍入操作 [15]„，能够在训练过程中反向传播梯度。对于 Scaffold-GS 表示所需的特征，我们也进行了类似的修改。具体而言，维度为 $3 * k$ 的`offsets` O 使用 Q_O 进行量化，维度为 m 的 `anchor features` A使用 Q_A 进行量化，维度为 6 的 `scaling factors` S 使用 Q_S 进行量化。请注意， `rotation` 和 `opacity` 无需作为 Scaffold-GS 的保存信息，因为它们是由 `anchorfeatures` 通过小型多层感知机推导得出的。所有必需的特征都经过 STE ROUND 函数处理。在重建结束时，这些特征的量化值，

$$F_Q = \text{STE_ROUND}(F * Q_F), F \in \{P, O, A, S\} \tag{2}$$

以整数形式保存，因此生成一个具有 $3 * k + m + 6$维属性、坐标为 P_Q 的高斯点云。

影响高斯点云大小的因素包括锚点数量（即点云中的点数）、属性维度以及属性的取值范围。减少锚点数量还可以加快重建过程中的训练速度。因此，我们在 Voxel-GS 中的自适应控制之前引入锚点剪枝：当所有由 k 生成的高斯基元都不可见时，就会剪除该锚点。体素分辨率 Q_P 会影响锚点数量，而特征的量化参数则直接控制属性的取值范围。 m 和 k 会影响属性维度；然而，由于锚点剪枝，它们也会显著改变锚点数量。这一复杂因素超出了本研究的范围。因此， Q_P、 Q_O、 Q_A、 Q_S将作为超参数在实验中进行研究，而 m 和 k 则按照 HAC++[3]确定。

码率代理 由于传统点云编解码器不可微分，我们提出一种码率代理，用于估算比特率并约束特征，从而在重建过程中进行率失真优化。游程编码（RLC） [16]广泛用于传统点云编解码器的熵编码器中，是一种高效的方法，尤其适用于存在大量连续相同数据的情况。我们

提出使用拉普拉斯分布来估算其概率模型，

$$q(x) = \int_{x-0.5}^{x+0.5} \mathcal{L}_{\mu,\sigma}(x)\, dx, \tag{3}$$

其中，均值 μ 和标准差 σ 是从所有待编码的 x 中得出的。游程编码的实际比特率近似正比于交叉熵 $R \approx \alpha H(p, q)$，其中 p 是真实的量化特征分布F_Q，而 α 表示游程编码相较于基于假定的拉普拉斯分布的代理方法所取得的性能提升比例。码率代理的损失项为

$$\ell_R = \sum_{F \in \{O,A,S\}} \mathbb{E}_{F_Q}(-\log q(F_Q)). \tag{4}$$

与引入复杂的深度熵模型相比，码率代理无需神经网络参数，因此模型大小远小于端到端压缩方法。这在小规模场景压缩中尤其有利。请注意，我们没有对 P_Q施加任何约束，它将通过八叉树编码。其比特率通常与体素分辨率成正比，并在训练过程中保持稳定。

重建与压缩 我们采用与 Scaffold-GS 相同的数据集配置。每个场景训练共 30,000 步，并在 1,500 至 15,000 步期间进行锚点剪枝和自适应控制。为帮助 Voxel-GS 生成稳定的表示，在 15,000 步之前禁用式（1）中的量化和码率代理。15,000 步之后，锚点数量保持不变，并启用量化和码率代理。因此，最终的损失函数为

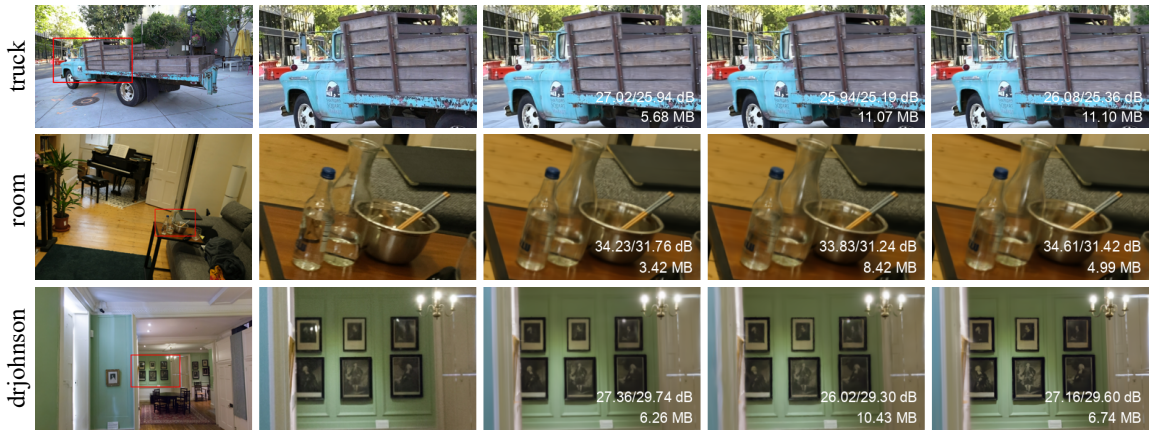
$$\ell = \lambda_1 \ell_{L1} + \lambda_2 (1 - \ell_{\text{SSIM}}) + \lambda_3 \ell_R. \tag{5}$$

重建过程完成后，量化后的锚点位置 P_Q 由 G-PCC 的八叉树分支（即 t mc13v23）进行无损压缩；量化后的偏移量 O_Q、锚点特征 A_Q和缩放因子 S_Q 则在按莫顿序重建的几何结构下，直接通过 RLC（旁路模式）进行无损压缩。此外，还会保存朴素 Scaffold-GS 的微型多层感知机。

实验结果

数据集与评估指标我们遵循 HAC++ [3] ，在多个数据集上进行评估，包括 Synthetic-NeRF [17], 、Mip-NeRF360 [18], 、Tank&Temples [19], 和 DeepBlending [20]。我们采用峰值信噪比、结构相似性和学习感知图像块相似度来评估渲染质量，并以比特流的大小评估压缩效率。

模型参数与实现 锚点特征维度设为 $m = 50$ 和 $k = 10$。体素分辨率 Q_P 在 Synthetic-NeRF 数据集上设为 1024，在其他三个数据集上设为 200。所有数据的量化尺度 Q_O，Q_A，Q_S 分别设为 1，1，8。损失函数中的权重设为 $\lambda_1 = 0.2$、$\lambda_2 = 0.8$和 $\lambda_3 = 0.0001$[1]。其他超参数与 Scaffold-GS 保持一致。



场景真实值 HAC++ CompGS Voxel-GS（我们的方法）

图2： 定性比较。该视图/所有视图的峰值信噪比及大小已标注 d.

表1：定 量比较结果：锚点、A-3DGS 和 I-3DGS 压缩方法

数据集	Synthetic-NeRF [17]				Mip-NeRF360 [18]				Tank&Temples [19]				DeepBlending [20]			
方法	峰值信噪比↑	结构相似性↑	学习感知图像块相似度↓	大小↓	峰值信噪比↑	结构相似性↑	学习感知图像块相似度↓	大小↓	峰值信噪比↑	结构相似性↑	学习感知图像块相似度↓	大小↓	峰值信噪比↑	结构相似性↑	学习感知图像块相似度↓	大小↓
3DGS[1]	33.80	0.970	0.031	68.46	27.46	0.812	0.222	750.9	23.69	0.844	0.178	431.0	29.42	0.899	0.247	663.9
ScaffoldGS[7]	33.41	0.966	0.035	19.36	27.50	0.806	0.252	253.9	23.96	0.853	0.177	86.50	30.21	0.906	0.254	66.00
HAC[2]	33.24	0.967	0.037	1.18	27.53	0.807	0.238	15.26	24.04	0.846	0.187	8.10	29.98	0.902	0.269	4.35
CompGS[4]	-	-	-	-	26.37	0.778	0.276	8.83	23.11	0.815	0.236	5.89	29.30	0.895	0.293	6.03
ContextGS[11]	32.79	0.965	0.040	1.01	27.62	0.808	0.237	12.68	24.20	0.852	0.184	7.05	30.11	0.907	0.265	3.45
HAC++[3]	33.03	0.966	0.039	0.88	27.60	0.803	0.253	8.34	24.22	0.849	0.190	5.18	30.16	0.907	0.266	2.91
LightGS[10]	32.73	0.965	0.037	7.84	27.00	0.799	0.249	44.54	22.83	0.822	0.242	22.43	27.01	0.872	0.308	33.94
Comp3D[8]	32.94	0.967	0.033	3.68	26.98	0.801	0.238	28.80	23.32	0.832	0.194	17.28	29.38	0.898	0.253	25.30
FCGS[12]	-	-	-	-	27.05	0.798	0.237	34.64	23.48	0.832	0.193	17.89	-	-	-	-
HyBridGS[14]	-	-	-	-	25.97	0.766	0.272	21.73	23.12	0.820	0.236	11.10	29.05	0.885	0.250	16.35
Ours	32.75	0.963	0.043	0.77	26.87	0.784	0.271	12.51	23.57	0.829	0.219	7.77	30.19	0.909	0.262	6.72

大小以 MB 为单位，PSNR 以 dB 为单位。

八叉树和游程编码通过 C++ 后端实现。实验在配备英伟达 4090 GPU 和英特尔 i9-13900K CPU 的单台设备上进行。

结果视觉对比见图2，Voxel-GS 展现出卓越的视觉质量。多个数据集上的定量对比见表1。在 I-3DGS 压缩方法中，我们的方法表现最佳。与最新的 HyBridGS 相比，我们的方法平均提升了 0.85 dB，同时将比特率降低了 45%。在三个大规模数据集上的平均结果显示，Voxel-GS（26.88 dB，17.92 MB）与 CompGS（26.26 dB，17.47 MB）表现相当。此外，我们的方法（28.34 dB，6.94 MB）在四个数据集上的平均表现与 HAC（28.70 dB，7.22 MB）非常接近。值得注意的是，在较小的场景中（即 Synthetic-NeRF），我们的方法优于 ContextGS，因为与深度熵模型相比，无参数代理在比特流预算有限时更具优势。尽管 Voxel-GS 与联合学习

[1]为平衡 ℓ_R (100∼1000) 与 $\ell_{L1}, \ell_{\text{SSIM}}$ (0∼1) 的取值范围差异。

表 2：Tank&Temples 数据集上的平均复杂度比较

方法	训练（分钟）	编码器（秒）	解码器（秒）	每秒帧数	GPU 内存	高斯泼溅数量
Scaffold-GS [7]	13.32	-	-	112.1	7 GB	517839
CompGS [4]	46.60	6.27	4.46	99.8	13 GB	244370
HAC++ [3]	24.67	8.13	13.33	105.6	10 GB	286650
Ours	14.01	1.28	1.47	111.8	6 GB	312238

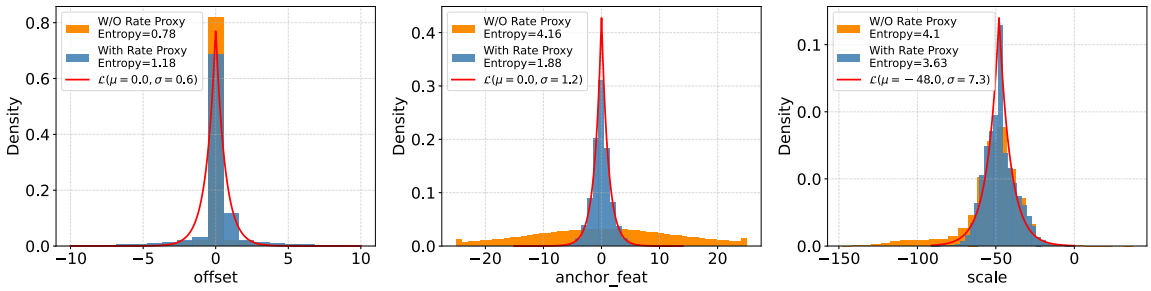
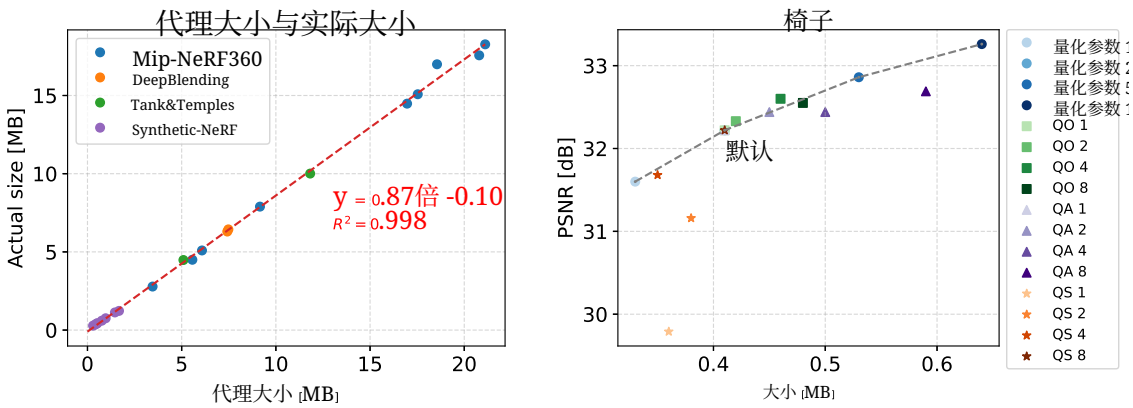


图 3：来自“椅子”场景的 `offset`、 `anchor feature` 和 `scaling factor` 示例直方图，以及估计的拉普拉斯分布（红色曲线）。

方法 HAC++ 在某些数据集上仍存在细微差距，但我们认为 I-3DGS 压缩方法的优势，尤其是在灵活性、模型大小和速度方面，更为显著。

复杂度 表 2 展示了在 Tank&Temples 数据集上进行的复杂度比较。与其他方法相比，Voxel-GS 在重建过程中不依赖复杂的熵模型，因此能够保留略多的高斯基元，同时训练速度仍比联合学习方法快约 2∼3倍，且 GPU内存占用几乎减半。此外，与深度熵模型相比，传统编码器的复杂度更低，因此相较于 HAC++，我们节省了 85% 的编码和解码时间。在压缩和解压缩大量场景时，这种时间节省尤为显著，使我们的方法更具实用性。

码率代理的有效性为验证所提出的码率代理，我们将其移除并进行了消融研究。图3展示了 Synthetic-NeRF 中“椅子”场景的结果，其他数据集上也观察到了类似结果。可以看出，加入码率代理后，特征 `anchor feature` 和`scaling factor` 明显更加集中于拉普拉斯分布附近，熵降低了 33%。对于 `offset`，其分布似乎意外地更加分散。一种可能的解释是，在原始 Scaffold-GS 中，许多偏移量未被激活，因此保持为零。加入约束后，为了维持渲染质量，更多派生高斯被激活。总体而言，代理约束将比特流大小从原始量化 Scaffold-GS 的 1.59 MB 降至 0.91 MB，同时基本保持渲染质量不变（34.15 dB 降至 34.14 dB）。这清楚地表明，拉普拉斯码率代理能够有效生成保真度高且更加紧凑的高斯点云表示。



(a) (b)

图4：（a）码率代理估计与实际游程编码的比特率。（b）使用不同量化参数时“椅子”上的率失真性能。默认值为 $Q_P = 256$、$Q_O = 1$、$Q_A = 1$、和 $Q_S = 8$。

表3：大场景和小场景的比特分配比较

场景	方法	P	O	A	S	多层感知机	其他	大小(MB)	峰值信噪比
训练	HAC++	3.08%	25.97%	45.67%	17.29%	4.34%	3.64%	7.61	22.61
	Ours	5.71%	22.02%	56.50%	14.62%	1.14%	0.00%	4.44	21.79
榕树	HAC++	6.86%	10.23%	32.28%	12.94%	32.68%	5.03%	0.64	34.53
	Ours	5.62%	13.21%	58.18%	13.35%	9.65%	0.00%	0.53	34.60

P：锚点位置，O：偏移量，A：锚点特征，S：缩放因子。

码率代理准确度按数据集划分的码率代理估计与游程编码（RLC）中特征（即，O，A，S）的码率大小如图4（a）所示。拟合直线的斜率表明，$\alpha = 0.87$ 在公式（3）之后。这意味着，与拉普拉斯分布相比，RLC 可节省约 13% 的比特率，说明 RLC 效率更高。实际编码比特率与估计比特率高度吻合，相关系数为 0.998。这表明，所提出的码率代理能够准确估计游程编码的码率，为重建过程中的率失真优化提供可靠依据。

参数选择 我们通过将一个量化参数从默认值开始逐一调整，分析其影响。默认值设为 Q_P =256，$Q_O = 1$，$Q_A = 1$，和 $Q_S = 8$。“椅子”场景上的率失真结果如图2 (b)所示。其中，调整 Q_P 的效果最显著且稳定（以黑色虚线表示），因此可将其用作多种码率编码和码率控制的常规参数。相比之下，调整 Q_S 会导致峰值信噪比大幅下降；因此，我们建议将其设为大于 8。调整 Q_O和 Q_A 的效果并不显著，通常建议在确定 Q_P后将它们作为辅助微调参数，推荐值为 1。

比特分配 与 HAC++ 的比特分配对比如表3所示。表中显示，除多层感知机（网络）外，与 HAC++ 相比，我们方法中大多数组件的比特占比呈现出一致的趋势。在“训练”等较大场景中，网络占整体大小的比例较小，因此 HAC++ 相较于我们的方法更具优势。相反，在“榕树”等较小场景中，受限的比特率预算会使网络占比显著上升，这可能成为联合学习方法的瓶颈。相比之下，Voxel-GS不包含深度熵模型，且多层感知机的大小仅为 0.07 MB，因此更适合压缩小型场景。

结论

本文提出了 Voxel-GS，用于生成和压缩高斯泼溅格式的点云。该方法引入了基于拉普拉斯分布的码率代理，以引导 Scaffold-GS 生成紧凑的整数型高斯点云。此外，采用八叉树和游程编码对量化后的 GS 点云进行无损压缩。Voxel-GS 的性能可与最先进的方法媲美，并且训练和编码速度更快。同时，它也符合标准化要求。我们相信，所提出的方法有助于推动高斯点云压缩的标准化。

附录

表4：Voxel-GS 在所有场景上的详细结果。灰色行表示平均值。

场景	峰值信噪比↑	结构相似性↑	学习感知图像块相似度↓	大小↓	场景	峰值信噪比↑	结构相似性↑	学习感知图像块相似度↓	大小↓
椅子	34.14	0.981	0.018	0.91	bicycle	24.42	0.699	0.317	19.38
drums	26.20	0.950	0.046	1.09	bonsai	31.12	0.934	0.207	7.91
榕树	34.60	0.984	0.015	0.53	柜台	27.86	0.884	0.245	3.38
热狗	36.89	0.981	0.030	0.51	花	20.56	0.520	0.429	14.63
lego	34.91	0.978	0.023	1.47	花园	26.74	0.825	0.175	20.63
材料	29.45	0.951	0.060	0.51	厨房	30.30	0.916	0.146	6.12
mic	35.16	0.988	0.012	0.39	room	31.42	0.919	0.219	4.99
ship	30.63	0.893	0.143	0.76	树桩	26.42	0.745	0.301	17.47
					树丘	22.95	0.617	0.397	18.07
Sync。	32.75	0.963	0.043	0.77	Mip360。	26.87	0.784	0.271	12.51
训练	21.79	0.789	0.259	4.44	drjohnson	29.60	0.906	0.261	6.74
卡车	25.36	0.868	0.180	11.10	游戏室	30.78	0.911	0.263	6.71
T2T.	23.57	0.829	0.219	7.77	DB.	30.19	0.909	0.262	6.72

参考文献

[1] B. Kerbl、G. Kopanas、T. Leimk¨uhler 和 G. Drettakis，《用于实时辐射场渲染的 3D 高斯泼溅》。*ACM Transactions on Graphics*，第 42 卷，第 4 期，页码 139–1，2023 年。

[2] Y. Chen、Q. Wu、W. Lin、M. Harandi 和 J. Cai，《HAC：用于 3D 高斯泼溅压缩的哈希网格辅助上下文》，载于欧洲计算机视觉会议论文集，2024 年，页码 422–438。

[3] ——, 《HAC++：迈向 3D 高斯泼溅的 100 倍压缩》，*IEEE Transactions on Pattern Analysis and Machine Intelligence*，第 47 卷，第 11 期，页码 10 210–10 226，2025 年。[4] X. Liu、X. Wu、P. Zhang、S. Wang、Z. Li 和 S. Kwong，《CompGS：通过压缩高斯泼溅实现高效的 3D 场景表示》，载于*ACM 多媒体国际会议论文集*，2024 年，页码 2936–2944。[5] J. Ball´e、V. Laparra 和 E. P. Simoncelli，《端到端优化图像压缩》，载于国际学习表征会议论文集，2017 年。[6] MPEG 技术要求，《高斯泼溅编码要求草案》，ISO/IEC JTC 1/SC 29/WG 2 MPEG，技术报告 N473，2025 年 7 月。[7] T. Lu、M. Yu、L. Xu、Y. Xiangli、L. Wang、D. Lin 和 B. Dai，《Scaffold-GS：用于视图自适应渲染的结构化 3D 高斯》，载于*IEEE/CVF 计算机视觉与模式识别会议论文集*，2024 年，页码 20 654–20 664。[8] S. Niedermayr、J. Stumpfegger 和 R. Westermann，《用于加速新视图合成的压缩 3D 高斯泼溅》，载于*IEEE/CVF 计算机视觉与模式识别会议论文集*，2024 年，页码 10 349–10 358。[9] J. C. Lee、D. Rho、X. Sun、J. H. Ko 和 E. Park，《用于辐射场的紧凑 3D 高斯表示》，载于*IEEE/CVF 计算机视觉与模式识别会议论文集*，2024 年，页码 21 719–21 728。[10] Z. Fan、K. Wang、K. Wen、Z. Zhu、D. Xu 和 Z. Wang，《LightGaussian：无界 3D 高斯压缩，压缩率达 15 倍并提升 200+每秒帧数》，载于神经信息处理系统进展论文集，第 37 卷，2024 年，页码 140 138–140 158。[11] Y. Wang、Z. Li、L. Guo、W. Yang、A. Kot 和 B. Wen，《ContextGS：采用锚点级上下文模型的紧凑 3D 高斯泼溅》，神经信息处理系统进展，第 37 卷，页码 51 532–51 551，2024 年。[12] Y. Chen、Q. Wu、M. Li、W. Lin、M. Harandi 和 J. Cai，《快速前馈式 3D 高斯泼溅压缩》，*arXiv* 预印本 *arXiv:2410.08017*，2024 年。[13] H. Huang、W. Huang、Q. Yang、Y. Xu 和 Z. Li，《一种用于 3D 高斯泼溅压缩的分层压缩技术》，载于*IEEE 国际声学、语音与信号处理会议论文集*，2025 年，页码 1–5。[14] Q. Yang、L. Yang、G. Van der Auwera 和 Z. Li，《HybridGS：利用双通道稀疏表示和点云编码器实现高效高斯泼溅数据压缩》，载于国际机器学习会议论文集，2025 年。[15] Y. Bengio、N. L´eonard 和 A. Courville，《针对条件计算，通过随机神经元估计或传播梯度》，*arXiv* 预印本 *arXiv:1308.3432*，2013 年。[16] MPEG 3DG，《关于通过移除字典和游程编码修改属性系数熵编码的报告》，ISO/IEC JTC1/SC29/WG11 MPEG，技术报告 m54265，2020 年 7 月。[17] B. Mildenhall、P. P. Srinivasan、M. Tancik、J. T. Barron、R. Ramamoorthi 和 R. Ng，《NeRF：将场景表示为用于视图合成的神经辐射场》，*ACM 通讯*，第 65 卷，第 1 期，页码 99–106，2021 年。[18] J. T. Barron、B. Mildenhall、D. Verbin、P. P. Srinivasan 和 P. Hedman，《Mip-NeRF 360：无界抗锯齿神经辐射场》，载于*IEEE/CVF 计算机视觉与模式识别会议论文集*，2022 年，页码 5470–5479。[19] A. Knapitsch、J. Park、Q.-Y. Zhou 和 V. Koltun，《坦克与寺庙：大规模场景重建基准测试》，*ACM 图形学汇刊*，第 36 卷，第 4 期，页码 1–13，2017 年。[20] P. Hedman、J. Philip、T. Price、J.-M. Frahm、G. Drettakis 和 G. Brostow，《用于自由视点图像渲染的深度混合》，*ACM 图形学汇刊*，第 37 卷，第 6 期，页码 1–15，2018 年。