Improved RAHT-based Compression of 3D Gaussian Splats

Annalisa Gallina, Giuseppe Valenzise, Sara Baldoni, Federica Battisti

Improved RAHT-based Compression of 3D Gaussian Splats

Annalisa Gallina[†‡], Giuseppe Valenzise[‡], Sara Baldoni[†], Federica Battisti[†]
[†]*University of Padova, Italy*
[‡] *Université Paris-Saclay, CNRS, CentraleSupelec, Laboratoire des Signaux et Systèmes, France*

***Abstract*—3D Gaussian Splatting (3DGS) enables high-quality rendering of 3D scenes, at the price of significant storage and bandwidth costs. By interpreting 3DGS as point clouds with rich attributes, we propose using the Region-Adaptive Hierarchical Transform (RAHT) — a simple and well-established tool in point cloud compression standards — to compress 3DGS attributes. Previous attempts to apply RAHT to 3DGS have shown limited coding efficiency due to the redundant and unstructured nature of the data. We address this with a simple yet effective fine-tuning of the 3DGS that promotes sparsity in the transform domain, thereby reducing bitrate. We further introduce a systematic bit allocation strategy across transformed attributes. Despite its simplicity, our method achieves competitive performance while keeping compatibility with existing standard coding tools.**

***Index Terms*—3D Gaussian Splatting, Compression, Rendering, Fine tuning**

I. Introduction

3D Gaussian Splatting (3DGS) has recently emerged as a promising technique for novel view synthesis and real-time rendering of 3D scenes. By representing geometry and appearance through anisotropic Gaussian primitives, 3DGS enables high visual fidelity, rapid training, and efficient rasterization on standard GPUs. However, 3DGS representations are highly redundant and can require hundreds of megabytes to store a single scene, hindering their practical deployment.

To address this, recent research has focused on compression techniques aimed at reducing storage and bandwidth requirements [4, 1]. However, existing 3DGS compression approaches often rely on learned representations or neural entropy models that, while effective, tend to introduce considerable complexity in both the encoder and decoder. These methods frequently require scene-specific tuning or retraining, making them difficult to generalize and costly to deploy at scale. Moreover, their reliance on neural components complicates standardization. In contrast, classical point cloud compression techniques, such as those based on transform coding and quantization, have shown strong performance in standardization efforts thanks to their interpretability, modularity, and decoder simplicity.

For this reason, in this work, we present a novel compression scheme for 3DGS that draws inspiration from traditional *point cloud compression methods*, specifically exploiting the Region Adaptive Hierarchical Transform (RAHT) [19] and taking as a reference the MPEG GPCC (Geometry-based Point Cloud Compression) pipeline [16]. In simple terms, 3DGS can be viewed as a point cloud where each point carries not only its 3D position and color but also additional attributes such as scale, rotation, opacity, and Spherical Harmonic (SH) coefficients. However, to the best of our knowledge, very few methods have adapted existing coding algorithms for point cloud compression on Gaussian splat attributes: [24] exploits RAHT transform to compress only few attributes, while most of them are vector quantized. Instead, in [10] the same transform is used for encoding attributes of anchor points. The coding efficiency of those methods is limited by the unstructured and redundant nature of 3DGS, which reduces the effectiveness of transform coding. In this work, we show that RAHT-based compression can be significantly improved by: (i) fine-tuning the 3DGS data to *enhance sparsity* in the transform domain, and (ii) *allocating bits* across attributes based on their impact on the final rendering quality.

Specifically, our proposed framework voxelizes positions and applies a transform coding step to decorrelate Gaussian attributes. The output of the transform is then quantized using a channel-wise quantization step derived by a rate-distortion (RD) analysis of the impact of each attribute on rendering. To improve rate-distortion performance, we fine-tune the 3DGS parameters by minimizing a sparsity loss in the transform domain. Both proposed contributions — sparsity-promoting fine-tuning and per-channel bit allocation — are encoder-side operations that require no additional side information in the bitstream. They yield significant coding gains over naïve RAHT-based compression of 3DGS, as well as competitive performance with recent and more sophisticated approaches.

II. Related Work

As outlined in [1], 3DGS compression techniques can be broadly categorized into two main classes: *unstructured* and *structured* methods.

Unstructured methods operate directly on the numerical values of model parameters, such as Gaussian attributes or Gaussians themselves, without explicitly considering their spatial or structural relationships. These techniques typically begin with a trained model and aim to optimize its efficiency. Redundancy is commonly addressed by pruning Gaussians with limited contribution to the scene representation, either through importance-based scoring [10, 24, 9] or by learning

This work was partially supported by the Italian Ministry of University and Research (MUR) under the PRIN 2022 project SEQUOIA (code 2022KCKYA2).

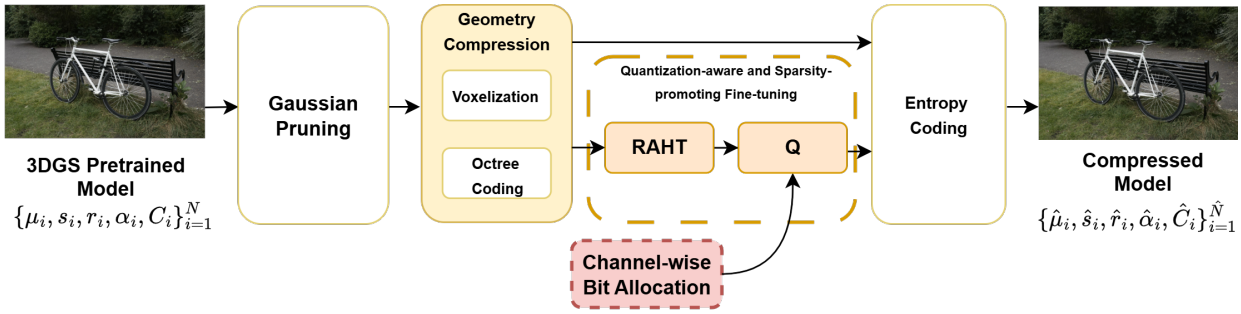

Fig. 1: Proposed compression pipeline.

dynamic masks [22, 3]. For attribute compression, a variety of techniques have been explored. Several methods employ codebooks to cluster similar attributes [17, 9], while others leverage entropy coding to model statistical distributions and reduce storage [21, 15].

In contrast, *structured* methods modify the training process to produce more compact representations. These approaches exploit the spatial and structural coherence of the scene to guide the placement and organization of Gaussians in a way that inherently reduces redundancy. A notable example is Scaffold-GS [13], which introduces anchor-based clustering to group nearby Gaussians with similar features. This strategy has inspired follow-up works which further improved compression effectiveness [8, 25, 23]. While effective, these methods require significant changes to the training pipeline and are often incompatible with existing implementations, making them less practical in scenarios where retraining is infeasible.

Transform-based methods can be regarded as a *hybrid* category: like unstructured approaches, they operate on pre-trained models, but they also exploit structural coherence thanks to the transform, similarly to structured techniques. Only a few works have explored this direction. Notably, MesonGS [24] applies the RAHT to compress a subset of Gaussian attributes, while the rest are vector quantized. In contrast, we systematically apply transform coding to all Gaussian attributes and introduces a channel-wise bit allocation scheme guided by signal features and rendering relevance.

III. Proposed Method

A. Preliminaries

3DGS is an explicit point-based 3D scene representation method: the scene is represented as a set of 3D anisotropic Gaussians, each defined by a mean $\mu \in \mathbb{R}^3$ and a covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$, that can be further factorized into a scale vector $s \in \mathbb{R}^3$ and a rotation vector $r \in \mathbb{R}^4$.

In addition to shape, each Gaussian is associated with an opacity value $\alpha \in \mathbb{R}$ and a color represented by SH coefficients $C \in \mathbb{R}^{16\times3}$.

Novel view synthesis is performed using a tile-based differentiable rasterizer. This rendering process projects 3D Gaussians onto the image plane to produce 2D splats. For each 2D tile in the rendered views, an ordering is established among the Gaussians that contribute to its rendering, and a point-based rendering operation is then applied. More details on the rendering process can be found in the original paper [11].

B. Overview

The proposed compression framework is summarized in Figure 1. The pipeline begins with the removal of redundant Gaussians produced during 3DGS generation [11]. Gaussian means are then voxelized and encoded using Octree coding. After this initial *geometry compression*, attributes of merged Gaussians are averaged, and their dimensionality is reduced by replacing the rotation quaternion ($q \in \mathbb{R}^4$) with its equivalent Euler angle representation ($e \in \mathbb{R}^3$), as proposed in [24]. In the *attribute compression* step, each attribute is processed independently: RAHT transform is applied, followed by block-wise scalar quantization. Each attribute channel is first partitioned into multiple blocks based on Morton ordering, to ensure spatially coherent grouping of Gaussians, and each block is quantized independently using uniform scalar quantizers. A distinct quantization parameter is chosen per channel based on its energy and significance, as detailed below. Prior to this step, a short *fine-tuning* is performd to balance the RD trade-off and improve compression efficiency. Finally, all data — geometry, attributes and metadata — undergo *entropy coding* using the *deflate* algorithm [18] to produce the final bitstream.

C. Preprocessing and Transform

Gaussian Pruning To mitigate the redundancy introduced by the baseline [11], we consider a well-established importance-based pruning strategy [9]. Gaussians are ranked by an importance score, and the least relevant ones are discarded. The importance score is computed as the product of two terms: a *view-dependent* score I_d, reflecting the Gaussian's relevance on rendering and a *view-independent* score I_i derived from the Gaussian's volume.

Geometry Compression The 3D positions of the Gaussians are voxelized using an Octree structure. The Octree depth d is selected to preserve most of the Gaussians and minimize rendering distortion. The geometry bitstream is then generated via Octree coding, using the MPEG GPCC codec [16].

RAHT Coding RAHT is a hierarchical wavelet-based transform originally designed for point cloud attribute compression [19]. It operates along the Octree hierarchy and applies

successive transforms across spatial axes to progressively decorrelate neighboring attribute values.

In our framework, we adopt the same transform used in point cloud compression and apply it independently to each attribute channel. Additionally, we introduce a pre-processing step on the Gaussian attributes, detailed below, which further enhances the sparsity of the transform coefficients.

D. Channel-wise Bit Allocation

To enable effective quantization and improve compression efficiency, a bit allocation scheme must account for the varying energy and rendering importance of each attribute. In this work, we adopt a scene-independent, channel-wise allocation strategy, which simplifies the encoder and avoids the overhead of transmitting large amounts of side information.

To formalize the problem, let b_i^* denote the optimal number of quantization bits for the i-th attribute. Our goal is to approximate it with a scene-independent value that performs well across different scenes. To this end, we conduct an extensive analysis of how the quantization of each attribute impacts final rendering quality. For every attribute, we plot the rendering distortion as a function of its per-channel bitrate:

$$D_{\text{render}} = f(R_i), \quad i \in [1, n_{\text{attr}}]. \tag{1}$$

Specifically, for each experiment we first apply the RAHT, and then code attribute i using a number of bits $b \in \{2, 4, 6, 8, 16, 32\}$. All the other attributes $j \neq i$ are instead coded at a fixed length of 8 bits. An example of the obtained RD curves is shown in Figure 2 for two attributes, rotation and second-order SH coefficients.

The observed RD curves reveal that different attributes exhibit distinct behaviors: they saturate at different bitrates — beyond which additional precision yields negligible improvements in rendering quality — and show varying slopes before saturation, reflecting their sensitivity to coarse quantization. For instance, in Figure 2, rotation (left) requires a higher bitrate to preserve quality, saturating around 8 bpp, while the second-order SH (right) exhibits a much flatter RD curve, with saturation near 4 bpp, indicating that it tolerates coarser quantization. These observations motivate the need for attribute-specific quantization, which we address by guiding our bit allocation scheme based on saturation points. This behavior is consistent across scenes, supporting the generalization of bit allocation decisions rather than requiring scene-specific tuning. Similar experiments were performed for all attributes and across a broader set of scenes, drawn from the three datasets used in the evaluation phase, leading to the design of the bit allocation scheme summarized in Table I.

E. Sparsity-promoting Fine-tuning

Before transform coding, we introduce a fine-tuning stage to improve energy compaction in the transform domain, and consequently, rate-distortion performance. During this phase, Gaussian positions remain fixed, while only appearance attributes are optimized. Densification is disabled, and quantization is included in the loop using a straight-through estimator [6] to approximate gradients through the non-differentiable

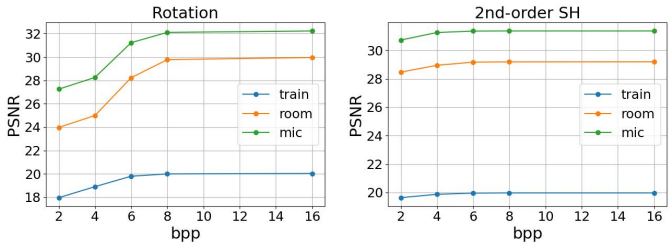

Fig. 2: RD Curves for Rotation and 2nd order SH coefficients.

TABLE I: Proposed channel-wise bit allocation.

Parameter (attribute channel)	b_i^*
α	8
$e_i,\ i \in [1,3]$	8
C_i, 0th order	8
$s_i,\ i \in [1,3]$	10
C_i, 1st order	4
C_i, 2nd order	4
C_i, 3rd order	2

quantization step. The distortion loss $\mathcal{L}_D$ in the original 3DGS pipeline [11] is defined as:

$$\mathcal{L}_D = (1 - \lambda_{SSIM})\mathcal{L}_1 + \lambda_{SSIM}\mathcal{L}_{SSIM}. \tag{2}$$

However, optimizing for visual quality alone often results in models where spatially close Gaussians exhibit highly diverse attributes, limiting the ability to exploit spatial redundancy in compression. To promote a more efficient representation, we introduce a rate proxy term in the form of a sparsity-promoting loss that penalizes large-magnitude transform coefficients, concentrating energy into fewer coefficients and thereby improving coding efficiency [20]. The sparsity loss $\mathcal{L}_S$ is given by:

$$\mathcal{L}_S = \frac{1}{n_{ch}} \sum_{i=1}^{n_{ch}} \sum_{j=1}^{n_{ac}} |x_{ij}|, \tag{3}$$

where n_{ch} is the number of attributes per Gaussian, n_{ac} is the number of AC coefficients after applying RAHT, and x_{ij} denotes the j-th transform coefficient of the i-th channel. This formulation corresponds to the average ℓ_1 norm per channel, which encourages sparsity by pushing many coefficients toward zero [7].

The total loss function is a linear combination of $\mathcal{L}_D$ and $\mathcal{L}_S$, with their relative importance controlled by the hyperparameter λ_S:

$$\mathcal{L} = (1 - \lambda_S)\mathcal{L}_D + \lambda_S\mathcal{L}_S. \tag{4}$$

A key advantage of this formulation is that it allows for a straightforward and effective control of the RD trade-off via λ_S tuning, thereby making the system much more flexible compared to most of existing approaches in the literature. Importantly, the computational overhead of this additional loss is negligible, as RAHT is already part of the training loop for quantization-aware fine-tuning.

TABLE II: Compression performance results on 3 datasets. PSNR is in dB while model size is in MB. Best, second best, and third best values for each column are denoted in red, yellow, and green, respectively.

	Mip-NeRF360				Tanks & Temples				NeRF-Synthetic			
	PSNR	SSIM	LPIPS	Size	PSNR	SSIM	LPIPS	Size	PSNR	SSIM	LPIPS	Size
3DGS [11]	27.21	0.815	0.214	734	23.14	0.841	0.183	411	31.07	0.959	0.051	61.39
MesonGS+FT [24]	28.61	0.856	0.206	27.62	23.32	0.837	0.193	16.99	32.92	0.968	0.033	3.66
LightGaussian [9]	27.28	0.805	0.243	42.0	23.11	0.817	0.231	22.0	32.73	0.965	0.037	7.8
CompGS-16K [17]	27.03	0.804	0.243	18.0	23.39	0.836	0.200	12.0	-	-	-	-
Trimming the Fat [2]	27.13	0.798	0.248	20.1	23.69	0.831	0.210	8.6	-	-	-	-
RDO-Gaussian [22]	27.05	0.802	0.239	23.5	23.34	0.835	0.195	12.0	32.97	0.966	0.035	1.84
HAC-lowrate [8]	27.53	0.807	0.238	15.3	24.04	0.846	0.187	8.1	33.24	0.967	0.037	1.18
HAC-highrate [8]	27.77	0.811	0.230	21.9	24.40	0.853	0.177	11.2	33.71	0.968	0.034	1.86
ContextGS-lowrate [23]	27.62	0.808	0.237	12.7	24.20	0.852	0.184	7.1	-	-	-	-
ContextGS-highrate [23]	27.75	0.811	0.231	18.4	24.29	0.855	0.176	11.8	-	-	-	-
Ours 5K	28.44	0.876	0.213	15.61	23.35	0.841	0.197	9.91	32.60	0.965	0.039	1.92
Ours 2K	28.54	0.867	0.206	18.04	23.41	0.841	0.198	10.71	32.53	0.955	0.039	2.20

IV. Experiments

A. Experimental Setting

Implementation Details We train the model based on 3DGS [11] using the standard configuration. The output after $30k$ steps is fed into our system. System parameters are set to values commonly used in the literature: pruning percentage, octree depth and number of quantization blocks are adopted from [24]. Results are reported after $2k$ and $5k$ fine-tuning iterations, with λ_{SSIM} fixed at 0.2 and λ_S set to $5 \cdot 10^{-7}$.

Datasets We conduct experiments on three commonly-adopted datasets for NVS: Mip-NeRF360 [5], Tanks and Temples [12], and NeRF-Synthetic [14]. For Mip-NeRF360, we considered the main part of the dataset consisting of 7 scenes, thus excluding the extra scenes *Flower* and *Treehill*. For Tanks and Temples we included only two scenes (*Train* and *Truck*), following a common practice in the related literature. As to NeRF-Synthetic, we considered all the provided scenes.

Baselines and Evaluation metrics We evaluate our method against several baselines, including the original 3DGS [11], unstructured methods [9, 2, 17, 22], and MesonGS [24], which employs RAHT-based transform coding. We also compare with representative structured methods [8, 23]. Performance is measured in terms of model size (MB) and rendering quality using PSNR, SSIM, and LPIPS on the test set. We follow the experimental setup of [11], using the same train-test split. Results for other methods are taken from their respective papers, except for 3DGS on NeRF-Synthetic dataset, which was recomputed using the 3DGS original code.

B. 3DGS Compression Results

The quantitative results in Table II clearly demonstrate the effectiveness of our proposed method. In terms of rate, our system outperforms all considered unstructured approaches [17, 9, 24, 2, 22], achieving compression gains of $47\times$, $41\times$, and $32\times$ over the baseline on Mip-NeRF360, Tanks and Temples, and NeRF Synthetic datasets, respectively, while maintaining or even improving the visual quality. Moreover, our method remains competitive with the structured approaches [8, 23] on real-world scenes, surpassing their high-rate variants on Mip-NeRF360. On the Tanks and Temples dataset, our PSNR is slightly lower — similar to other unstructured methods — due to baseline artifacts that are challenging to mitigate. Conversely, structured methods show slightly better performance on synthetic objects. However, the baseline 3DGS models we employed achieve PSNR scores on average 2.25 dB lower than those reported in the original paper [11], which penalizes the results of our method. This is likely due to minor modifications introduced to the original implementation to properly handle background-free synthetic images, as the unmodified code produced background artifacts.

TABLE III: Our bit allocation scheme vs. uniform allocation

	Room		Truck		Mic	
	PSNR	Size	PSNR	Size	PSNR	Size
8 bits	31.7	26.15	25.62	44.87	35.54	7.44
6 bits	24.88	20.71	18.08	28.22	20.30	6.21
Ours	31.15	9.09	25.61	10.85	35.13	2.57

C. Ablation Studies

We conducted ablation studies to assess the impact of our main contributions: the fine-tuning strategy and the bit allocation scheme. While pruning yields a significant size reduction, it is a standard practice and is therefore not analyzed here. We selected three scenes — *Room* (Mip-NeRF360), *Truck* (Tanks and Temples), and *Mic* (NeRF Synthetic) — using $2k$ fine-tuning iterations when in place. For each scene, we report the model size (in MB) and PSNR (in dB).

Bit Allocation Table III compares the proposed bit allocation scheme against two uniform quantization baselines using 8 and 6 bits per attribute, respectively. Compared to the 8-bit uniform allocation, our method achieves an average compression gain of $3.3\times$ with a negligible quality loss of only 0.32 dB, an imperceptible difference to the human eye. In contrast, when compared to the 6-bit uniform allocation, our scheme outperforms both quality and rate, demonstrating that some attributes — especially scale parameters — require finer

TABLE IV: Comparison of fine-tuning methods: (i) no fine-tuning, (ii) fine-tuning without sparsity loss, and proposed.

	Room		Truck		Mic	
	PSNR	Size	PSNR	Size	PSNR	Size
(i)	30.01	11.04	22.14	16.35	33.03	3.08
(ii)	31.26	11.03	25.55	16.36	35.12	3.07
Ours	31.15	9.09	25.61	10.85	35.13	2.57

precision due to their high impact on rendering quality. These results confirm that different attributes vary in importance for rendering, and approximating them uniformly is suboptimal. The proposed scheme effectively addresses this by adapting precision according to attribute significance.

Fine-tuning Table IV compares the results of our proposed fine-tuning scheme with two alternatives: (i) no fine-tuning and (ii) fine-tuning without the sparsity-promoting term.

Although skipping fine-tuning (i) still results in a substantial size reduction compared to the baseline 3DGS — mainly due to pruning and RAHT with quantization — it comes at the cost of notable quality degradation, since the original parameters are not optimized for spatial coherence. Fine-tuning without using sparsity loss (ii) can improve the rendering quality due to additional iteration and quantization-aware training, but does not reduce the bitstream size, limiting its compression benefits.

Overall, these results suggest that multiple parameter sets can represent a 3D scene equivalently well. Our fine-tuning strategy guides the model toward solutions that achieve higher compression gains without sacrificing quality.

V. Conclusion

By treating 3DGS as point clouds with rich attributes, we significantly improve RAHT-based compression by introducing an effective bit allocation strategy for each attribute channel – analogous to quantization matrices in JPEG – and by fine-tuning the 3DGS to promote sparsity in the transform domain. Together, these two contributions enable state-of-the-art performance among unstructured 3DGS coding methods, and in many cases, achieve results competitive with more recent approaches. Notably, our method could be easily integrated into and extend standard point cloud codecs.

In future works, we will perform a deeper analysis and optimization of the time and space complexity of the proposed pipeline. We will use a higher-quality, state-of-the-art pre-trained 3DGS model to better assess the potential of our compression framework. Additionally, we also plan to explore more sophisticated, scene-adaptive bit allocation strategies and improved loss formulations to further drive the representation toward a compressible form without retraining from scratch.

References

[1] M. S. Ali et al. *Compression in 3D Gaussian Splatting: A Survey of Methods, Trends, and Future Directions*. 2025. arXiv: 2502.19457.

[2] M. S. Ali et al. *Trimming the Fat: Efficient Compression of 3D Gaussian Splats through Pruning*. 2024. arXiv: 2406.18214.

[3] A. Armagan et al. *Trick-GS: A Balanced Bag of Tricks for Efficient Gaussian Splatting*. 2025. arXiv: 2501.14534.

[4] M. T. Bagdasarian et al. *3DGS.zip: A survey on 3D Gaussian Splatting Compression Methods*. 2025. arXiv: 2407.09510.

[5] J. T. Barron et al. *Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields*. 2022. arXiv: 2111.12077.

[6] Y. Bengio et al. *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*. 2013. arXiv: 1308.3432.

[7] E. J. Candes et al. "Enhancing sparsity by reweighted l 1 minimization". In: *Journal of Fourier analysis and applications* 14.5 (2008), pp. 877–905.

[8] Y. Chen et al. "HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression". In: *European Conference on Computer Vision*. 2024.

[9] Z. Fan et al. *LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS*. 2023. arXiv: 2311.17245.

[10] H. Huang et al. *A Hierarchical Compression Technique for 3D Gaussian Splatting Compression*. 2025. arXiv: 2411.06976.

[11] B. Kerbl et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". In: *ACM Transactions on Graphics* 42.4 (July 2023).

[12] A. Knapitsch et al. "Tanks and temples: benchmarking large-scale scene reconstruction". In: *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073599.

[13] T. Lu et al. "Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 20654–20664.

[14] B. Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *ECCV*. 2020.

[15] W. Morgenstern et al. "Compact 3D Scene Representation via Self-Organizing Gaussian Grids". In: *Computer Vision – ECCV 2024*. Cham: Springer Nature Switzerland, 2025, pp. 18–34. DOI: 10.1007/978-3-031-73013-9_2.

[16] MPEG 3DG. *G-PCC codec description v4*. Technical Report N18673. Geometry-based Point Cloud Compression (G-PCC). ISO/IEC JTC1/SC29/WG11, Sept. 2019.

[17] K. Navaneet et al. "CompGS: Smaller and Faster Gaussian Splatting with Vector Quantization". In: *ECCV* (2024).

[18] S. Oswal et al. "Deflate compression algorithm". In: *International Journal of Engineering Research and General Science* 4.1 (2016), pp. 430–436.

[19] R. L. de Queiroz et al. "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform". In: *IEEE Transactions on Image Processing* 25.8 (2016), pp. 3947–3956. DOI: 10.1109/TIP.2016.2575005.

[20] K. Sayood. *Introduction to Data Compression*. 5th. Morgan Kaufmann, 2017.

[21] K. G. Sharath Girish et al. *EAGLES: Efficient Accelerated 3D Gaussians with Lightweight EncodingS*. 2024. arXiv: 2312.04564.

[22] H. Wang et al. "End-to-End Rate-Distortion Optimized 3D Gaussian Representation". In: *European Conference on Computer Vision*. 2024.

[23] Y. Wang et al. "ContextGS : Compact 3D Gaussian Splatting with Anchor Level Context Model". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024.

[24] S. Xie et al. "MesonGS: Post-training Compression of 3D Gaussians via Efficient Attribute Transformation". In: *European Conference on Computer Vision*. Springer. 2024.

[25] Y.-T. Zhan et al. *CAT-3DGS: A Context-Adaptive Triplane Approach to Rate-Distortion-Optimized 3DGS Compression*. 2025. arXiv: 2503.00357.