

Gaussian splatting model compression

Jure Dežman^{1,*}, Miha Mitič^{1,*}, Uroš Šmajdek¹,
Matija Marolt¹, Ciril Bohak¹, Žiga Lesar^{1,2}

¹Univerza v Ljubljani, Fakulteta za računalništvo in informatiko

²Univerza v Novi Gorici

* Shared first authorship

E-mail: {jm1540, mm2704}@student.uni-lj.si, ziga.lesar@ung.si,
{uros.smajdek, matija.marolt, ciril.bohak}@fri.uni-lj.si

Abstract

3D Gaussian Splatting (3DGS) achieves real-time novel view synthesis but its uncompressed models use hundreds of megabytes, limiting deployment on memory-constrained devices. We implement and evaluate a compression pipeline combining opacity pruning, Hilbert-curve reordering with delta encoding, vector quantization of spherical harmonic coefficients, and 16-bit scalar quantization across six standard benchmark scenes. The best configuration achieves 11–12× compression rate while retaining PSNR above 39 dB and SSIM above 0.988, demonstrating that a simple pipeline can meaningfully reduce 3DGS storage with minimal perceptual loss. Code is available at GitHub¹.

1 Introduction

3DGS [1] represents a scene as a set of anisotropic 3D Gaussians, each defined by a position, rotation, scale, opacity, and spherical harmonic (SH) coefficients for view-dependent color. Unlike neural radiance fields (NeRF) [2], which encode scenes using neural network weights and require volumetric ray sampling at render time, 3DGS is an explicit point-based representation that maps well to GPU rasterization pipelines, enabling real-time framerates.

The cost of this representation is memory. Each primitive carries 59 float32 attributes: a position, a rotation quaternion, a log-scale, an opacity logit, and 48 SH coefficients encoding view-dependent color (3 terms approximating diffuse color and 45 higher-order terms). A typical scene requires millions of such primitives, stored uncompressed in .ply files, which pushes models from roughly 150 MB to over 1300 MB. This makes streaming over limited connections or deploying on memory-constrained devices such as mobile phones and VR/AR headsets impractical.

In this paper we investigate a Gaussian representation compression pipeline as was previously done for volumes [3, 4], mesh geometry [5] or 4D simulation vascular data [6]. The pipeline is composed of four stages: opacity pruning, Hilbert-curve reordering with delta encoding, vector quantization of SH coefficients, and scalar quantization. We evaluate all meaningful combinations across six standard benchmark scenes and analyze the trade-offs

between compression ratio, reconstruction quality (PSNR and SSIM), and decompression speed.

2 Related work

The storage demands of 3DGS [1] have driven intensive research into model reduction. Recent surveys categorize these efforts into *compression* (efficiently encoding attributes) and *compaction* (reducing primitive counts) [7], or divide them into unstructured pipelines and structured frameworks that exploit spatial topology [8]. For foundational contexts on compressing unordered 3D point data outside radiance fields, see Cao *et al.* [9].

Compaction techniques eliminate redundant or low-contribution primitives to counter the over-densification caused by standard 3DGS heuristics [1]. Early solutions discarded low-opacity Gaussians post-optimization. More recent pipelines inject structural constraints directly into training; for instance, Fan *et al.* [10] proposed LightGaussian, which uses global significance maps to prune insignificant primitives alongside SH distillation. While conceptually aligned with our opacity pruning step, these methods operate during optimization, whereas our pipeline acts entirely as a post-processing framework.

Quantization targeting the 59 float32 attributes per Gaussian significantly yields smaller file sizes. However, low-bit uniform scalar quantization introduces visible geometric and scaling artifacts. To mitigate this, vector quantization is widely used for high-dimensional spherical harmonic (SH) coefficients [7, 8]. Niedermayr *et al.* [11] combined sensitivity-aware vector clustering with quantization-aware training to compact directional features. Our pipeline adopts a similar strategy but isolates the diffuse and higher-order terms to create specialized, dual-codebook representations.

Exploiting spatial locality helps compress disordered point data. Sorting Gaussians along space-filling curves, such as Hilbert or Morton curves, projects 3D spatial neighborhood structures into a 1D sequence [9, 8]. This highly correlates neighboring attributes, making them well-suited for delta encoding paired with traditional entropy tools (*e.g.*, zlib). Alternatively, structured frameworks redesign the primitives themselves; anchor-based designs like Scaffold-GS [12] store sparse spatial anchors and predict local Gaussian attributes dynamically via neural networks. While our work centers on post-training

¹<https://github.com/jimocool2/GaussianSplattingCompression>

unstructured pipelines, these orthogonal developments delineate the performance upper-bounds of explicit attribute reduction.

3 Method

We implement a compression pipeline with four optional stages applied sequentially. They are opacity pruning, Hilbert curve sorting, vector quantization of SH, and scalar quantization. Optionally the final output can be compressed with zlib² regardless of which stages are active.

3.1 Opacity pruning

This technique aims to eliminate Gaussians with low opacity values. These contribute minimally to the scene. Gaussians with $\sigma(o_i) \leq \tau$ are removed, where σ is the sigmoid function and τ is the opacity threshold. This is a lossy compression step. The default value chosen is $\tau = 0.08$, higher thresholds yield smaller file sizes, but may introduce visible holes or artifacts in the scene. The remaining Gaussians are passed to the subsequent compression pipeline.

3.2 Hilbert curve sorting

This step works by reordering Gaussians based on their spatial positions. Coordinates are normalized and mapped onto a discrete grid, from which a scalar Hilbert distance is computed per Gaussian and used as the sort key. This maps multidimensional data into a linear sequence while preserving locality. With this ordering, consecutive Gaussians are spatially close, so their attribute differences are small. Before compression, we apply delta encoding, storing the difference between consecutive attribute values rather than the values themselves. When Hilbert curve sorting is active, delta-encoded data is particularly amenable to lossless compression. Therefore, zlib is applied automatically to the final output.

3.3 Vector quantization

In this step we encode spherical harmonic attributes by treating them as two separate vector quantization problems rather than one. We split the 48 SH coefficients into 3 diffuse color channels and 45 higher-order SH coefficients. We use separate codebooks to also allow different quantization budgets for each. This process works by clustering high dimensional data, where each cluster is approximated by a centroid. For that we use the K-means method where we iteratively assign data points to clusters, producing a codebook of 4096 centroids. When computing centroids we stop iteration if the maximum centroid movement is less than 1×10^{-6} . The result is stored as integer indices into the codebook rather than the full float32 vectors. For example for color coefficients a single uint16 index (2 byte) replaces 3 float32 values (12 bytes). Both the codebook and indices are stored in the compressed file, allowing the decoder to reconstruct the SH values by looking up each index in the codebook.

²<https://zlib.net/>

3.4 Scalar quantization

Here we convert float32 values to lower byte count representations (uint16 or uint8), reducing the accuracy but also decreasing file size requirements. With lower accuracy we introduce quantization error which produces visible artifacts in the rendered scene. To minimize quantization error we used min and max values from the original and then scaled them to the min and max ranges of the new value type, so that we keep as much accuracy as possible. The formula is:

$$q = \text{round} \left(\frac{x - x_{\min}}{x_{\max} - x_{\min}} \cdot (2^b - 1) \right), \quad (1)$$

where b is the bit depth of the new type. We reconstruct the float32 using this formula:

$$x \approx \frac{q}{2^b - 1} \cdot (x_{\max} - x_{\min}) + x_{\min}. \quad (2)$$

4 Results

For the evaluation we rendered each scene three times from three fixed camera angles (0° , 120° , and 240°) with each compression configuration active. We evaluated six scenes: train and truck from Tanks and Temples [13], bicycle and garden from Mip-NeRF 360 [14], and Dr Johnson and playroom from Deep Blending [15]. Rendering was performed using the OpenGL-based rasterizer running at a resolution of 1280×720 pixels. Figure 1 shows example renders from the *bicycle* scene contrasting the reference, the best configuration, and the worst.

We used four metrics that capture both render quality and representation efficiency: peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), compression ratio, and model size (in megabytes):

- **PSNR** measures pixel-level fidelity between two images; higher values indicate less distortion.
- **SSIM** measures perceptual similarity in terms of luminance, contrast, and structure; higher values indicate closer resemblance to the reference.
- **Compression ratio** is the original PLY size divided by the compressed size; higher is more compact.

All experiments were conducted on a machine equipped with an AMD Ryzen 7 9800X3D processor, AMD Radeon RX 7800 XT, and 32 GB of RAM.

We evaluated 21 compression configurations in total, combining subsets of the four pipeline stages. Rather than reporting all configurations, we rank each one by a composite score and show the top 10 in Table 1, and all 21 combinations in Table 2. For each configuration we compute a per-group average of PSNR, SSIM, and compressed size in MB across the two scenes in each benchmark group (T&T, Mip-NeRF 360, Deep Blending), then rank all 21 configurations independently on each of those three grand-mean values. The final score is computed as:

$$\text{Score} = \frac{\text{rank}(\text{PSNR})}{4} + \frac{\text{rank}(\text{SSIM})}{4} + \frac{\text{rank}(\text{MB})}{2},$$

where rank 1 denotes the best value for that metric and rank(MB) assigns rank 1 to the *smallest* file. A lower



Figure 1: Visual comparison on the *bicycle* scene (0° camera angle): reference render, best configuration Q16+H+VQ₄₀₉₆, and worst configuration Q8 which introduces severe quantization artifacts.

Table 1: Compression results averaged per benchmark group (T&T: train+truck; Mip-NeRF 360: bicycle+garden; Deep Blending: drjohnson+playroom), all 7 k-iteration checkpoints. **Q8/16**= scalar quantisation; **H**= Hilbert reordering + delta coding + zlib; **VQ_k**= vector quantisation of SH coefficients; **P**= opacity pruning ($\tau=0.08$). Configurations are ranked by Score = $\frac{1}{4}$ rank(PSNR) + $\frac{1}{4}$ rank(SSIM) + $\frac{1}{2}$ rank(MB), computed across all 21 tested configurations (lower is better); the top 10 are shown. Cell shading:

Method	Score↓	T&T				Mip-NeRF 360				Deep Blending			
		PSNR↑	SSIM↑	Ratio↑	MB↓	PSNR↑	SSIM↑	Ratio↑	MB↓	PSNR↑	SSIM↑	Ratio↑	MB↓
Q16+H+VQ₄₀₉₆	5.50	38.88	.986	11.8×	22.7	40.38	.989	12.3×	77.3	42.91	.989	11.5×	37.4
Q16+H+VQ ₂₅₆	6.50	34.57	.979	13.0×	20.8	37.22	.982	13.2×	71.6	38.83	.984	12.5×	34.4
Q16+VQ ₄₀₉₆	8.00	38.92	.986	9.2×	29.2	40.46	.989	9.5×	100.0	43.05	.989	9.4×	45.8
H+VQ ₄₀₉₆	8.50	40.24	.990	5.8×	46.1	41.54	.992	6.0×	158.5	42.97	.989	5.9×	73.0
P+H+VQ ₄₀₉₆ +Q16	8.50	30.44	.920	18.0×	14.8	34.00	.977	19.1×	49.7	37.27	.982	18.5×	23.2
Q16+VQ ₂₅₆	9.00	34.32	.980	10.3×	26.3	37.18	.982	10.3×	91.6	39.03	.983	10.3×	41.7
VQ ₄₀₉₆	9.00	40.28	.990	5.1×	53.2	41.66	.992	5.1×	183.9	43.11	.989	5.1×	84.0
P+H+VQ ₂₅₆ +Q16	9.25	29.44	.915	20.3×	13.3	33.06	.970	20.7×	45.8	35.55	.977	20.3×	21.1
H+VQ ₂₅₆	9.50	35.02	.983	6.1×	44.2	37.74	.985	6.2×	152.8	38.85	.984	6.1×	70.0
Q16	10.00	44.53	.994	2.1×	128.9	46.21	.996	2.1×	450.3	56.60	.999	2.1×	204.6

score is better. Weighting file size at $\frac{1}{2}$ treats compactness as equally important as reconstruction quality.

Best general purpose configuration: Q16+H+VQ₄₀₉₆ (score 5.50) is the clear winner and the recommended default. It achieves a high PSNR score at all benchmarking groups, with SSIM consistently at 0.986–0.989, and compresses to 11–12× the original size. The near-indistinguishable quality relative to the reference is visible in Figure 1. It does not rank first in any single column, but it consistently finishes in the upper half of every ranking. The runner-up, Q16+H+VQ₂₅₆ (score 6.50), achieves a slightly better compression ratio (13×) but loses 3–4 dB of PSNR relative to the winner.

Best quality with moderate compression: H+VQ₄₀₉₆ (score 8.50) achieves the highest PSNR among the top-10 lossy configurations, with SSIM at 0.989–0.992. The tradeoff is a lower compression ratio ($\approx 6\times$). Adding Q16 on top (Q16+VQ₄₀₉₆, score 8.00) reduces file size to a 9–10× ratio at a cost of roughly 1 dB PSNR, making it a better choice when both quality and compactness are required.

High-compression: P+H+VQ₂₅₆+Q16 (score 9.25) achieves the highest compression ratios of all tested configurations. This comes at a significant quality cost where PSNR falls to 29.44 dB on T&T and 33.06 dB on Mip-NeRF 360. P+H+VQ₄₀₉₆+Q16 (score 8.50) recovers 1–2 dB at the cost of roughly 1.5 MB extra per scene and is preferable when some quality loss is tolerable but cannot be severe. Both configurations are only appropriate when file size is the dominant constraint.

Effect of VQ codebook size: Across all configurations, VQ₄₀₉₆ outperforms VQ₂₅₆ by 4–5 dB PSNR while adding less than 8 MB per scene. The larger codebook better represents the full distribution of SH coefficients. The file-size difference between the two codebook sizes is small enough that VQ₄₀₉₆ is preferable in virtually every configuration.

Failure of 8-bit quantization (Q8): Q8 ranks last of all 21 configurations with score 18.50. Its grand-mean PSNR of 18.0 dB is well below any perceptual threshold. The fundamental problem is that position, rotation, and scale attributes are each compressed to a shared 8-bit range, leading to a spatial grid pattern. This causes Gaussians to snap to a discrete lattice, producing large visible clusters and color discontinuities, as shown in Figure 1. Q8 is not recommended for any quality sensitive use case.

5 Conclusion

We presented a four-stage 3DGS compression pipeline combining opacity pruning, Hilbert reordering, SH vector quantization, and 16-bit scalar quantization. The best configuration, Q16+H+VQ₄₀₉₆, achieves 11–12× compression while retaining PSNR above 38.8 dB and SSIM above 0.98 (perceptually close to the uncompressed baseline). However, these ratios fall short of state-of-the-art methods surveyed in [7] and [8], which use more complex compression schemes that exploit inter-Gaussian structure during training.

Several improvements remain for future work. Using separate VQ codebooks per SH frequency band, pruning

Table 2: Compression results averaged per benchmark group (T&T: train+truck; Mip-NeRF 360: bicycle+garden; Deep Blending: drjohnson+playroom), all 7 k-iteration checkpoints. **Q8/16**= scalar quantisation; **H**= Hilbert reordering + delta coding + zlib; **VQ_k**= vector quantisation of SH coefficients; **P**= opacity pruning ($\tau=0.08$). Configurations are ranked by Score = $\frac{1}{4}$ rank(PSNR) + $\frac{1}{4}$ rank(SSIM) + $\frac{1}{2}$ rank(MB), computed across all 21 tested configurations (lower is better). Cell shading: 1st 2nd 3rd per column.

Method	Score↓	T&T				Mip-NeRF 360				Deep Blending			
		PSNR↑	SSIM↑	Ratio↑	MB↓	PSNR↑	SSIM↑	Ratio↑	MB↓	PSNR↑	SSIM↑	Ratio↑	MB↓
Q16+H+VQ₄₀₉₆	5.50	38.88	.986	11.8×	22.7	40.38	.989	12.3×	77.3	42.91	.989	11.5×	37.4
Q16+H+VQ ₂₅₆	6.50	34.57	.979	13.0×	20.8	37.22	.982	13.2×	71.6	38.83	.984	12.5×	34.4
Q16+VQ ₄₀₉₆	8.00	38.92	.986	9.2×	29.2	40.46	.989	9.5×	100.0	43.05	.989	9.4×	45.8
H+VQ ₄₀₉₆	8.50	40.24	.990	5.8×	46.1	41.54	.992	6.0×	158.5	42.97	.989	5.9×	73.0
P+H+VQ ₄₀₉₆ +Q16	8.50	30.44	.920	18.0×	14.8	34.00	.977	19.1×	49.7	37.27	.982	18.5×	23.2
Q16+VQ ₂₅₆	9.00	34.32	.980	10.3×	26.3	37.18	.982	10.3×	91.6	39.03	.983	10.3×	41.7
VQ ₄₀₉₆	9.00	40.28	.990	5.1×	53.2	41.66	.992	5.1×	183.9	43.11	.989	5.1×	84.0
P+H+VQ ₂₅₆ +Q16	9.25	29.44	.915	20.3×	13.3	33.06	.970	20.7×	45.8	35.55	.977	20.3×	21.1
H+VQ ₂₅₆	9.50	35.02	.983	6.1×	44.2	37.74	.985	6.2×	152.8	38.85	.984	6.1×	70.0
Q16	10.00	44.53	.994	2.1×	128.9	46.21	.996	2.1×	450.3	56.60	.999	2.1×	204.6
Q16+H	10.00	44.48	.994	2.9×	91.2	46.20	.996	2.9×	327.9	56.59	.999	2.8×	151.1
H	11.00	68.95	1.000	1.2×	224.1	70.28	1.000	1.2×	808.6	73.43	1.000	1.2×	356.1
VQ ₂₅₆	11.00	34.73	.983	5.4×	50.3	37.72	.985	5.4×	175.6	39.05	.984	5.4×	79.8
P+H+VQ ₄₀₉₆	11.50	30.68	.924	9.1×	29.4	34.24	.979	9.4×	100.4	37.27	.982	9.7×	44.4
P+H+VQ ₂₅₆	12.00	29.62	.918	9.6×	27.9	33.25	.973	9.8×	96.5	35.55	.977	10.2×	42.4
P _{0.12} +H+VQ ₂₅₆	12.50	27.23	.901	11.3×	23.6	30.56	.959	11.8×	80.0	33.68	.968	12.2×	35.2
P _{0.12} +H+VQ ₄₀₉₆	12.50	27.82	.906	10.6×	25.0	31.03	.966	11.3×	83.4	34.69	.972	11.6×	37.1
P+Q16+H	14.00	30.92	.925	4.6×	57.8	34.60	.983	4.6×	207.3	38.73	.989	4.7×	91.1
P	16.00	31.21	.930	1.7×	162.1	34.91	.986	1.7×	565.4	38.76	.990	1.7×	247.3
P _{0.12}	18.25	28.14	.912	1.9×	137.0	31.27	.971	2.0×	468.1	35.32	.979	2.1×	205.5
Q8	18.50	13.62	.490	4.2×	64.5	14.50	.379	4.2×	225.1	24.19	.760	4.2×	102.3

higher-order SH coefficients, and replacing zlib with a stronger entropy coder could all improve the compression ratio without significant quality loss. Broader evaluation across more scenes and full 30k-iteration checkpoints would also give a more complete picture of generalization. Nonetheless, the results show that simple pipelines can meaningfully compress 3DGS models while preserving visual quality.

6 Acknowledgment

The research was conducted within the projects Cell visualization of unified microscopic data and procedurally generated sub-cellular structures (J2-50221) and Geospatial Information Technologies for Resilient and Sustainable Society (GeoAI) (GC-0006), both funded by the Slovenian Research and Innovation Agency (ARIS).

References

- [1] B. Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics* 42.4 (2023). DOI: 10.1145/3592433.
- [2] B. Mildenhall et al. “NeRF: representing scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* 65.1 (2021), pp. 99–106. DOI: 10.1145/3503250.
- [3] A. Kristan et al. “Neural-network-based volume compression”. In: *ERK 2024*. Ljubljana, 2024, str. 452–455.
- [4] Ž. Lesar et al. “Blocky volume package : a web-friendly volume storage and compression solution”. In: *WSCG 2023*. Plzen, 2023, str. 213–221. DOI: 10.24132/CSRN.3301.25.
- [5] A. Jočić et al. “Polygonal mesh connectivity compression using graph symmetries”. In: *ERK 2024*. Ljubljana, 2024, str. 423–426.
- [6] R. Oblak et al. “Web-Based Vascular Flow Simulation Visualization with Lossy Data Compression for Fast Transmission”. In: *AVR 2018*. Cham: Springer, 2018, pp. 3–17. DOI: 10.1007/978-3-319-95282-6_1.
- [7] M. T. Bagdasarian et al. “3DGS.zip: A survey on 3D Gaussian Splatting Compression Methods”. In: *Computer Graphics Forum* 44.2 (2025), e70078. DOI: <https://doi.org/10.1111/cgf.70078>.
- [8] M. Salman Ali et al. “Compression in 3D Gaussian Splatting: A Survey of Methods, Trends, and Future Directions”. In: *IEEE Transactions on Circuits and Systems for Video Technology* (2026), pp. 1–1. DOI: 10.1109/TCSVT.2026.3676048.
- [9] C. Cao et al. “3D Point Cloud Compression: A Survey”. In: *Proceedings of the 24th International Conference on 3D Web Technology*. Web3D ’19. LA, CA, USA: Association for Computing Machinery, 2019, pp. 1–9. DOI: 10.1145/3329714.3338130.
- [10] Z. Fan et al. “LightGaussian: unbounded 3D Gaussian compression with 15x reduction and 200+ FPS”. In: *Proceedings of the 38th International Conference on Neural Information Processing Systems*. NIPS ’24. Vancouver, BC, Canada: Curran Associates Inc., 2024. DOI: 10.5555/3737916.3742363.
- [11] S. Niedermayr et al. “Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 20623–20632. DOI: 10.1109/CVPR52733.2024.00985.
- [12] 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 (CVPR)*. 2024, pp. 20654–20664. DOI: 10.1109/CVPR52733.2024.01952.
- [13] A. Knapitsch et al. “Tanks and temples: benchmarking large-scale scene reconstruction”. In: *ACM Transactions on Graphics* 36.4 (2017). DOI: 10.1145/3072959.3073599.
- [14] J. T. Barron et al. “Mip-nerf 360: Unbounded anti-aliased neural radiance fields”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 5470–5479. DOI: 10.1109/CVPR52688.2022.00539.
- [15] P. Hedman et al. “Deep blending for free-viewpoint image-based rendering”. In: *ACM Transactions on Graphics* 37.6 (2018). DOI: 10.1145/3272127.3275084.