

On Improving the Compression Efficiency of C3DGS

Luís Serra and Renam Silva

Abstract—3D Gaussian Splatting (3DGS) enables real-time novel view synthesis but produces scene representations of several hundred megabytes, limiting practical deployment. This work proposes four enhancements to the C3DGS compression pipeline: softmax-based centroid weighting for sensitivity-aware clustering, two reconstruction-error sensitivity measures, adaptive arithmetic coding, and an entropy-driven criterion for codebook size selection. Each is evaluated over 5 runs per scene with paired significance testing. The combined model achieves a 21% size reduction relative to C3DGS for 0.25 dB of PSNR across standard benchmarks.

Keywords—3D Gaussian Splatting, compression, vector quantization, arithmetic coding, entropy, novel view synthesis.

I. INTRODUCTION

3D Gaussian Splatting (3DGS) [1] has been established as a competitive approach for real-time novel view synthesis, offering an alternative to implicit Neural Radiance Field (NeRF) [2] representations with an explicit set of anisotropic Gaussian primitives rendered via differentiable α -blending. A scene is parametrized by millions of Gaussians, each described by a position, rotation, scaling, opacity, and spherical harmonic (SH) coefficients. This explicit design enables rendering at over 100 frames per second [1] but yields uncompressed scene files of hundreds of megabytes to multiple gigabytes [3], posing a significant challenge for network streaming, mobile, and VR/AR headsets with constrained power and storage.

Existing compression methods often fall into two categories. *Value-based* methods exploit redundancy within individual Gaussian parameters through pruning, vector quantization, and scalar quantization. *Structure-based* methods exploit dependencies among primitives, constructing conditional probability distributions over quantized attribute values so that entropy coding also captures inter-Gaussian correlations [4], [3].

Within the value-based category, C3DGS [5] encodes SH coefficients and covariance matrix parameters into compact codebooks via sensitivity-aware k-means clustering, combined with quantization-aware fine-tuning and DEFLATE to code the resulting codebooks and side information, achieving $31\times$ compression with only 0.23 dB average PSNR loss. Such methods operate directly on the standard 3DGS format, making them broadly applicable and reproducible; their principal limitation is that, by treating each primitive independently, they leave spatial correlations among neighboring Gaussians unexploited.

Luís Serra, Electrical Engineering Department, Universidade de Brasília, e-mail: luis.serra@aluno.unb.br; Renam Silva, Digital Signal Processing Group, Electrical Engineering Graduate Program, Electrical Engineering Department, Universidade de Brasília, e-mail: renam.silva@unb.br. This work was partially supported by Programa de Iniciação Científica da Universidade de Brasília (ProIC/UnB) & Grupo de Processamento Digital de Sinais (GPDS-UnB)

Structure-based methods address this limitation at the cost of an appreciable increase in complexity. HAC++ [6], the state-of-the-art in this category, builds upon Scaffold-GS [7], jointly training a binarized multi-resolution hash grid with the scene and using MLP-based context models to predict per-attribute probability distributions for arithmetic entropy coding, achieving over $100\times$ compression. FCGS [4] instead compresses in a single feed-forward pass without per-scene optimization, reducing compression time from minutes to seconds, but its rate-distortion performance remains below that of optimization-based methods. As pointed out by Bagdasarian et al. [3], no single approach currently dominates across compression ratio, compression speed, and computational burden.

This work focuses on the value-based paradigm and proposes four enhancements to C3DGS: a softmax replacement for the linear sensitivity weighting in the centroid update; reconstruction-error-based sensitivity measures as alternatives to the image-energy gradient; adaptive arithmetic coding as a lossless drop-in replacement for DEFLATE; and an entropy-driven criterion for codebook size selection. Each is evaluated independently on the pre-trained scene models used by C3DGS, with quality measured by Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS), following the benchmark of [3].

II. METHODOLOGY

Leveraging the C3DGS reference implementation [5], four independent modifications were devised to increase compression efficiency while preserving visual quality, each evaluated separately to isolate its contribution. For all experiments, pre-trained 3DGS models were used as input to ensure a fair comparison with the baseline. Image quality was measured using PSNR, SSIM, and LPIPS on the test split of each scene, following the evaluation protocol of Bagdasarian et al. [3]. Because the differences between configurations are small, every configuration was executed 5 times per scene with independent seeds over the 13 benchmark scenes. For each configuration, the result attributed to a scene is the mean of the values measured by its individual seeds, the comparison against the baseline is then paired scene by scene. Significance is assessed with the two-sided Wilcoxon signed-rank test over the $n = 13$ scene pairs, Holm–Bonferroni corrected across the family of comparisons, and effect magnitude by the paired effect size d_z ; where a confidence interval is quoted, it is a percentile bootstrap 95% interval on the mean paired difference. This protocol follows the established practice for comparing two algorithms over a suite of benchmark instances, where the

non-parametric paired test avoids distributional assumptions that a handful of scenes cannot support [8]. Results are further compared with Compact3D [9] and MesonGS [10], two value-based methods that likewise operate on the standard 3DGS format.

III. PROPOSED MODIFICATIONS

A. Softmax Weighting for Sensitivity-Aware Vector Clustering

The C3DGS clustering scheme assigns each Gaussian attribute vector $\mathbf{x}$ the weighted distance $\mathcal{D}(\mathbf{x}, \mathbf{c}_k) = S(\mathbf{x})\|\mathbf{x} - \mathbf{c}_k\|_2^2$ to a centroid $\mathbf{c}_k$, where $S(\mathbf{x}) = \max_{d \in [1..D]} S(x_d)$ is the maximum sensitivity over the components of the vector, and updates the centroid as the sensitivity-weighted mean of all assigned vectors. Because $S(\mathbf{x})$ enters linearly, in a cluster whose sensitivity values span a narrow range the resulting centroid is almost identical to the unweighted mean. We propose instead a softmax-based weighted mean:

$$\mathbf{c}_k = \sum_{\mathbf{x}_i \in \mathcal{A}(k)} \frac{\exp(S(\mathbf{x}_i))}{\sum_{\mathbf{x}_j \in \mathcal{A}(k)} \exp(S(\mathbf{x}_j))} \cdot \mathbf{x}_i \quad (1)$$

where $\mathcal{A}(k)$ is the set of vectors assigned to centroid k . The softmax amplifies the relative weight of the most sensitive vectors within the cluster. This is motivated by the highly skewed sensitivity distribution pointed out in [5]: fewer than 5% of all SH vectors exhibit a measurable impact on the rendered image, yet they determine the perceptual fidelity of the compressed scene. With linear weighting the low-sensitivity majority dilutes the centroid position, whereas the softmax concentrates the update on those few vectors.

B. Alternative Sensitivity Measures for Parameter Importance Estimation

The C3DGS sensitivity score [5] is based on the absolute gradient of each parameter with respect to the total rendered image energy E_i , which reflects pixel intensity rather than reconstruction accuracy, so Gaussians in well-reconstructed regions may receive high scores despite contributing negligibly to perceived quality. Two alternatives based on reconstruction error were evaluated, both defined as pixel-wise errors between the original image I_i^o and the rendered image I_i^r : an absolute-difference loss $\mathcal{L}_i^{\text{abs}} = \sum |I_i^o - I_i^r|$ and a squared-difference loss $\mathcal{L}_i^{\text{sq}} = \sum (I_i^o - I_i^r)^2$. The sensitivity is recomputed as $S(p) \propto \sum_i |\partial \mathcal{L}_i / \partial p|$ normalized by pixel count. Both variants concentrate non-zero gradients on parameters whose deviation drives the rendered image away from the ground truth, the squared version additionally penalizing large residuals and diminishing small ones. All other hyperparameters were kept as C3DGS defaults, including the importance threshold that classifies high-sensitivity vectors as *direct* codebook entries, so only the distribution of sensitivity scores changes.

C. Arithmetic Coding of Quantized Gaussian Parameters

C3DGS encodes the codebooks, indexes, and side information with DEFLATE, that is, LZ77 dictionary coding followed

by Huffman coding [5]. We replace it with adaptive arithmetic coding (AC), already adopted by other authors such as HAC++ [6], whose frequency tables are updated after every symbol and therefore need no separate distribution estimation pass. Two parameter groups are AC-coded; all others remain on DEFLATE.

VQ indexes. Each Gaussian carries either a *clustered* index in $[0, K - 1]$, pointing to a centroid shared by many Gaussians, or a *direct* index, assigned to a high-sensitivity Gaussian excluded from clustering and stored individually. Direct indexes occur exactly once and carry no statistical redundancy, so a binary mask separates the two subsets and only the clustered indexes are AC-coded; the mask and the direct indexes stay on DEFLATE.

Codebook entries. Scaling, rotation, and SH DC and AC coefficients are scalar quantized and their bins AC-coded, each parameter with its own model, initialized uniform and adapted during coding. The SH AC coefficients use one model per coefficient, matching their approximately Laplace distribution.

Decoding reverses the process in encoding order with identically initialized models, recombines the two index subsets through the stored mask, and writes the result in the format expected by the standard 3DGS loader.

D. Codebook Size Selection via Shannon Entropy Analysis

The codebook size K is one of the main hyperparameters governing the rate-distortion trade-off of the vector quantization stage: a larger codebook approximates the original parameters more finely at the cost of more bits per index and a larger stored codebook. C3DGS uses $K = 4096$ for both color and covariance codebooks across all scenes [5]; since scenes may differ substantially in parameter redundancy, a fixed K is likely suboptimal.

Following Huang et al. [11], who show that the efficiency of lossless entropy coding on 3DGS attributes is governed by the spread of their distributions, we propose the average empirical Shannon entropy $\bar{H}$ of the integer parameters of the *uncompressed* model as a scene-specific complexity measure. This choice admits a rate-distortion reading [12]: $\log_2 K$ is the rate available to each vector-quantized index, a budget reached in full only when the codebook entries are used with uniform probability, whereas $\bar{H}$ estimates the information content of the attribute populations. Requiring the available rate to cover that content therefore places a lower bound on $\log_2 K$. This makes the ratio $\bar{H} / \log_2 K$ a dimensionless variable in which the trade-off is naturally expressed: it grows as K shrinks and the rate spent on each index falls, and the quality drop grows linearly with it over the regime of interest (Eq. (2)), so bounding the admissible drop also bounds how far the rate can be lowered. It is therefore worth investigating whether this ratio can predict the smallest K that satisfies a given quality constraint without exhaustive search. For each attribute $a \in \{\text{SH's AC, rotation, scaling, opacity}\}$, the empirical Shannon entropy of the N -bit quantized attribute values is computed as $H_a = -\sum_v p_a(v) \log_2 p_a(v)$, with v ranging over the 2^N quantization levels and $p_a(v)$ the empirical probability of symbol v in a , estimated from the full population of Gaussians

in the scene. The average entropy $\bar{H}$ is the arithmetic mean of H_a over all attributes.

Each scene is compressed at codebook sizes $K \in \{4, \dots, 4096\}$ for both color and covariance matrix codebooks, and the resulting quality metrics are recorded alongside $\bar{H}$. The analysis seeks a systematic relationship between $\bar{H}$ and the minimum codebook size that keeps the quality loss within a desired threshold, which would allow K to be selected from a single measurement on the uncompressed model, removing manual tuning and reducing the computational burden whenever the selected K is below the default.

IV. RESULTS

Table I presents the quantitative comparison and Table II the paired significance tests. Since the quality differences are of the order of tenths of a decibel, they must be read against the noise floor of the pipeline: over 5 seeds the per-scene run-to-run standard deviation averages 0.035 dB for PSNR, 4.9×10^{-4} for SSIM, 5.7×10^{-4} for LPIPS and 0.048 MB for size, reaching at most 0.112 dB, 1.1×10^{-3} , 1.5×10^{-3} and 0.138 MB on individual scenes. Summarizing each configuration by its seed mean divides this variability by $\sqrt{5}$, and every comparison below is made on those means, which is what allows differences of this magnitude to be resolved.

In Table I the first three rows are as published by their authors, whereas *C3DGS (reproduced)* is our own run of the reference implementation and the baseline every modification is compared against. The two rightmost columns give compression and decompression time per scene averaged over the 13 scenes, measured on an RTX 3090 with the CPU arithmetic coder; the four DEFLATE-based configurations share the same sub-second decoder.

In Table II, pairing is per scene on the seed mean ($n = 13$) and each cell gives the mean paired difference Δ with the effect size d_z in parentheses. All differences are significant at $p \leq 0.005$ under the Holm-corrected Wilcoxon test; PSNR is in dB, SSIM and LPIPS in units of 10^{-3} , and size as a per-scene percentage of the baseline. No quality comparison is reported for the arithmetic coding row, as that stage is lossless and cannot alter the reconstruction.

A. Softmax Weighting

The softmax centroid update trades a small, consistent quality loss for a small, consistent size reduction. Averaged over the 13 scenes the model shrinks by 0.39 MB, or 1.4%, while every quality metric degrades slightly: -0.039 dB PSNR, -0.0012 SSIM and $+0.0017$ LPIPS. All four differences are significant after Holm correction (Table II), yet all are marginal in absolute terms, the PSNR loss being comparable to the run-to-run standard deviation. The direction of the effect is uniform across datasets, so the amplification of high-sensitivity contributions behaves as intended, shifting the codebook towards denser coverage of high-sensitivity regions at the cost of a slightly coarser representation elsewhere.

B. Sensitivity Measures

Both alternative sensitivity measures reduce model size relative to the baseline, trading size against image quality in a manner consistent with the aggressiveness of each loss function.

The absolute-difference measure (**Sens. Abs**) achieves an average size reduction of 11.1% at a cost of -0.111 dB PSNR (95% CI $[-0.158, -0.067]$), with SSIM and LPIPS shifting by -0.0036 and $+0.0051$ (Table II): by directing the sensitivity budget only towards parameters that contribute to visible reconstruction errors, it identifies a broader set of parameters as compressible while largely preserving perceptual structure. The contrast between the effect sizes is what makes this variant attractive, the size gain being a very large effect ($d_z = -3.2$) while the quality losses remain moderate ($|d_z| \leq 1.9$).

The squared-difference measure (**Sens. SQ**) achieves a larger average size reduction of 21.6% at the cost of a more pronounced PSNR loss of -0.372 dB. LPIPS degrades more noticeably: for residuals in $[0, 1]$, $x^2 < x$, so the squared formulation down-weights small errors and concentrates the sensitivity signal on the largest residuals, over-compressing the majority of Gaussians while protecting only those responsible for the largest artifacts.

A secondary effect of maintaining the original importance threshold is that a larger fraction of Gaussians passes through the clustering path rather than being stored as direct entries. This effect is most pronounced for color attributes, whose importance threshold in C3DGS is lower than that of geometry attributes, making them more susceptible to reclassification when the sensitivity scores are redistributed by a different loss function. The additional Gaussians entering the clustering path contribute to the observed size reductions, but the precise interaction between the choice of sensitivity loss and the importance threshold falls outside the scope of the present investigation and is identified as a direction for future work.

C. Arithmetic Coding

Replacing DEFLATE with adaptive arithmetic coding reduces the model size by 1.33 MB on average, or 4.7%, significantly across scenes ($p = 4.2 \times 10^{-3}$, $d_z = -4.3$; Table II). It is noted that the major space improvement comes from the two vector-quantization index streams, whose clustered values follow the skewed distribution an adaptive model exploits; on the scalar-quantized attribute arrays, closer to uniform, arithmetic coding is at best marginal and occasionally worse than DEFLATE, so a selective per-field choice of coder is a natural refinement left as future work.

Quality is unchanged by construction, the stage being lossless, so no quality comparison is reported for it in Table II. As a direct check, the decoded model of every scene was reloaded and re-evaluated, reproducing the encoder-side metrics.

This approach has two practical limitations. First, arithmetic coding is a sequential algorithm: of the 1338 s reported in Table I, the coding stage alone accounts for 1105 s against 233 s for the entire remaining pipeline, so enabling it multiplies the compression time by $5.6\times$, while decoding averages 1423 s where DEFLATE decompresses the same model in

TABLE I

 COMPARISON ACROSS MODELS. PSNR ($\uparrow$), SSIM ($\uparrow$), LPIPS ($\downarrow$), SIZE IN MB ($\downarrow$), AND COMPRESSION AND DECOMPRESSION TIME IN SECONDS ($\downarrow$).

Model	TanksAndTemples				MipNeRF360				DeepBlending				Time (s)	
	PSNR	SSIM	LPIPS	Size	PSNR	SSIM	LPIPS	Size	PSNR	SSIM	LPIPS	Size	Comp.	Dec.
Compact3D 32K [9]	23.44	0.838	0.198	13.0	27.12	0.806	0.240	19.0	29.90	0.907	0.251	13.5	–	–
MesonGS c3 [10]	23.29	0.835	0.197	17.4	26.99	0.797	0.246	25.9	29.48	0.903	0.252	29.0	–	–
C3DGS [5]	23.32	0.832	0.194	17.3	26.98	0.801	0.238	28.8	29.38	0.898	0.253	25.3	–	–
C3DGS (reproduced)	23.23	0.832	0.196	17.3	26.97	0.801	0.240	28.8	29.37	0.899	0.257	25.3	240	< 1
C3DGS + Softmax	23.23	0.831	0.197	17.1	26.92	0.799	0.242	28.4	29.35	0.898	0.258	24.9	242	< 1
C3DGS + Sens. Abs	23.18	0.829	0.200	15.2	26.83	0.796	0.246	26.1	29.35	0.898	0.259	22.8	238	< 1
C3DGS + Sens. SQ	23.03	0.822	0.211	12.4	26.51	0.789	0.255	24.1	29.25	0.896	0.262	20.6	237	< 1
C3DGS + AC encoding	23.23	0.832	0.196	16.6	26.97	0.801	0.240	27.3	29.37	0.899	0.257	23.9	1338	1423
C3DGS (K=512)	23.20	0.830	0.198	16.5	26.89	0.798	0.244	27.1	29.33	0.898	0.258	23.9	197	< 1
Ours full	23.12	0.827	0.203	13.8	26.66	0.791	0.253	23.0	29.27	0.897	0.261	20.1	526	560

TABLE II

PAIRED COMPARISON OF EACH MODIFICATION AGAINST THE REPRODUCED C3DGS BASELINE.

Model	PSNR	SSIM($\times 10^{-3}$)	LPIPS($\times 10^{-3}$)	Size(%)
+ Softmax	-.039 (-1.3)	-1.2 (-1.5)	+1.7 (+2.3)	-1.4 (-5.5)
+ Sens. Abs	-.111 (-1.3)	-3.6 (-1.6)	+5.1 (+1.9)	-11.1 (-3.2)
+ Sens. SQ	-.372 (-1.2)	-9.8 (-2.1)	+13.6 (+2.5)	-21.6 (-2.3)
+ AC	–	–	–	-4.7 (-4.3)
Ours full	-.247 (-1.5)	-7.6 (-1.7)	+10.6 (+2.2)	-20.9 (-10.1)

under a second. The coder used here is an unoptimized *Python* CPU implementation and an efficient GPU implementation would narrow this gap, though [6] reports that even a highly optimized one still requires several seconds per scene. The remaining stages scale more favourably: each clustering iteration scales with K , so selecting one below the default 4096 via Eq. (3) also reduces clustering time. Second, the size reduction applies only to the on-disk storage representation: at render time the model must be fully decoded, so the memory footprint during inference is identical to that of the C3DGS baseline. The benefit is therefore limited to transmission and storage scenarios.

D. Codebook Size

The codebook size experiment reveals a consistent pattern: quality degrades as K decreases, though not linearly in K . Plotted against the ratio $\bar{H}/\log_2 K$, the data exhibits a linear trend across scenes in the regime of small quality degradation (Fig. 1(a)), which motivates the empirical model of Eq. (2); inverting it gives Eq. (3), a closed-form expression for the minimum codebook size that keeps the quality loss within a prescribed tolerance:

$$Q_{\%}^{\text{drop}} = A\bar{H}/\log_2 K + B, \quad \log_2 K = \left\lceil A\bar{H}/(Q_{\%}^{\text{drop}} - B) \right\rceil. \quad (2) \quad (3)$$

Here A and B are coefficients obtained by linear regression on the observed data, and $Q_{\%}^{\text{drop}}$ is the normalized quality drop relative to the $K=4096$ baseline. This allows the codebook size to be determined from a single entropy measurement of the uncompressed model, without fine-tuning or exhaustive search. Eq. (2) is valid only in the regime of small quality degradation, where the linear approximation holds. Panel (b) of Fig. 1 makes that boundary visible: the dispersion is negligible

for large K and widens sharply below $K = 32$, where the trajectories in panel (a) also bend away from the straight line.

Among the three quality metrics, LPIPS demonstrates the most pronounced degradation as K decreases. This aligns with its perceptual nature, which detects structural and geometric distortions from aggressive codebook quantization more acutely than signal-fidelity metrics. Consequently, LPIPS is adopted as the governing metric for codebook size selection in the full model, establishing the most conservative and perceptually relevant lower bound on the minimum K required to satisfy quality constraints. Fitted on the seed mean of each scene-codebook pair, it gives $A = 0.0660$ and $B = 0.9751$, with a slope that remains significant under standard errors clustered by scene ($p = 3.0 \times 10^{-4}$) and a coefficient of determination $R^2 = 0.49$.

This moderate R^2 does not invalidate the criterion, because Eq. (2) is not intended to predict the quality of a particular scene but to fix one codebook size for a population of scenes. The unexplained variance is dominated by the scene-to-scene spread of the intercept, visible in Fig. 1(a) as the vertical offset between trajectories, which is precisely what the tolerance $Q_{\%}^{\text{drop}}$ absorbs. A leave-one-scene-out check indicates that the coefficients are not tuned to the scenes used to fit them: re-estimating A and B on 12 scenes and applying Eq. (3) to the held-out one selects the same K in 12 of the 13 cases.

Its limits should be stated plainly. Across the 13 scenes $\bar{H}$ spans only 5.55 to 6.00 bits, so Eq. (3) selects $K = 512$ for every scene: the criterion is consistent, but its scene-adaptivity cannot be demonstrated on a set of such similar entropy. At that K the measured LPIPS degradation has a median of 1.3% and stays within the 2% tolerance for 10 of the 13 scenes, the exceptions reaching at most 3.7%. The narrow spread of $\bar{H}$ also masks a wider dispersion among its terms, since the individual attribute groups differ appreciably in entropy: averaging them into a single scalar discards information that a per-attribute criterion could exploit. Assigning a different codebook size to each parameter group, as done in Compact3D [9], is therefore a natural refinement of Eq. (3). That refinement, and whether the coefficients generalize to large-scale reconstructions where $\bar{H}$ would span a wider range, are left for future work.

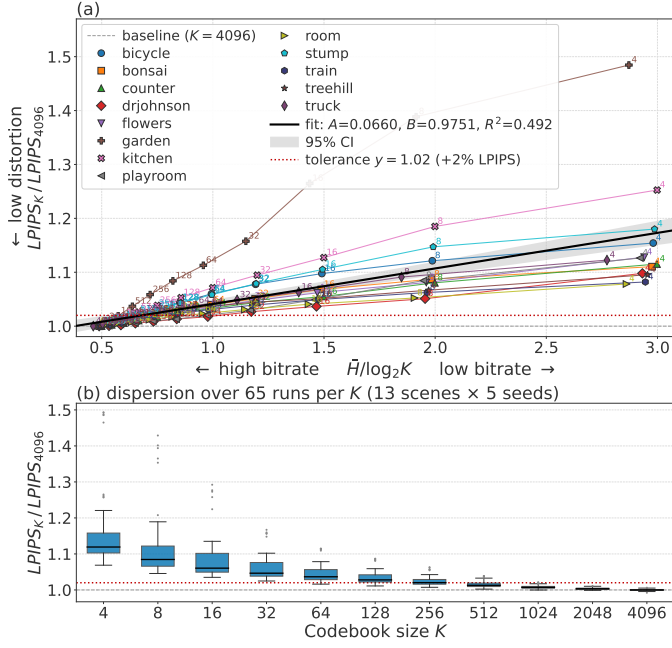


Fig. 1. LPIPS normalized by the $K=4096$ value of the same scene. (a) One trajectory per scene, annotated with K , over the fit of Eq. (2) to the seed means of all 143 scene-codebook pairs; the band is the 95% CI of the mean response. (b) Dispersion behind those means, 65 runs (13 scenes $\times$ 5 seeds).

E. Full Model

The full model integrates all four modifications: softmax centroid weighting, absolute-difference sensitivity, entropy-based codebook size selection via Eq. (3) with the coefficients above and tolerance $Q_{\%}^{\text{drop}} = 1.02$ (hence $K = 512$), and adaptive arithmetic coding. It reduces the size by 5.36 MB, or 20.9%, relative to the reproduced baseline, and the reduction is uniform across the three datasets, at an average PSNR loss of 0.247 dB (95% CI $[-0.336, -0.164]$), an SSIM loss of 0.0076 and an LPIPS increase of 0.0106, or 4.5%; all four differences are significant after Holm correction (Table II). As anticipated from the analysis in Section IV-D, LPIPS is the most sensitive metric and shows the largest relative degradation.

The four components act on different parts of the pipeline, which is what allows them to be combined. The sensitivity measure and the codebook size both reduce the bits produced by the vector quantization stage, the first by reclassifying more Gaussians as compressible, the second by shortening the indexes and the stored codebook. The softmax weighting changes where the codebook spends its resolution. Finally, arithmetic coding operates downstream of all three, on whatever symbol distribution they produce.

Regarding Eq. (3) the tolerance constrains only the codebook-size term, at $K = 512$ the codebook contributes a median 1.3% of LPIPS degradation, whereas the 4.5% of the full model accumulates all four components. The practical benefit of the smaller codebook is twofold. It reduces the cost of the clustering stage itself, compression takes 197 s at $K = 512$ against 240 s at $K = 4096$, an 18% reduction equivalent to a $1.22\times$ speedup. It also shortens the symbol streams the entropy coder must process, the stage drops from 1105 s to 335 s per scene, so the full model compresses in

526 s, or $2.2\times$ the DEFLATE baseline rather than the $5.6\times$ measured at $K = 4096$. Regarding the methods of Table I, it is competitive with Compact3D on TanksAndTemples but larger on the others, at comparable PSNR.

V. CONCLUSIONS

This work investigated four enhancements to the C3DGS compression pipeline, each evaluated over 5 runs per scene and compared against the baseline with paired significance tests. Softmax centroid weighting and reconstruction-error sensitivity measures provide compression gains, the absolute-difference offering the best balance at 11.1% smaller models for 0.11 dB of PSNR. Adaptive arithmetic coding reduces file size by 4.7% with no measurable quality cost, at the expense of a $5.6\times$ increase in compression time. An empirical entropy-to-codebook-size mapping enables selecting a suitable K without exhaustive hyperparameter search, the slope is consistent across all 13 scenes and remains significant under scene-clustered standard errors, though the modest entropy spread of the benchmark limits how much scene-adaptivity can be demonstrated. Since the clustering cost scales with K and a smaller codebook also leaves fewer symbols to code, a wiser codebook selection substantially offsets the overhead introduced by arithmetic coding, bringing the combined pipeline to $2.2\times$ the DEFLATE baseline against the $5.6\times$ of AC alone. The combined model achieves 20.9% additional compression for 0.25 dB of PSNR and a 4.5% increase in LPIPS, small differences that are nonetheless significant rather than lost in run-to-run noise. Generalization to large-scale scenes, an intermediate sensitivity exponent $n \in (1, 2)$, a GPU implementation of the entropy coder, extending the entropy coding to the Gaussian positions, choosing the coder per field, and re-calibration of the importance threshold under reconstruction-error sensitivity are identified as directions for future work.

REFERENCES

- [1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3D Gaussian splatting for real-time radiance field rendering,” *ACM Trans. Graph.*, vol. 42, no. 4, 2023.
- [2] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” *Commun. ACM*, vol. 65, no. 1, pp. 99–106, 2021.
- [3] M. T. Bagdasarian *et al.*, “3DGS.zip: A survey on 3D Gaussian splatting compression methods,” *arXiv:2407.09510*, 2025.
- [4] Y. Chen, Q. Wu, M. Li, W. Lin, M. Harandi, and J. Cai, “Fast feedforward 3D Gaussian splatting compression,” in *Proc. ICLR*, 2025.
- [5] S. Niedermayr, J. Stumpfegger, and R. Westermann, “Compressed 3D Gaussian splatting for accelerated novel view synthesis,” in *Proc. CVPR*, pp. 10349–10358, 2024.
- [6] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, “HAC++: Towards 100X compression of 3D Gaussian splatting,” *arXiv:2501.12255*, 2025.
- [7] T. Lu *et al.*, “Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering,” in *Proc. CVPR*, 2024.
- [8] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.
- [9] K. L. Navaneet, K. P. Meibodi, S. A. Koohpayegani, and H. Pirsiavash, “Compact3D: Smaller and faster Gaussian splatting with vector quantization,” *arXiv:2311.18159*, 2023.
- [10] S. Xie *et al.*, “MesonGS: Post-training compression of 3D Gaussians via efficient attribute transformation,” in *Proc. ECCV*, 2024.
- [11] Y. Huang, J. Pang, F. Zhu, and D. Tian, “EntropyGS: An efficient entropy coding on 3D Gaussian splatting,” *arXiv:2508.10227*, 2025.
- [12] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer, 1992.