GaussianImage++: Boosted Image Representation and Compression with 2D Gaussian Splatting

Tiantian Li[1], Xinjie Zhang[2,3], Xingtong Ge[3], Tongda Xu[1], Dailan He[4], Jun Zhang[3], Yan Wang[1]*

[1]Institute for AI Industry Research (AIR), Tsinghua University
[2]Microsoft Research Asia
[3]The Hong Kong University of Science and Technology
[4]The Chinese University of Hong Kong
litiantian2er@hnu.edu.cn, wangyan@air.tsinghua.edu.cn

Abstract

Implicit neural representations (INRs) have achieved remarkable success in image representation and compression, but they require substantial training time and memory. Meanwhile, recent 2D Gaussian Splatting (GS) methods (*e.g.*, GaussianImage) offer promising alternatives through efficient primitive-based rendering. However, these methods require excessive Gaussian primitives to maintain high visual fidelity. To exploit the potential of GS-based approaches, we present GaussianImage++, which utilizes limited Gaussian primitives to achieve impressive representation and compression performance. Firstly, we introduce a distortion-driven densification mechanism. It progressively allocates Gaussian primitives according to signal intensity. Secondly, we employ context-aware Gaussian filters for each primitive, which assist in the densification to optimize Gaussian primitives based on varying image content. Thirdly, we integrate attribute-separated learnable scalar quantizers and quantization-aware training, enabling efficient compression of primitive attributes. Experimental results demonstrate the effectiveness of our method. In particular, GaussianImage++ outperforms GaussianImage and INRs-based COIN in representation and compression performance while maintaining real-time decoding and low memory usage.

Code —
https://github.com/Sweethyh/GaussianImage_plus.git

Introduction

Neural image representations and compression have recently emerged as promising techniques for storing, streaming, and rendering visual data. Most neural compression models (Ballé et al. 2018; Cheng et al. 2020; He et al. 2022) are based on autoencoders, with an encoder mapping an image to a high-dimensional but low-resolution latent, a quantizer processing the latent representation for storage and transmission, and a decoder decoding a lossy reconstruction from the quantized latent.

Implicit neural representations (INRs) (Sitzmann et al. 2020; Dupont et al. 2021, 2022) have significantly shifted this paradigm. Assisted with lightweight multi-layer perceptrons (MLPs), they have demonstrated remarkable visual fidelity across various applications. However, they suffer from large memory overhead and long training time. Recently, Gaussian Splatting (GS) (Kerbl et al. 2023) has become popular for tasks ranging from dynamic scene reconstruction (Zhang et al. 2025; Liu et al. 2024) to super-resolution (Peng et al. 2025). GaussianImage (Zhang et al. 2024) first applied GS to 2D image representation and compression, significantly reducing memory and training time. However, GaussianImage lacks a densification mechanism to control the number of 2D Gaussians adaptivity based on image content, which affects the full utilization of the representation capabilities, limiting the fitting performance. Mirage (Waczyńska et al. 2024) represents 2D images using flat 3D Gaussian components, *i.e.* it initializes Gaussians on the XZ plane with the y-axis reduced to zero. It still employs Adaptive Density Control (ADC) (Kerbl et al. 2023) in 3D GS to prune and grow the Gaussians, easily causing uncontrolled growth of the number of Gaussians and leading to out-of-memory errors during training. LIG (Zhu

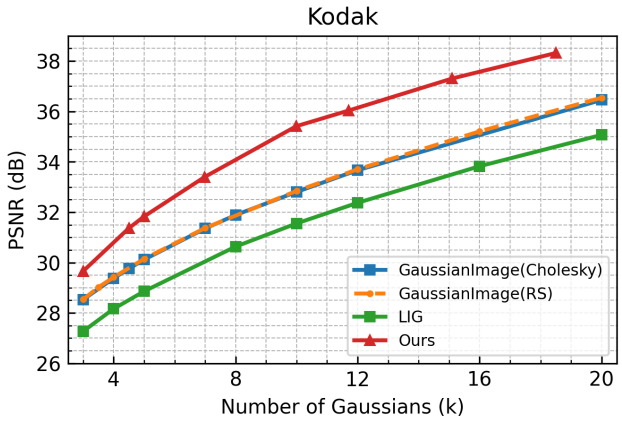

Figure 1: Image representation based on 2D GS methods with different numbers of Gaussians. Our GaussianImage++ exhibits significant performance gains.

*Corresponding author.

et al. 2025)focuses on fitting large images with numerous 2D Gaussians without compact compression of Gaussian attributes. For storage efficiency, various approaches are tailored for the compression of 3D GS, like LightGaussian (Fan et al. 2023). In contrast, 2D GS has received less attention. On the one hand, the redundant spherical harmonic coefficient in 3D GS is more suitable for compact Vector Quantization (VQ) compared with color attributes in 2D GS. On the other hand, 3D GS compression methods HAC (Chen et al. 2024) and ContextGS (Wang et al. 2024) are built upon the neural Gaussians, *i.e.* Scaffold (Lu et al. 2024), which introduce fundamental differences that prevent their direct application to our 2D GS framework. Our 2D GS utilizes explicitly stored attributes. In contrast, neural Gaussian approaches assign high-dimensional latent feature vectors to Gaussians, which an MLP then decodes into view-dependent color and opacity.This design is crucial for capturing complex 3D view-dependent appearance and for achieving compression by operating in the latent space. However, it is architecturally mismatched and introduces unnecessary overhead for our goal of efficiently representing a fixed 2D image.

We introduce GaussianImage++ to achieve efficient image representation and compression with 2D GS. Our explorations mainly include progressive training, distortion-driven densification, and context-aware Gaussian low-pass filters. Through these explorations, GaussianImage++ achieves superior performance compared to vanilla 2D GaussianImage on multiple benchmark datasets in image representation and compression while maintaining small disk overhead and fast rendering speeds. Furthermore, our densification and context-aware Gaussian low-pass filters are flexible and universal enhancement techniques for current 2D GS approaches. Our main contributions are threefold:

- We develop a universal boosting technique based on a progressive distortion-driven densification to control the density of 2D Gaussian primitives generated per image.
- We employ content-aware Gaussian low-pass filters to adaptively adjust the incidence and intensity of each Gaussian primitive, which echoes our densification and improves visual quality.
- Extensive experimental results demonstrate that our approach achieves significant improvements in various benchmark datasets for image representation and compression. Comprehensive ablation and analysis demonstrate the effectiveness and robustness of proposed components when applied to other 2D GS methods.

Related Work

Explicit Image Representation with 2D Gaussians

3D Gaussian Splatting represents 3D scenes by a set of explicit Gaussian ellipsoids with learnable attribute parameters. In 2D scenes, Mirage adapts flat 3D Gaussians within the XZ plane for 2D image editing and representation. GaussianImage pioneers 2D GS for image representation and compression by presenting a compact 2D GS representation and a novel accumulated blending-based rasterization. However, this pioneering work lacks a critical densification process necessary for adaptive Gaussian primitive allocation, and rate-distortion (RD) performance needs to be improved. A follow-up 2D GS work LIG focuses on fitting large images with numerous Gaussians but does not explore the compact compression of these multi-attribute Gaussians, leading to significant overhead on resource-constrained devices.

Neural Image Compression

Traditional image codecs, like JPEG (Wallace 1991), JPEG2000 (Taubman, Marcellin, and Rabbani 2002), BPG (Bellard 2014) and so on, feature high visual quality and excellent compression ratio, but complex handcrafted modules (transformation, quantization and entropy coding) lead to substantial decoding latency and limit real-time applications. Learning-based image compression (LIC) methods (Ballé, Laparra, and Simoncelli 2016; Ballé et al. 2018; He et al. 2022; Cheng et al. 2020) based on hierarchical variational autoencoders (VAEs) surpass traditional codecs in RD performance through learned nonlinear transforms and entropy models. However, cascaded convolutional blocks and autoregressive entropy coding introduce substantial computational overhead during decoding, limiting their practical deployment. While efforts toward real-time decoding like EVC (Wang et al. 2023) improve inference efficiency, they remain grounded in the VAEs/hyperprior framework. GaussianImage++ currently still lags behind these methods in RD performance but offers a solution diagram, at the same time, with significant advantage in decoding speed and memory efficiency.

Image-based INR methods (Sitzmann et al. 2020; Dupont et al. 2021, 2022; Ladune et al. 2023; Girish, Shrivastava, and Gupta 2023) provide a novel compression paradigm: An encoder approximates a spatial-to-color mapping via an MLP, quantizes and transmits the MLP weights as compressed codes. A decoder reconstructs the image by querying the MLP at each pixel coordinate. However, purely relying on implicit MLP architectures leads to prolonged optimization cycles and suboptimal RD performance. Subsequent hybrid approaches C3 (Kim et al. 2024), COOL-CHIC (Ladune et al. 2023), and work (Girish, Shrivastava, and Gupta 2023) integrate explicit multi-resolution hash-grid features to accelerate MLP convergence and enhance visual quality.Nevertheless, these hybrid models require instance-specific quantizers and autoregressive context entropy models for additional decoding grid latents, introducing substantial computational overhead and decoding time. Diverging from VAEs and INRs paradigms, we employ explicit 2D GS that achieves high visual fidelity, memory efficiency, and real-time decoding speeds.

Methodology

In this section, we present our GaussianImage++ framework as illustrated in Figure 2. We begin with the basics of GS and its variants. Subsequently, we detail our two critical enhancements: the distortion-driven densification (D^3) and content-aware Gaussian filters (CAF) with adaptive variance. Finally, we discuss our compression techniques.

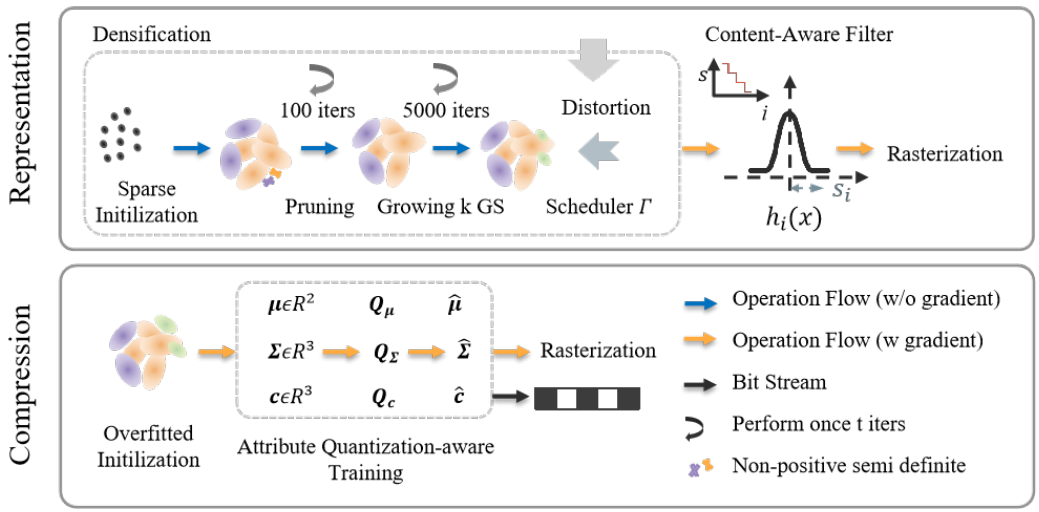

Figure 2: Our proposed GaussianImage++ framework. Our representation pipeline uses densification to initialize sparse Gaussians, growing them periodically to improve under-reconstructed areas. A content-aware filter is applied to all Gaussians before accumulated sum rasterization. For compression, we initialize with overfitted 2D Gaussians and employ quantization-aware training to encode attributes into compact bitstreams.

2D Gaussian Splatting. Motivated by 3D Gaussian Splatting (3DGS), GaussianImage represents 2D scenes using a set of 2D Gaussian ellipticals parameterized by position $\mu \in \mathbb{R}^2$, covariance $\mathbf{\Sigma} \in \mathbb{R}^{2\times2}$ and color $c \in \mathbb{R}^3$. Oriented to 2D image representation, GaussianImage eliminates view and project transformation and depth-based Gaussian sorting, and replaces spherical harmonics with RGB colors, α blending with accumulated summation. Consequently, the rendered image changes to

$$G_i(x) = \exp(-\frac{(\mathbf{x}-\mu_\mathbf{i})^T\Sigma^{-1}(\mathbf{x}-\mu_\mathbf{i})}{2}) \quad (1)$$

$$C = \sum_{i\in N} c_i G_i(x). \quad (2)$$

According to the properties of covariance matrices $\mathbf{\Sigma}$, different parameterization variants exist for 2D GS: 1) Parameterize a rotation matrix $\mathbf{R}$ and scaling matrix $\mathbf{S}$ like 3D GS, where $\mathbf{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T$. 2) Parameterize a lower triangular matrix $\mathbf{L}$ like GaussianImage, leveraging Cholesky factorization $\mathbf{\Sigma} = \mathbf{LL}^T$. 3) Directly parameterize $\mathbf{\Sigma}$ without decomposition like LIG. As stated in LIG, while the positive semi-definite nature of $\mathbf{\Sigma}$ helps Gaussians retain their physical meaning, it is not strictly necessary for image fitting. We also directly parameterize $\mathbf{\Sigma}$. In the experiment, we have proven that our two key components are effective regardless of these three parameterization methods.

Distortion Driven Densification

Adaptive Density Control (ADC) in 3D GS removes transparent Gaussians and densifies Gaussians with an average magnitude of view-space position gradients above a threshold. Directly applying ADC in 2D GS is unfeasible since changes accumulated from position gradients are are typically too low to trigger the densification mechanism effectively.Prior 2D GS methods have limitations: GaussianImage lacks progressive densification, leading to substantial under-reconstructed areas. Although LIG employs a hierarchical approach, which fits a low-scale image and a high-scale residual image, it lacks dynamic Gaussian growing and pruning within each level.

Motivated by these observations, we propose a progressive, distortion-driven densification mechanism, which is a straightforward and image-quality-oriented method with explicit regulation of Gaussians. In summary, our densification mechanism comprises three stages: Sparse initialization, Gaussian growing, and Gaussian pruning. We limit the maximum number of Gaussians to M, allowing users to easily trade off rendering quality and memory overhead, especially on devices with limited memory.

Sparse Initialization. A good initialization is crucial for image quality. We employ a random initialization strategy with specific designs. Let N_t denote the number of Gaussians at iteration iteration t. In initialization ($N_0 = \frac{M}{2}$), we uniformly and randomly sample Gaussians positions within image coordinates $[0, H) \times [0, W)$, where H and W are image height and width. The covariances Σ are randomly and uniformly sampled within the normalized range (0, 1), with diagonal elements constrained by a lower bound of 0.5 to ensure positive semi-definiteness. Colors are initialized as zeros. Sparse initialization also accelerates early training due to the small N_0.

Gaussains Growing. As sparse Gaussians fail to represent high-frequency details, we progressively densify more Gaussians to under-reconstructed regions. A concurrent densification work (Bulò, Porzi, and Kontschieder 2024) in 3D GS re-distributes the per-pixel errors to each Gaussian and

uses per-Gaussian error to decide the clone-split of Gaussians. We directly decide the densification based on a per-pixel distortion $D(X, \hat{X})$ between raw image X and rendered image $\hat{X}$, and the current Gaussians allowance $M - N_t$. We periodically (interval 5000 iterations) perform our densification. In each densification step, we sample k new Gaussian primitives $\Psi := \{g_1, \ldots, g_k\}$ at pixels with top-k reconstruction distortion. We parameterize their position attributes as

$$\mu_\Psi = \xi\left(\mathrm{Top}_k\left(D(X, \hat{X})\right)\right), \tag{3}$$

color attributes as

$$\mathbf{c}_\Psi = X\left(\xi\left(\mathrm{Top}_k\left(D(X, \hat{X})\right)\right)\right). \tag{4}$$

The covariance attributes Σ_Ψ are parameterized with the same method as the initialization stage. Among k is a variable decided by a scheduler $\tau(t, N_t, M) = \frac{M - N_t}{2}$, ξ is an index function that returns the two-dimensional coordinates of pixels with top-k distortion. $D()$ is an arbitrary image quality distortion function. We use L1 loss for computational simplicity.

Gaussians Pruning. Directly optimizing covariance matirx $\mathbf{\Sigma}$ may violate the positive semidefinite in mathematics, which makes Equation (1) not work. Consequently, these invalid Gaussians make no contributions to rendering. We prune them to save memory overhead and allowance of Gaussians. As shown in Figure 2, we periodically (100 iterations) check the semi-positive quality of $\mathbf{\Sigma}$ and prune invalid ones for saving memory overhead and allowance of Gaussians.

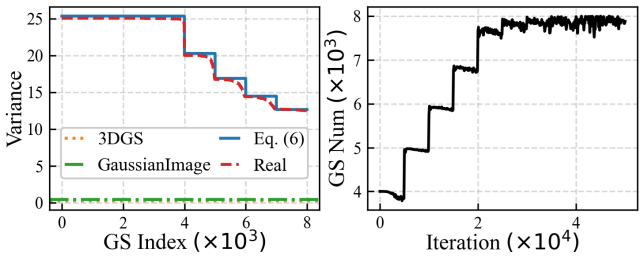

Figure 3: Left: the variance s_i of the i-th Gaussians primitives. Right: Number of Gaussians N_t at iteration t.

Content Aware Gaussian Filters

3D GS commonly uses elliptical weighted average (EWA) splatting (Zwicker et al. 2002) for anti-aliasing before tile-based rasterization. This addresses sampling rate variations caused by perspective projection from 3D Gaussian kernels to 2D screen-space splats. Such anti-aliasing is achieved by applying a resampling filter to the projected 2D Gaussian splats. Consequently, the original footprint function $G_i(x)$ is modified to

$$G_i'(x) = e^{-\frac{1}{2}(x-\mu_i)^T(\Sigma_i + sI)^{-1}(x-\mu_i)}. \tag{5}$$

In essence, this applies a zero-mean Gaussian low-pass filterr $h(x)$ to the original Gaussian reconstruction kernel $G_i(x)$, *i.e.*, $G_i'(x) = (G_i \otimes h)(x)$, where s is a scalar representing the variance of $h(x)$ and I is a unit matrix. 3D GS and GaussianImage use a constant s, 0.3 and 0.5, respectively. In GaussianImage++, we propose a content-aware filter $h(x)$ governed by an adaptive variance vector $\mathbf{s} \in \mathbf{R}^{N_t}$. Each element $s_i \in \mathbb{R}_+$ $(i = 1, \ldots, N_t)$ controls the filter strength for the i-th Gaussians and is derived by

$$s_i = \begin{cases} \frac{HW}{\alpha} N_t, & i > N_{t-1} \\ s_{i-1}, & i \leq N_{t-1} \end{cases} \tag{6}$$

where α is a scaling factor. Figure 3 plots theoretical values derived from Equation (6) and real values of $\mathbf{s}$ observed during training. Real $\mathbf{s}$ fluctuates slightly due to pruning. Importantly, $\mathbf{s}$ does not consume additional storage, as we store the filtered covariance $\mathbf{\Sigma} + \mathbf{s}I$.

Our CAF adaptively adjusts the sampling rate of the pixel grid by scaling the intersection area of Gaussians and pixels, effectively reducing holes and artifacts in the rendered image. As shown in Figure 3 (left), for the early GS, we impose a strong filter $h(x)$ with large s_i. The large s_i enlarges the original 2D Gaussian reconstruction kernel, which significantly magnifies their coverage area and effectively mitigates undersampling and reduces large "holes" in the rendered image. This strategy enables early-stage rendering to produce a recognizable, coarse image, crucial for guiding optimization. As densification begins, we gradually weaken the filter strength, *i.e.* reducing s_i variance for newly added Gaussian primitives. This prevents the filter from dominating the reconstruction kernel, allowing new GS to focus on finer details.

Moreover, the adaptive filtering mechanism aligns with our progressive densification strategy to assist in Gaussian optimization for distinct image content. As shown in Figure 4, at early training stages ($t = 500$ and $t = 1000$), both GaussianImage and our method without filters (first and second rows) inevitably exhibit significant artifacts and large holes due to $N_t \ll HW$ (the regions are marked with red squares and zoomed into the next patches), since some pixels without any GS intersected. In contrast, the third row in Figure 4, GaussianImage++ with CAF, renders a smooth and recognizable image plane even at $t = 500$ and $N_t \approx \frac{M}{2}$. This early coarse rendering capability ultimately contributes to better final visual quality, as later Gaussians can focus on refining details more effectively.

Compression Framework

As shown in Figure 2, we first perform the image representation pipeline with certain steps, enabling the Gaussians with overfitted initial attributes for quantization. Secondly, we conduct attribute quantization-aware training to fine-tune Gaussians with more compact attributes. Given a collection of 2D Gaussian primitives, we apply distinct bit-depth LSQ+ (Bhalgat et al. 2020) quantizers to different attributes. LSQ+ is an effective low-bit quantization technique through learnable offsets β and scale s. For a vector $\mathbf{v}$, it is quantized and de-quantized with gradient flow by

$$\overline{\mathbf{v}} = \lfloor \mathrm{clip}(\frac{\mathbf{v} - \beta}{s}, 0, 2^b - 1) \rfloor, \quad \hat{\mathbf{v}} = \overline{\mathbf{v}} * s + \beta. \tag{7}$$

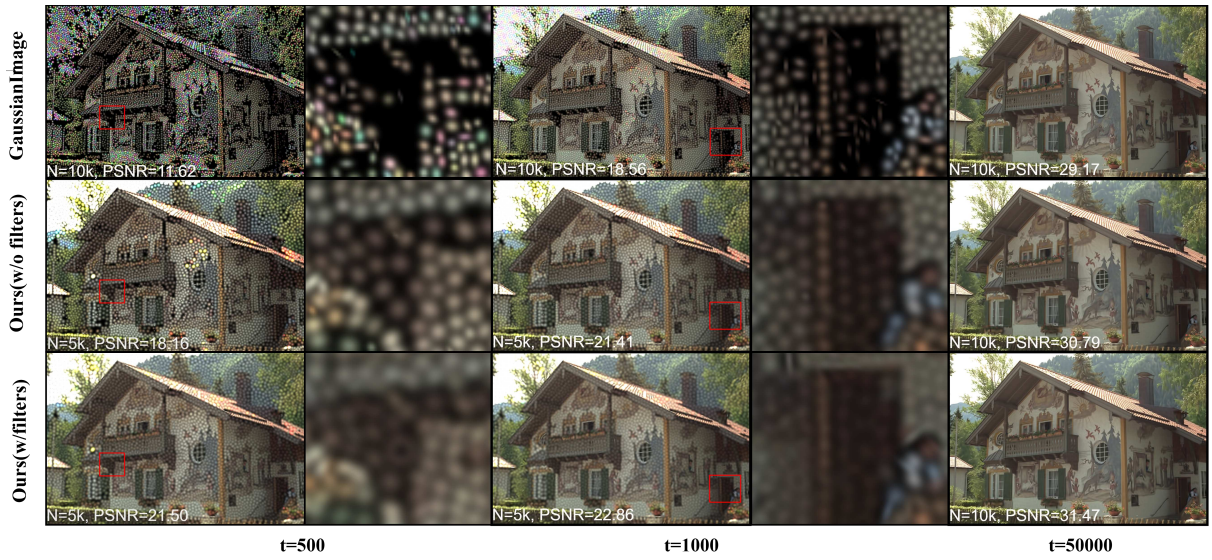

Figure 4: Progressive training of GaussianImage++ ($M = 10k$, $T = 5 \times 10^4$). Before densification (t=500), GaussianImage++ with CAF enables reducing the holes and artifacts and represent a coarse structure with sparse Gaussians. As densification steps (t=1000 or t=50000), the final rendered image shows enhanced visual quality.

Specifically, since the sensitivity of geometry attributes (μ and $\mathbf{\Sigma}$), we adopt 12 and 10-bit quantization precision for μ and Σ. For color attributes, we adopt 6-bit quantization precision. Such a bit-depth configuration is to balance bitrate and image quality. More comparative analysis and ablation studies are detailed in the supplementary materials.

Experiments

Experimental Setup

Datasets. We evaluate image representation and compression on two popular datasets: Kodak (Kodak 1993) (24 images, 768×512 resolution) and the DIV2K HR validation set (Agustsson and Timofte 2017) (100 images, 2K resolution).

Evaluation Metrics. We use PSNR and MS-SSIM to quantify visual fidelity, comparing GaussianImage++ with GS-based state-of-the-art and INR-based counterparts. We also report parameter size, training time and rendering speed in the image representation task. For image compression, we use rate-distortion performance and encoding/decoding time.

Implementation Details. GaussianImage++ is developed on top of GaussianImage with 8 trainable parameters as noted in Equation (2), among covariance Σ without decomposition unless otherwise stated. We optimize these parameters for 50,000 iterations using the Adam optimizer and L2 loss between ground truth and rendered images. Learning rates for all attributes are initialized at 0.18, halving to 0.5 after 20,000 iterations. In compression experiments, we perform warm-up training (6000 iterations) before the attribute quantization-aware fine-tuning. Quantizer learning rates are 0.001, decaying by half after 20,000 iterations. Experiments are conducted using NVIDIA A30 GPUs and PyTorch.

Image Representation

Comparison Results. We compare our method with GS-based methods (3D GS, GaussianImage, LIG and Mirage) and INR-based methods like Siren (Sitzmann et al. 2020). Table 1 presents quantitative results including PSNR, MS-SSIM, Parameters Number, GPU Memory, Training and Rendering time. For fair comparison, all GS-based methods are set to the same maximum numbers of GS. It is clear that compared with GS-based methods, INRs like Siren suffer from large training memory and parameter size, long encoding time and lower FPS. Compared with 2D GS methods, GaussianImage++ significantly enhances visual quality only at the expense of slightly longer training time, since periodic GS densification operation. Fortunately, our rendering speed is not affected by these operations and even exceeds the GaussianImage. Furthermore, as shown in Figure 1, our method consistently outperforms previous 2D GS approaches across all the gaussian number range.

Image Compression

RD Performance. Figure 5 presents the RD curves across Kodak and DIV2K datasets.Our codec outperforms INR-based COIN (Dupont et al. 2021) regardless of PSNR and MS-SSIM across all datasets. Compared to the GaussianImage baseline, GaussianImage++ surpasses it at the majority of bit-rate points across datasets and metrics, bringing significant improvement regarding bpp-PSNR. Furthermore, our method demonstratesnotable advantages over conventional JPEG codec at low bitrates (0.1-0.7bpp in Kodak and 0.1-0.5bpp in DIV2K). In high bitrate points, GaussianImage++ shows comparatively inferior performance. For learnable image codecs, all codecs based on 2D GS still have obvious performance gaps.

Computational Efficiency. Table 2 reports the computa-

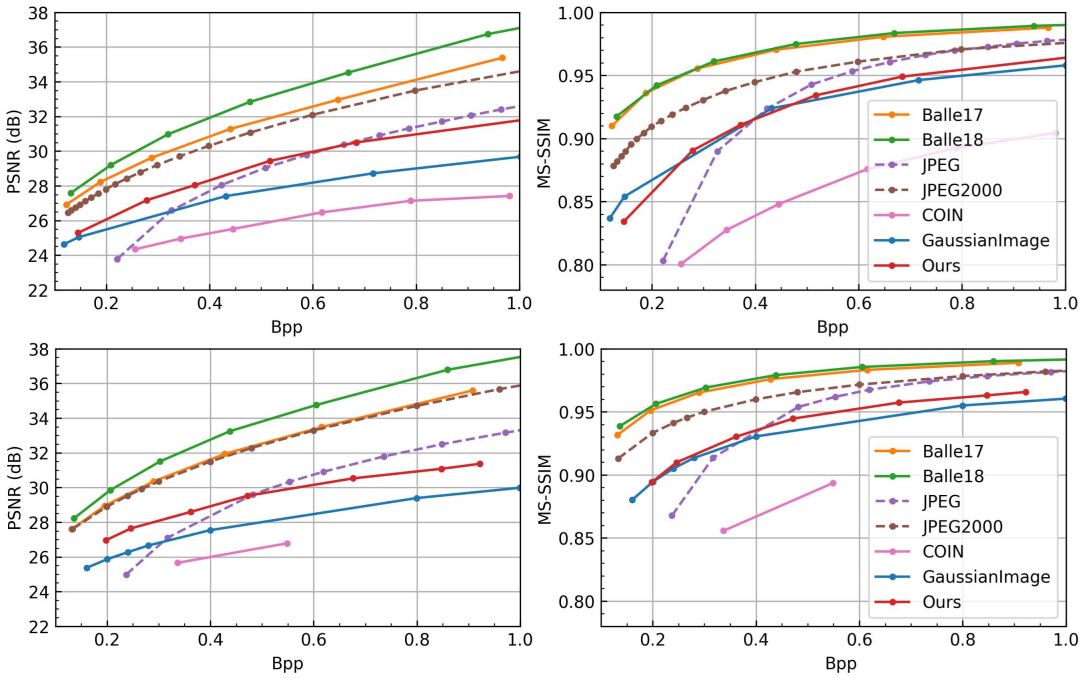

Figure 5: Rate-distortion curves of GaussianImage++ and different baselines on Kodak (top) and DIV2K (bottom) datasets.

tional efficiency of various codecs on Kodak and DIV2K datasets at low/high bitrate. Unlike the image representation, the integration of attribute quantization-aware training prolongs the encoding times of GaussianImage and GaussianImage++. The conventional and learnable codec still has faster encoding speeds, which remains a singnifcant rooms for GS-based methods. GS-based methods display the most impressive decoding speed, especially for GaussianImage++, maintaining real-time decoding superiority over conventional codec JPEG and learning-based codecs. See supplementary materials for more quality/efficiency comparisons with other state-of-the-art codecs.

Ablation Study

Effect of Different Components. To highlight the contributions of the key components distortion-driven densification (D^3) and content aware Gaussian filters (CAF), we conduct comprehensive ablation studies. These components are selectively integrated into existing 2D GS methods and our approach, all sharing identical training settings for fair comparison. As illustrated in Figure 6, **GS** and **GS (RS)** refer to GaussianImage methods using Cholesky and RS factorization for 2D covariance decomposition, respectively. **LIG** and all variants of **Our** directly optimize learnable covariance without decomposition. It is obviously that our two components consistently enhance representation performance regardless of the specific optimization strategy. In particular, the D^3 component yields the most significant quality enhancement - up to approximate 2dB PSNR gains over the GS with the same number of Gaussians. And alone D^3 also enhances baseline performance, especially for the case of fewer Gaussians. Combining D^3 with CAF, our

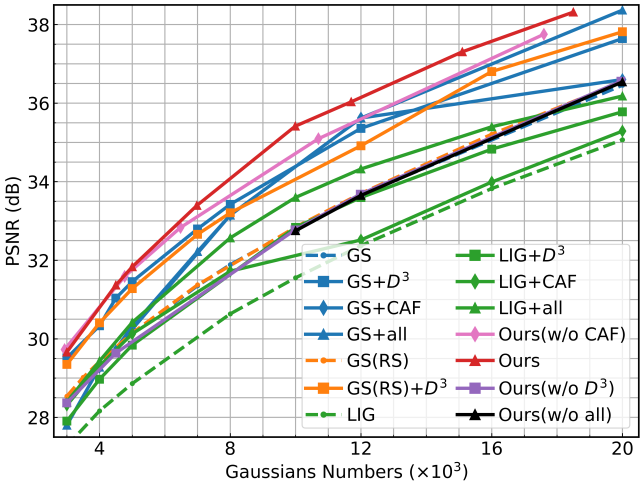

Figure 6: Ablation of components : distortion-driven densification (D^3) and content-aware Gaussian filters (CAF) across 2D GS-based image representation methods on Kodak.

GaussianImage++ attains further enhancements - up to 3dB PSNR gains over GaussianImage and 4dB PSNR gains over LIG.

Effect of Different Attributes Quantizers. Table 3 investigates the effect of different attribute quantization strategies on image compression. We alternatively replace original quantizers (LSQ+) of position/color attributes with Residual Vector Quantization (RVQ) and FP16 (the quantization in GaussianImagae). In this context, "FP16/LSQ+" denotes that positions use FP16 quantization while colors

Methods	Kodak PSNR	MS-SSIM	Params(M)	GPU MEM(MiB)	Training/s	Render/fps	DIV2K PSNR	MS-SSIM	Params	GPU MEM	Training	Render
INR-based												
Siren	26.50	0.875	3.74	2044	889.45	977	29.00	0.886	0.42	23913	4957	–
GS-based	Number of GS (5k)						Number of GS (10k)					
3D GS	27.02	0.942	0.30	922	331.39	988	24.60	0.897	0.59	2652	1287.44	458
MiraGe†	29.27	0.958	0.10	1242	247.93	1695	27.32	0.921	0.25	2802	2146	584
GaussianImage	29.85	0.962	0.04	812	93.49	2305	26.54	0.910	0.08	1042	275.49	825
LIG	28.25	0.946	0.04	832	118.40	1394	24.66	0.848	0.08	964	288.57	834
Ours	31.83	0.960	0.04	856	116.05	2365	28.14	0.914	0.08	958	323.74	1011
	Number of GS (10k)						Number of GS (50k)					
3D GS	29.99	0.968	0.59	918	337.67	947	29.58	0.965	2.95	3298	1422.47	351
MiraGe†	30.41	0.961	0.21	1234	387.71	496	33.23	0.981	1.16	2268	2153.88	419
GaussianImage	32.48	0.982	0.08	814	115.89	2009	31.45	0.977	0.40	1062	192.61	662
LIG	31.00	0.975	0.08	832	125.37	1331	29.71	0.965	0.40	1085	294.32	620
Ours	35.41	0.983	0.08	876	118.76	2216	33.75	0.978	0.38	962	356.71	765

Table 1: Quantitative evaluation across various baselines on Kodak and DIV2K datasets. 5k represents the number of GS primitives. Mirage† represents that we early stop ADC in Mirage when the current numbers of Gaussians surpass the set maximum number M.

	Kodak Bpp	PSNR	MS-SSIM	Encode/s	Decode/FPS	DIV2K Bpp	PSNR	MS-SSIM	Encode/s	Decode/FPS
JPEG	0.22/1.03	23.8/32.8	0.803/0.979	0.012/0.014	377/148	0.24/0.85	25.0/32.5	0.867/0.978	0.011/0.014	188/144
JPEG2K	0.13/1.19	26.6/35.7	0.882/0.981	0.582/0.419	2/3	0.27/0.96	29.9/25.7	0.945/0.982	0.533/0.554	4/5
Ballé17	0.12/0.97	26.9/35.4	0.910/0.988	0.029/0.049	63/32	0.29/0.91	30.3/35.6	0.965/0.989	0.216/0.908	6/6
Ballé18	0.13/0.94	27.5/36.7	0.917/0.989	0.034/0.051	44/28	0.30/0.86	31.6/38.8	0.913/0.965	0.237/0.385	7/7
COIN	0.17/0.98	24.9/27.4	0.827/0.904	457/726	769/344	0.27/1.33	26.6/29.0	0.856/0.886	4487/4957	11/22
GaussianImage	0.15/1.00	25.0/29.7	0.854/0.958	293/376	1827/1822	0.28/1.00	26.7/29.3	0.914/0.960	376/396	857/723
Ours	0.15/1.08	25.3/31.1	0.834/0.961	338/347	1839/1666	0.25/0.92	27.6/31.4	0.910/0.966	488/576	440/748

Table 2: Computational efficiency evaluation of various image codecs on two Datasets at low and high Bpp. Traditional codecs are run on a 12th Gen Intel(R) Core(TM) i5-12600K 3.70 GHz processor. Other codecs are operated on an NVIDIA A30 GPU.

Variants	BD-PSNR(dB)↑	BD-Rate(%)↓	BD-MS-SSIM↑	BD-Rate
LSQ+/LSQ+	0	0	0	0
FP16/LSQ+	-0.761	25.11	-0.009	17.77
FP16/RVQ	-2.471	138.88	-0.036	89.75
LSQ+/RVQ	-2.491	147.24	-0.030	68.29

Table 3: Ablation of quantization strategies. The first is our final strategy. The left and right of "/" respectively indicate the quantization strategies of position/color attributes.

use LSQ+, and so forth. The results show significant performance degradation when substituting our LSQ+ position/color quantization modules with other approaches. For the first two rows, learnable LSQ+ quantization of position proves more bit-efficient and slightly improves image quality, since quantization-aware training can supervise the Gaussians to adjust their attributes to adapt to the quantization errors. From the last two rows, RVQ for color leads to obvious performance drops due to the limited representation capability of its codebook. Moreover, we ablate the bit-depth configuration for our quantization strategies in supplementary material.

Conclusion

In this work, we introduce GaussianImage++, a boosted 2D Gaussian Splatting paradigm for image representation and compression. It achieves significant improvements in visual quality and rate-distortion performance while maintaining real-time rendering speeds. These advancements are driven by two novel components: 1) A distortion-driven densification progressively densifies Gaussians to the image plane, which structurally reduces the number of Gaussian primitives and achieves a better trade-off between the Gaussian compactivity and rendering quality. 2) The content-aware Gaussian filters, in conjunction with our densification, assist in the 2D GS progressive fitting of distinct details. These two components address limitations in existing 2D GS methods. Furthermore, they are generalizable techniques, which can be friendly integrated with other 2D GS methods for boosted performance.

Limitations

GaussianImage++ still currently lags behind state-of-the-art neural image codecs, especially in high bitrates. Bridging this gap requires exploring more advanced attribute encoding schemes and entropy models. Furthermore, current encoding time is far from realtime, requiring further optimizing the training and quantization processes.

Acknowledgments

This work was supported by Wuxi Research Institute of Applied Technologies, Tsinghua University under Grant

20242001120.

References

Agustsson, E.; and Timofte, R. 2017. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 126–135.

Ballé, J.; Laparra, V.; and Simoncelli, E. P. 2016. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*.

Ballé, J.; Minnen, D.; Singh, S.; Hwang, S. J.; and Johnston, N. 2018. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*.

Bellard, F. 2014. BPG image format. https://bellard.org/bpg/.

Bhalgat, Y.; Lee, J.; Nagel, M.; Blankevoort, T.; and Kwak, N. 2020. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 696–697.

Bulò, S. R.; Porzi, L.; and Kontschieder, P. 2024. Revising densification in gaussian splatting. *arXiv preprint arXiv:2404.06109*.

Chen, Y.; Wu, Q.; Lin, W.; Harandi, M.; and Cai, J. 2024. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, 422–438. Springer.

Cheng, Z.; Sun, H.; Takeuchi, M.; and Katto, J. 2020. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7939–7948.

Dupont, E.; Goliński, A.; Alizadeh, M.; Teh, Y. W.; and Doucet, A. 2021. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*.

Dupont, E.; Loya, H.; Alizadeh, M.; Goliński, A.; Teh, Y. W.; and Doucet, A. 2022. Coin++: Neural compression across modalities. *arXiv preprint arXiv:2201.12904*.

Fan, Z.; Wang, K.; Wen, K.; Zhu, Z.; Xu, D.; and Wang, Z. 2023. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS.

Girish, S.; Shrivastava, A.; and Gupta, K. 2023. Shacira: Scalable hash-grid compression for implicit neural representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 17513–17524.

He, D.; Yang, Z.; Peng, W.; Ma, R.; Qin, H.; and Wang, Y. 2022. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5718–5727.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4): 139–1.

Kim, H.; Bauer, M.; Theis, L.; Schwarz, J. R.; and Dupont, E. 2024. C3: High-performance and low-complexity neural compression from a single image or video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9347–9358.

Kodak, E. 1993. Kodak lossless true color image suite (PhotoCD PCD0992). *URL http://r0k. us/graphics/kodak*, 6.

Ladune, T.; Philippe, P.; Henry, F.; Clare, G.; and Leguay, T. 2023. Cool-chic: Coordinate-based low complexity hierarchical image codec. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 13515–13522.

Liu, Z.; Hu, Y.; Zhang, X.; Shao, J.; Lin, Z.; and Zhang, J. 2024. Dynamics-aware gaussian splatting streaming towards fast on-the-fly training for 4d reconstruction. *Training*, 101: 102.

Lu, T.; Yu, M.; Xu, L.; Xiangli, Y.; Wang, L.; Lin, D.; and Dai, B. 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20654–20664.

Peng, L.; Wu, A.; Li, W.; Xia, P.; Dai, X.; Zhang, X.; Di, X.; Sun, H.; Pei, R.; Wang, Y.; et al. 2025. Pixel to gaussian: Ultra-fast continuous super-resolution with 2d gaussian modeling. *arXiv preprint arXiv:2503.06617*.

Sitzmann, V.; Martel, J.; Bergman, A.; Lindell, D.; and Wetzstein, G. 2020. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33: 7462–7473.

Taubman, D. S.; Marcellin, M. W.; and Rabbani, M. 2002. JPEG2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2): 286–287.

Waczyńska, J.; Szczepanik, T.; Borycki, P.; Tadeja, S.; Bohné, T.; and Spurek, P. 2024. Mirage: Editable 2d images using gaussian splatting. *arXiv preprint arXiv:2410.01521*.

Wallace, G. K. 1991. The JPEG still picture compression standard. *Communications of the ACM*, 34(4): 30–44.

Wang, G.-H.; Li, J.; Li, B.; and Lu, Y. 2023. Evc: Towards real-time neural image compression with mask decay. *arXiv preprint arXiv:2302.05071*.

Wang, Y.; Li, Z.; Guo, L.; Yang, W.; Kot, A. C.; and Wen, B. 2024. Contextgs: Compact 3d gaussian splatting with anchor level context model. *arXiv preprint arXiv:2405.20721*.

Zhang, X.; Ge, X.; Xu, T.; He, D.; Wang, Y.; Qin, H.; Lu, G.; Geng, J.; and Zhang, J. 2024. Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In *European Conference on Computer Vision*, 327–345. Springer.

Zhang, X.; Liu, Z.; Zhang, Y.; Ge, X.; He, D.; Xu, T.; Wang, Y.; Lin, Z.; Yan, S.; and Zhang, J. 2025. Mega: Memory-efficient 4d gaussian splatting for dynamic scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 27828–27838.

Zhu, L.; Lin, G.; Chen, J.; Zhang, X.; Jin, Z.; Wang, Z.; and Yu, L. 2025. Large Images are Gaussians: High-Quality Large Image Representation with Levels of 2D Gaussian Splatting. *arXiv preprint arXiv:2502.09039*.

Zwicker, M.; Pfister, H.; Van Baar, J.; and Gross, M. 2002. EWA splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3): 223–238.