CALISPLAT: SPARSE GAUSSIAN FITTING AND CALIBRATED RATE ALLOCATION FOR IMAGE COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

2D Gaussian representations provide an explicit and lightweight alternative for image representation, enabling fast deterministic decoding without a heavy neural synthesis network. However, existing per-image Gaussian codecs still require costly image-specific optimization to determine how limited Gaussian capacity should be distributed across heterogeneous regions. We introduce CaliSplat, an optimization-efficient Gaussian image codec that selectively explores candidate representations and globally allocates a constrained Gaussian budget. We exploit the closed-form reconstruction error of the single-Gaussian case as an inexpensive measure of local difficulty, fitting higher-capacity candidates only in regions where additional Gaussians are likely to be useful. To reduce selection errors caused by mismatched convergence rates across candidate sizes, a small set of longer-horizon fits is used to calibrate short-horizon distortion estimates before global budget-aware selection. The selected representation is then compressed into a bitstream. Across standard and large image datasets, CaliSplat achieves competitive or superior rate–distortion performance while reducing per-image fitting time by $3.0\times$–$15.7\times$ compared with representative Gaussian codecs. It further maintains high-throughput deterministic decoding at 175.9–478.2 MP/s. These results demonstrate that content-adaptive Gaussian compression can substantially reduce per-image optimization cost while preserving strong reconstruction quality and efficient explicit decoding.

1 INTRODUCTION

Image compression aims to represent visual signals compactly while preserving reconstruction quality and enabling efficient decoding. Modern learned codecs achieve strong rate–distortion performance through expressive transforms and entropy models (Ballé et al., 2016; 2018; Minnen et al., 2018; Cheng et al., 2020), while per-image approaches optimize compact representations for individual inputs and can employ substantially lighter decoders (Dupont et al., 2021; 2022; Ladune et al., 2023; Kim et al., 2024). More recently, 2D Gaussian Splatting has emerged as an attractive explicit alternative for per-image representation and compression (Zhang et al., 2024a; Zhu et al., 2025; Zeng et al., 2025; Pan et al., 2026). By reconstructing images from explicit Gaussian primitives through highly parallel rasterization, these methods offer lightweight and high-throughput decoding.

Despite these advantages, efficient Gaussian compression remains challenging because natural images require highly non-uniform representation capacity. GaussianImage establishes 2D Gaussian primitives for image compression (Zhang et al., 2024a), while LIG introduces hierarchical Gaussian representations (Zhu et al., 2025), Instant GaussianImage accelerates fitting through learned initialization and adapts the overall Gaussian count (Zeng et al., 2025), and SGI improves structure and optimization through seed-based generation and coarse-to-fine fitting (Pan et al., 2026). However, these methods do not explicitly address how a limited Gaussian budget should be distributed across spatially heterogeneous regions. Smooth areas may require only a few Gaussians, whereas edges and textures benefit from substantially larger local capacity. Determining such fine-grained allocation naively requires fitting many candidate representations that are ultimately discarded. This creates a central trade-off between content-adaptive capacity allocation and per-image optimization efficiency.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

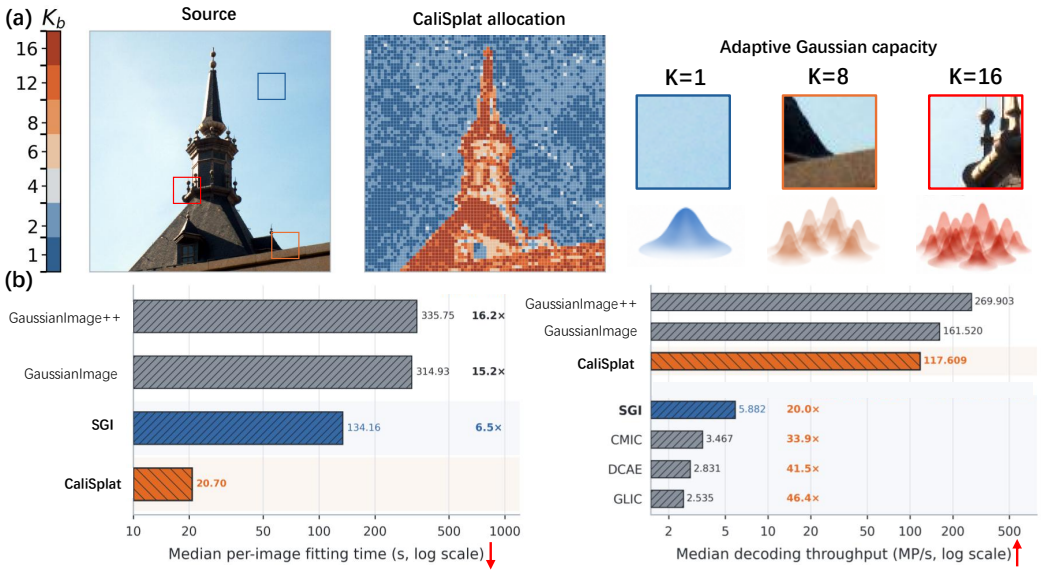


Figure 1: **(a)** Content-adaptive Gaussian allocation assigns larger Gaussian capacity to more complex regions. **(b)** CaliSplat achieves fast fitting while maintaining high decoding throughput.

We introduce CaliSplat, a per-image Gaussian codec that addresses this trade-off through sparse candidate fitting and global budget-aware allocation. Our key observation is that the single-Gaussian case admits a closed-form solution: under squared error, its distortion $D_b(1)$ can be computed analytically and serves as an inexpensive measure of local representation difficulty. We therefore fit higher-capacity candidates only in regions likely to benefit from additional Gaussians, substantially reducing unnecessary optimization. Since short-horizon distortions may mis-rank candidates with different convergence rates, a small set of longer-horizon fits is used to calibrate their scores before global selection. The resulting scores guide image-level allocation of a constrained Gaussian budget, concentrating capacity in structurally complex regions as illustrated in Fig. 1(a). Finally, the selected Gaussians are compressed into a bitstream using deterministic geometric templates and structured template-relative parameter coding.

Experiments demonstrate both the component effectiveness and end-to-end efficiency of CaliSplat. The analytic difficulty gate reduces the K-weighted candidate-fitting workload to 65% of exhaustive search with essentially unchanged reconstruction quality, while budget-aware allocation improves PSNR by 1.20 dB over uniform allocation and by 2.92 dB over a matched spatially shuffled baseline. On the main datasets, CaliSplat achieves competitive rate–distortion performance on standard images and superior PSNR on large-image datasets, while reducing per-image fitting time by $3.0\times$–$15.7\times$ compared with representative Gaussian codecs. As shown in Fig. 1(b), it additionally retains high-throughput deterministic decoding at 175.9–478.2 MP/s.

Our contributions are summarized as follows:

- We introduce an analytic difficulty-guided sparse fitting strategy that leverages the closed-form distortion of a single Gaussian to identify regions that require higher representation capacity, substantially reducing redundant per-image optimization.
- We propose a calibrated, budget-aware Gaussian allocation framework that calibrates inexpensive short-horizon candidate evaluations with a small set of longer-horizon fits, enabling globally optimized capacity allocation under a constrained image-level Gaussian budget.
- Extensive experiments demonstrate that CaliSplat achieves competitive or superior rate–distortion performance while substantially improving per-image optimization efficiency and maintaining high decoding throughput, with particularly strong performance in high-rate and high-resolution regimes.

2 Related Work

2.1 Image Compression

Learned image compression typically maps images to quantized latents and reconstructs them with a learned synthesis transform. Early work established end-to-end rate–distortion optimization, hyperpriors, autoregressive context modeling, and expressive likelihood models (Minnen et al., 2018; Cheng et al., 2020). Later methods improve coding efficiency and context modeling through space–channel grouping and hybrid Transformer–CNN architectures (He et al., 2022; Liu et al., 2023). Recent systems further explore linear attention, dictionary-based entropy models, content-adaptive state-space models, and hierarchical context modeling (Feng et al., 2025; Lu et al., 2025a; Qin et al., 2025; Li et al., 2025), while practical and adaptive codec design remains an active direction (Tatwawadi et al., 2026; Chen et al., 2026b; Ma et al., 2026).

Per-image codecs instead optimize a representation for each test image. COIN and subsequent INR-based methods encode image-specific representations (Dupont et al., 2021; Strümpler et al., 2022; Ladune et al., 2023). C3 further demonstrates strong rate–distortion performance with a low-complexity overfitted decoder (Kim et al., 2024), while recent work studies perceptual objectives and online image-specific optimization (Ballé et al., 2025; Kuang et al., 2025). GaussianImage introduced 2D Gaussian splatting for image representation and compression (Zhang et al., 2024a). LIG explore content-adaptive or multilevel Gaussian representations (Zhu et al., 2025), while Instant GaussianImage reduces fitting cost and adapts the primitive count to image content (Zeng et al., 2025).

2.2 3D Gaussian Splatting Compression

3D Gaussian Splatting represents radiance fields using explicit anisotropic primitives and differentiable rasterization (Kerbl et al., 2023). Because high-quality scenes often contain a large number of Gaussians with geometry, opacity, and appearance attributes, compression methods have explored pruning, masking, vector quantization, and compact parameter representations (Niedermayr et al., 2024; Fan et al., 2024). Subsequent work further exploits spatial dependencies through structured representations and contextual entropy models (Lu et al., 2024; Wang et al., 2024b), while MesonGS studies efficient post-training coding (Xie et al., 2024). Recent methods extend these directions with feed-forward compression (Chen et al., 2025), feature-plane coding with standard video codecs (Lee et al., 2025), and spatial-distribution optimization (Yang et al., 2026). These studies highlight that Gaussian compression benefits not only from compact attribute coding, but also from structured representations and adaptive allocation of limited Gaussian capacity.

3 Proposed Method: CaliSplat

We introduce **CaliSplat**, a per-image compression framework based on explicit anisotropic Gaussians. Given an RGB image I, the encoder produces a bitstream S, from which the decoder recovers an explicit Gaussian representation $\widehat{\Theta}$ and deterministically reconstructs the image:

$$\widehat{\Theta} = \mathrm{Dec}_{\Theta}(S), \qquad \widehat{I} = \mathcal{R}(\widehat{\Theta}). \tag{1}$$

The main challenge is to determine where Gaussian should be spent without fully optimizing every possible representation. As shown in Fig. 2, we first construct sparse candidates only where additional Gaussian is likely to be useful (Sec. 3.1). Since short optimization can favor candidate sizes that simply converge faster, we use a small number of longer-horizon reference fits to calibrate their scores (Sec. 3.2). These calibrated scores guide the global Gaussian-budget allocation, after which the selected parameters are compactly quantized and entropy-coded into the bitstream S (Sec. 3.3). The decoding only recovers the explicit Gaussian parameters from S and rasterizes them into the reconstructed image (Sec. 3.4). Alg. 1 and 2 summarize the complete encoding and decoding procedures, respectively.

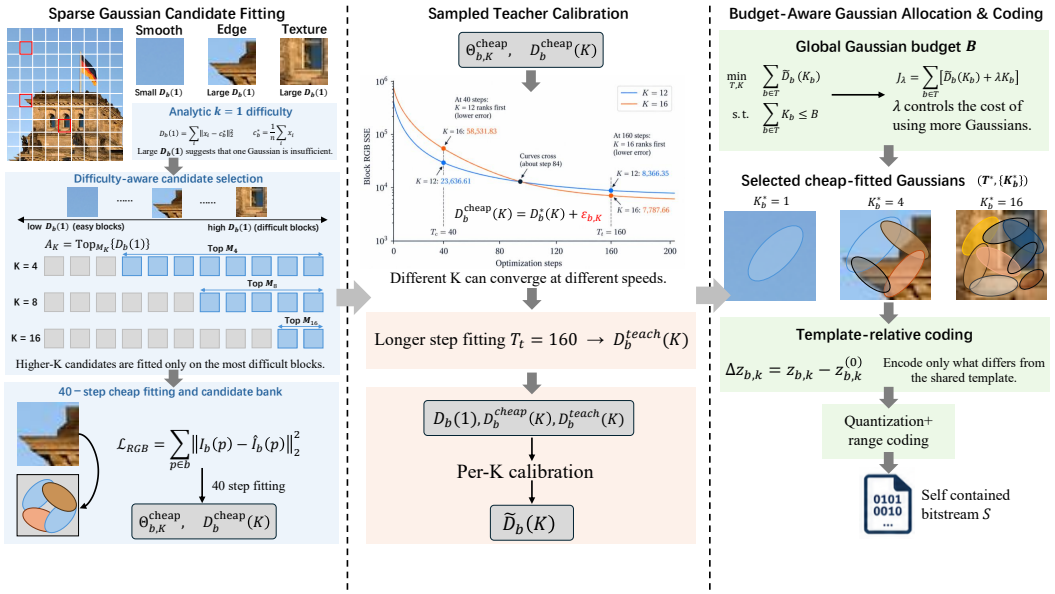


Figure 2: The overview of the CaliSplat encoding pipeline. Sparse candidate fitting reduces unnecessary optimization, sampled teacher calibration corrects short-horizon scores, and budget-aware allocation selects and codes the final Gaussian representation.

3.1 Sparse Gaussian Candidate Fitting

Local Gaussian representation. We partition the image into many blocks and represent each block with a variable number K of local anisotropic Gaussians. The smooth regions can often be represented with few Gaussians, whereas edges and textures may require more. A Gaussian in block b is parameterized as

$$\theta_{b,k} = (\boldsymbol{\mu}_{b,k}, \boldsymbol{s}_{b,k}, \phi_{b,k}, a_{b,k}, \boldsymbol{c}_{b,k}), \tag{2}$$

where $\boldsymbol{\mu}$, $\boldsymbol{s} = (s_1, s_2)$, ϕ, a, and $\boldsymbol{c}$ denote center, principal-axis scales, orientation, log-weight, and RGB color, respectively. We denote a K-Gaussian candidate by $\Theta_{b,K} = \{\theta_{b,k}\}_{k=1}^{K}$.

For a local coordinate $\mathbf{p}$, the Gaussian logit is

$$\ell_{b,k}(\mathbf{p}) = -\frac{1}{2}\left[\left(\frac{d_{1,b,k}(\mathbf{p})}{s_{1,b,k}}\right)^2 + \left(\frac{d_{2,b,k}(\mathbf{p})}{s_{2,b,k}}\right)^2\right] + a_{b,k}, \tag{3}$$

where d_1 and d_2 are coordinates along the rotated principal axes. The normalized Gaussian weights and block reconstruction are

$$\pi_{b,k}(\mathbf{p}) = \frac{\exp(\ell_{b,k}(\mathbf{p}))}{\sum_{j=1}^{K}\exp(\ell_{b,j}(\mathbf{p}))}, \qquad \widehat{I}_b(\mathbf{p}) = \sum_{k=1}^{K}\pi_{b,k}(\mathbf{p})\boldsymbol{c}_{b,k}. \tag{4}$$

Thus, each pixel is reconstructed as a local mixture of Gaussian colors without depth ordering or alpha compositing. Although a larger K provides more fitting capacity, optimizing every possible K for every block would make per-image encoding expensive. We therefore first ask a cheaper question: which blocks are likely to need more than the simplest representation?

Analytic difficulty estimation. We answer this question using the $K = 1$ case, which requires no iterative optimization. With a single Gaussian, the softmax weight is identically one, so the entire block reduces to one constant RGB color. Under squared error, the optimal color is simply the block mean:

$$\boldsymbol{c}_b^{\star} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \qquad D_b(1) = \sum_{i=1}^{n}\|\mathbf{x}_i - \boldsymbol{c}_b^{\star}\|_2^2. \tag{5}$$

This gives an intuitive difficulty signal at essentially zero fitting cost. If $D_b(1)$ is small, a single color already explains the block well; if it is large, the block may benefit from additional Gaussians. We therefore use $D_b(1)$ to decide where more expensive candidates are worth trying.

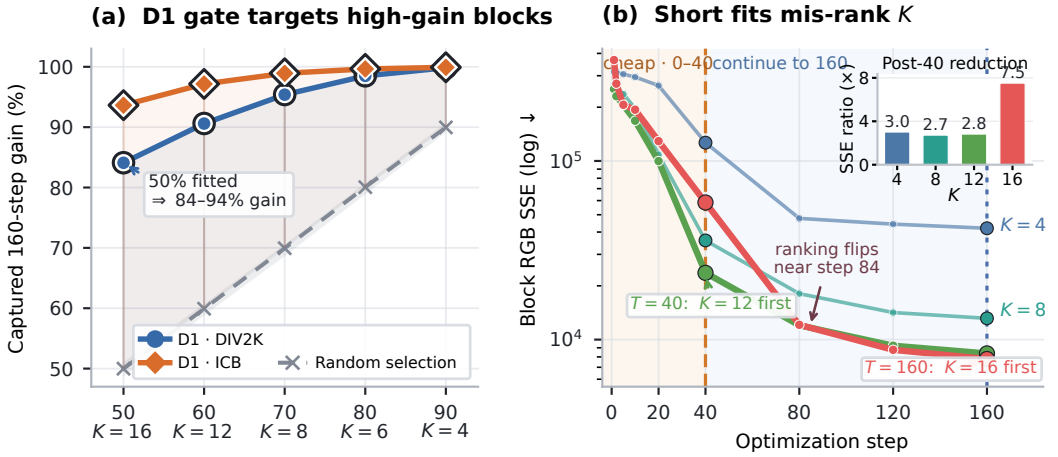


Figure 3: (a) The analytic D1 gate preserves most long-horizon fitting gain while evaluating only a subset of candidates. DIV2K and ICB denote the datasets used for these analyses. (b) A real order-reversal example shows that short fitting can mis-rank different K due to convergence mismatch, motivating sampled teacher calibration.

Difficulty-aware sparse fitting. For each candidate size K, we rank all image blocks by the analytic difficulty $D_b(1)$ and retain the top M_K blocks:

$$\mathcal{A}_K = \mathrm{Top}_{M_K} \{D_b(1)\}_b \,, \tag{6}$$

where M_K is determined by a predefined retention fraction. This gate M_K is a computational filter: it decides where optimization effort is spent, while the final Gaussian count is chosen later by the global allocator. As shown in Fig. 3(a), this simple ranking preserves most of the long-horizon fitting gain while evaluating only a subset of candidates.

For each admitted pair (b, K), $\theta_{b,k}$ is initialized from a deterministic template defined by the block size r_b and Gaussian count K, and colors are initialized in closed form as

$$C^{(0)} = W^{\dagger} X, \tag{7}$$

where W is the template responsibility matrix and X is the flattened RGB block. Starting from this initialization, we perform only $T_c = 40$ Adam steps. At each step, the current Gaussians are rasterized using Eq. 4, their RGB reconstruction error is computed against the source block, and gradients are backpropagated through the differentiable rasterizer. The resulting short-horizon candidate is

$$\Theta^{\mathrm{cheap}}_{b,K}, \qquad D^{\mathrm{cheap}}_b(K). \tag{8}$$

Candidates outside $\mathcal{A}_K$ are simply skipped, so they incur no fitting cost. For the admitted candidates, the short optimization produces a sparse bank of cheap-fitted Gaussian representations $\Theta^{\mathrm{cheap}}_{b,K}$ and their corresponding distortions $D^{\mathrm{cheap}}_b(K)$. Template construction, retention fractions, and optimization details are provided in Appendices A.2.1 and A.2.2.

3.2 Sampled Teacher Calibration

The sparse fitting stage in Sec. 3.1 efficiently produces multiple candidates for each block. However, different candidate sizes can converge at different speeds, so their errors after only $T_c = 40$ steps may not preserve their longer-horizon ranking. Fig. 3(b) shows a real order-reversal example: $K = 12$ has lower error at 40 steps, whereas $K = 16$ becomes better after longer optimization. We therefore use longer-horizon (teachers) fits as reference signals for correcting the short-fit scores.

Short-horizon optimization bias. We write

$$D^{\mathrm{cheap}}_b(K) = D^{\star}_b(K) + \varepsilon_{b,K}, \qquad \varepsilon_{b,K} \geq 0, \tag{9}$$

where $D^{\star}_b(K)$ denotes the distortion after sufficiently long optimization and $\varepsilon_{b,K}$ is the remaining error caused by stopping early.

Sampled teacher fitting. We run teacher optimization only on a small representative subset. For each $K > 1$, we sample at most 128 admitted candidates across the range of $D_b^{\text{cheap}}(K)$. Each sampled candidate is reinitialized from the same deterministic template and optimized for $T_t = 160$ steps, producing the teacher error $D_b^{\text{teach}}(K)$. The sampling rule is detailed in Appendix A.2.3.

Per-K distortion calibration. The teacher errors provide longer-horizon references for correcting the 40-step candidate scores. For each candidate, we construct

$$\mathbf{x}_{b,K} = \left[1,\, \log\Big(1 + D_b^{\text{cheap}}(K)\Big)\,,\, \log(1 + D_b(1))\right]^\top, \tag{10}$$

where $D_b^{\text{cheap}}(K)$ describes the observed short fit and $D_b(1)$ provides a simple measure of the block's intrinsic difficulty. Since different values of K can have different convergence behavior, we fit a separate least-squares mapping for each K to predict the corresponding teacher distortion. This gives the calibrated score $\widetilde{D}_b(K)$, with $\widetilde{D}_b(1) = D_b(1)$.

3.3 Budget-Aware Gaussian Coding

After calibration in Sec. 3.2, each block has several candidate representations with different Gaussian counts K. The remaining question is how to choose among these candidates when the entire image is allowed to use only a limited number of Gaussians.

Gaussian-budget allocation. A larger K usually gives a block more fitting capacity, but spending more Gaussians on one block leaves fewer for the others. We therefore allocate a global budget B by jointly choosing the image partition $\mathcal{T}$ and the Gaussian count K_b for each selected block:

$$\min_{\mathcal{T},\mathbf{K}} \quad \sum_{b\in\mathcal{T}} \widetilde{D}_b(K_b) \qquad \text{s.t.} \quad \sum_{b\in\mathcal{T}} K_b \le B. \tag{11}$$

Intuitively, the allocator spends more Gaussians on blocks where additional capacity reduces reconstruction error the most, and fewer on blocks that are already represented well with small K.

Directly enforcing the global budget while choosing both the block structure and K is difficult. We assign a cost λ to each Gaussian and minimize

$$\mathcal{J}_\lambda = \sum_{b\in\mathcal{T}} \Big[\widetilde{D}_b(K_b) + \lambda K_b\Big]. \tag{12}$$

A smaller λ allows more Gaussians, while a larger λ favors more compact representations. We adjust λ until the total number of selected Gaussians is close to the target budget B, and retain the corresponding cheap-fitted candidates $\Theta_{b,K}^{\text{cheap}}$. The detailed allocation procedure is given in Appendix A.2.4.

Compression-aware parameter coding. Once the allocation is fixed, only the selected Gaussian parameters need to be transmitted. We encode the fitted geometry relative to the decoder-known template:

$$\Delta\mathbf{z}_{b,k} = \mathbf{z}_{b,k} - \mathbf{z}_{b,k}^{(0)}, \tag{13}$$

where $\mathbf{z}_{b,k}^{(0)}$ and $\mathbf{z}_{b,k}$ denote the template and fitted geometry, respectively. The resulting geometry residuals, relative weights, colors, topology, and Gaussian counts are quantized and losslessly entropy-coded with a standard range coder (Witten et al., 1987), forming the self-contained bitstream S. Detailed coding settings are provided in Appendix A.3.

3.4 Deterministic Gaussian Decoding

Since the bitstream stores an explicit Gaussian representation, decoding only requires entropy decoding, inverse quantization, and rasterization. The decoded block size r_b and Gaussian count K_b determine the shared geometric template, from which the fitted geometry is reconstructed as

$$\widehat{\mathbf{z}}_{b,k} = \mathbf{z}_{b,k}^{(0)}(r_b, K_b) + Q_z^{-1}\big(\mathbf{q}_{b,k}^z\big). \tag{14}$$

The remaining attributes are recovered analogously, yielding $\widehat{\Theta}$, which is rasterized using Eq. 4 to obtain the final reconstruction $\widehat{I}$. Details are provided in Appendix A.2.5.

Method	Low-rate				High-rate			
	PSNR (dB) ↑	MS-SSIM ↑	Training time (s) ↓	Decode speed (MP/s) ↑	PSNR (dB) ↑	MS-SSIM ↑	Training time (s) ↓	Decode speed (MP/s) ↑
Standard images								
GaussianImage	26.21	0.8829	332.12	213.58	30.47	0.9596	345.42	129.41
GaussianImage++	26.30	0.8853	341.99	**256.53**	30.30	0.9578	346.98	**241.74**
Image-GS	**27.95**	**0.8994**	77.50	227.00	32.08	0.9607	71.51	205.51
CaliSplat	27.71	0.8866	**22.03**	215.82	**32.95**	**0.9655**	**23.84**	175.90
Large images								
LIG	23.64	0.7104	848.69	390.84	29.70	0.9163	1015.55	**374.36**
SGI	26.67	**0.8634**	958.89	8.49	30.55	0.9533	1617.21	1.93
CaliSplat	**27.10**	0.8504	**94.86**	**478.19**	**31.86**	**0.9645**	**103.25**	255.24

Table 1: Quantitative comparison on standard- and large-image datasets. We compare with GaussianImage (Zhang et al., 2024a), GaussianImage++ (Li et al., 2026), Image-GS (Zhang et al., 2024b), LIG (Zhu et al., 2025), and SGI (Pan et al., 2026). The best and second-best results within each setting are shown in bold and underlined, respectively.

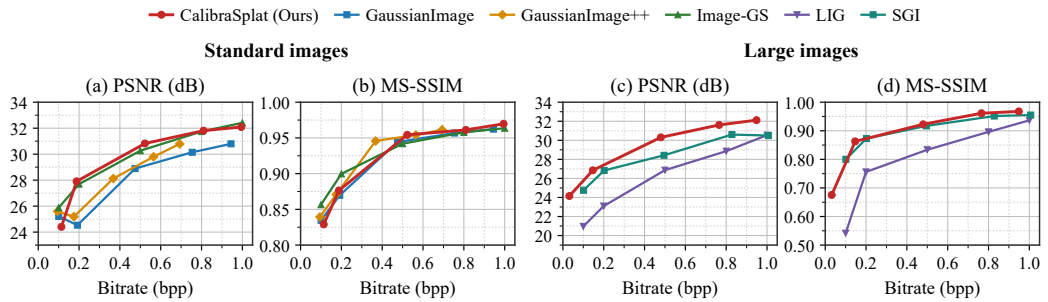


Figure 4: Rate-distortion curves of CaliSplat and different baselines on standard images and large images datasets.

4 Experiments

4.1 Experimental Setup

Datasets and Baselines. We evaluate CaliSplat on two groups of datasets covering both standard-resolution and large images. The standard-resolution set contains 75 images, with 25 images each from Kodak (Toderici et al., 2017), Tecnick (Asuni & Giachetti, 2013), and CLIC2020 (Cheng et al., 2020). The large-image set contains 9 images, with 3 images each from FGF2 (Wang et al., 2024a), ICB (Ororbia et al., 2019), and STimage (Chen et al., 2024). We compare with representative Gaussian-based methods, including GaussianImage (Zhang et al., 2024a), GaussianImage++ (Li et al., 2026), Image-GS (Zhang et al., 2024b), SGI (Pan et al., 2026), and LIG (Zhu et al., 2025).

Implementation details. We use 16×16 root blocks with candidate Gaussian counts $K \in \{1, 2, 4, 6, 8, 12, 16\}$. Each admitted candidate is optimized for 40 Adam steps with a learning rate of 0.035, while the sampled teacher fits use 160 steps and at most 128 samples for each K. Additional parameterization and coding settings are provided in Appendix A.3.

4.2 Results and Analysis

Rate–distortion performance. Fig. 4 compares rate–distortion performance on both standard- and large-image datasets. On standard images, CaliSplat remains competitive across the bitrate range and becomes particularly strong at higher rates. As shown in Table 1, Image-GS achieves slightly higher low-rate PSNR (27.95 vs. 27.71 dB), while CaliSplat obtains the best high-rate performance with 32.95 dB PSNR and 0.9655 MS-SSIM. This trend suggests that the additional representation capacity available at higher rates can be more effectively assigned to regions that benefit from additional Gaussians.

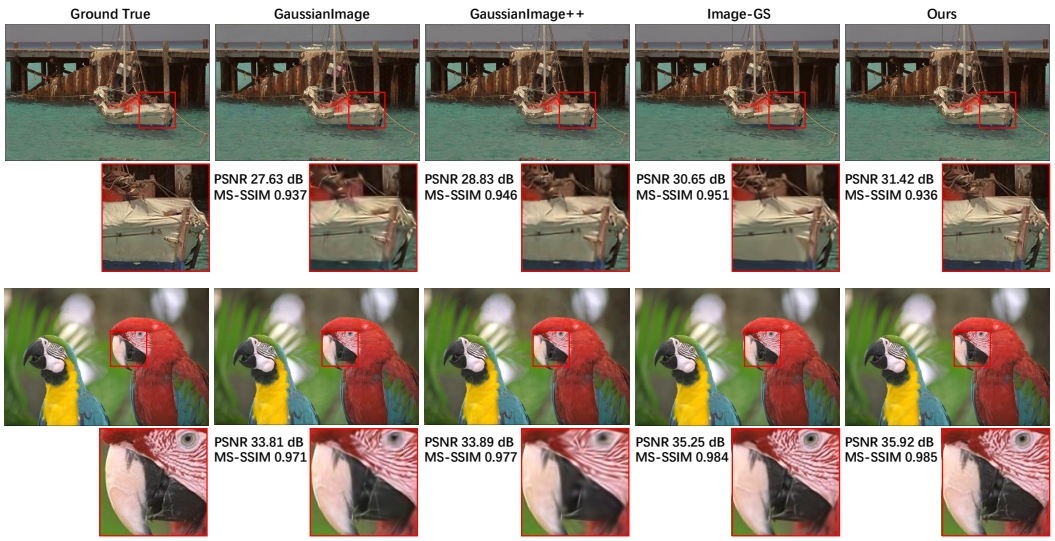


Figure 5: Qualitative reconstruction comparison of CaliSplat with other baselines. Our method preserves sharper edges and finer local structures with fewer visible artifacts.

The advantage becomes more pronounced on large images. As shown in Fig. 4 and Table 1, CaliSplat achieves the best PSNR at both low and high rates, reaching 27.10 and 31.86 dB, respectively. At the high-rate point, it outperforms SGI and LIG by 1.31 dB and 2.16 dB, respectively, while also achieving the highest MS-SSIM of 0.9645. Although SGI performs better in MS-SSIM at the low-rate point, CaliSplat overtakes it as the bitrate increases. Fig. 5 further shows that CaliSplat preserves finer local structures with fewer visible artifacts. Overall, these results demonstrate that CaliSplat maintains competitive performance on standard images while scaling favorably to larger and more complex content.

Encoding and decoding efficiency. CaliSplat also substantially reduces the cost of per-image optimization. On standard images, Table 1 reports fitting times of only 22.03–23.84 seconds, corresponding to approximately $3.0\times$–$15.5\times$ faster optimization than the compared Gaussian baselines. Despite this substantial reduction in fitting cost, CaliSplat maintains high decoding throughput of 175.90–215.82 MP/s, remaining competitive with efficient explicit Gaussian decoders.

The efficiency gain is even more pronounced on large images, where expensive per-image optimization becomes a greater bottleneck. CaliSplat requires only 94.86–103.25 seconds of fitting, yielding approximately $8.9\times$–$15.7\times$ faster optimization than LIG and SGI, while retaining 255.24–478.19 MP/s decoding throughput. These results indicate that the efficiency improvement primarily comes from reducing unnecessary encoder-side representation search rather than simplifying the final representation at the expense of reconstruction or decoding efficiency. Together with the RD results, CaliSplat provides a favorable trade-off between reconstruction quality, per-image fitting cost, and explicit Gaussian decoding efficiency.

4.3 Ablation Studies

Structured parameter coding. Fig. 6(a) evaluates the contribution of our structured parameter coding. We keep the selected Gaussians, quantization, and entropy coder fixed, and cumulatively replace absolute coding with template-relative geometry, relative log-weights, and palette-based RGB coding. The result reports the resulting bitstream size relative to absolute coding. The full scheme reduces the stream by 25.57% on DIV2K and 28.07% on ICB, with the largest additional saving coming from palette-based RGB coding. These results show that exploiting the structure of the Gaussian representation substantially reduces the actual transmitted bits without changing the underlying Gaussian representation.

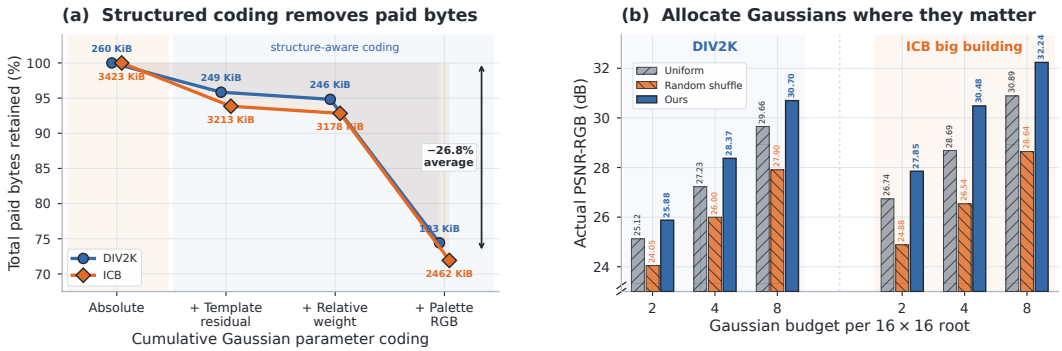


Figure 6: Ablation of parameter coding and Gaussian-budget allocation. **(a)** Cumulative coding stages progressively reduce the actual transmitted bytes while keeping the selected Gaussians and quantization fixed. **(b)** Budget-aware allocation consistently outperforms uniform allocation and a histogram-matched random spatial shuffle under the same Gaussian budget.

Variant	Candidates fitted	Fit work(%) ↓	Enc.(s) ↓	bpp ↓	PSNR ↑	SSIM ↑
Exhaustive	All	100.0	292.49	0.5165	30.485	0.8901
Random gate	Random subset	**65.0**	280.76	**0.5162**	30.316	0.8859
Full Model	$D_b(1)$ subset	**65.0**	**221.69**	0.5164	**30.509**	**0.8906**

Table 2: Ablation of sparse candidate fitting at $B/N = 4$. Fit work denotes the K-weighted candidate-fitting work relative to exhaustive fitting. Bold values indicate the best result.

Budget-aware Gaussian allocation. Fig. 6(b) evaluates how the available Gaussian budget should be distributed after candidate fitting. We compare our method with *Uniform*, which assigns the same K to all root blocks, and *Random shuffle*, which keeps the same overall K distribution as our method but randomly moves these counts across spatial locations. Across three budgets $B/N \in \{2, 4, 8\}$, our allocation outperforms both controls in all six cases. Compared with Uniform, it improves PSNR by 1.20 dB on average, showing that different regions benefit from different Gaussian counts. Compared with Random shuffle, the average gain increases to 2.92 dB, showing that these capacities must also be assigned to the appropriate spatial regions.

Sparse candidate fitting. Table 2 evaluates which higher-K candidates need to be fitted before Gaussian-budget allocation. *Exhaustive* fits all supported candidates. *Random gate* and the *Full Model* use the same reduced fitting work, but Random gate selects candidates randomly, whereas the Full Model uses $D_b(1)$ to decide which higher-K candidates are worth fitting. Compared with exhaustive fitting, the Full Model reduces the K-weighted fitting work from 100% to 65% while slightly improving PSNR from 30.485 to 30.509 dB. At the same 65% fitting work, random selection gives lower PSNR. This shows that many candidate fits can be skipped, but the candidates should be selected using $D_b(1)$ rather than removed at random.

5 Conclusion

We presented CaliSplat, an efficient per-image Gaussian codec that reduces content-adaptive representation search while preserving lightweight explicit decoding. By using the closed-form single-Gaussian distortion to guide sparse candidate fitting, followed by calibrated budget-aware allocation, CaliSplat concentrates optimization and Gaussian capacity where they are most useful. The selected representation is further compressed using deterministic templates and structured template-relative coding. Experiments on standard and large image datasets show competitive or superior rate–distortion performance with 3.0×–15.7× faster per-image fitting and high-throughput decoding. These results demonstrate the effectiveness of selective representation search for efficient content-adaptive Gaussian image compression.

AI USE STATEMENT

In this work, we used generative AI tools to assist with feedback on experimental design and the interpretation and presentation of experimental results. We additionally used generative AI tools for drafting and editing parts of the manuscript, improving writing clarity and organization, literature search, identifying relevant references, and formatting citations. All AI-assisted technical statements were manually reviewed and verified by the authors. All reported quantitative results were obtained from author-run experiments and were not generated or modified by generative AI. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Nicola Asuni and Andrea Giachetti. Testimages: A large data archive for display and algorithm testing. *Journal of Graphics Tools*, 17(4):113–125, 2013.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

Jona Ballé, Luca Versari, Emilien Dupont, Hyunjik Kim, and Matthias Bauer. Good, cheap, and fast: Overfitted image compression with wasserstein distortion. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23259–23268. IEEE, 2025.

Jiawen Chen, Muqing Zhou, Wenrong Wu, Jinwei Zhang, Yun Li, and Didong Li. Stimage-1k4m: A histopathology image-gene expression dataset for spatial transcriptomics. *Advances in Neural Information Processing Systems*, 37:35796–35823, 2024.

Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feed-forward 3d gaussian splatting compression. In *International Conference on Learning Representations*, 2025.

Yunuo Chen, Bing He, Zezheng Lyu, Hongwei Hu, Qunshan Gu, Yuan Tian, and Guo Lu. Adaptive learned image compression with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12150–12161, June 2026a.

Yunuo Chen, Bing He, Zezheng Lyu, Hongwei Hu, Qunshan Gu, Yuan Tian, and Guo Lu. Adaptive learned image compression with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12150–12161, 2026b.

Yunuo Chen, Zezheng Lyu, Bing He, Hongwei Hu, Qi Wang, Yuan Tian, Li Song, Wenjun Zhang, and Guo Lu. Content-aware mamba for learned image compression. In *International Conference on Learning Representations*, volume 2026, pp. 78452–78477, 2026c.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7936–7945. IEEE, 2020.

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021.

Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Goliński, Yee Whye Teh, and Arnaud Doucet. COIN++: Neural compression across modalities. *Transactions on Machine Learning Research*, 2022.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4447.

Donghui Feng, Zhengxue Cheng, Shen Wang, Ronghua Wu, Hongwei Hu, Guo Lu, and Li Song. Linear attention modeling for learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7623–7632, 2025.

Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. 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*, pp. 5718–5727, 2022.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023. doi: 10.1145/3592433.

Hyunjik Kim, Matthias Bauer, Lucas Theis, Jonathan Richard Schwarz, and Emilien Dupont. 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*, pp. 9347–9358, 2024.

Haowei Kuang, Wenhan Yang, Zongming Guo, and Jiaying Liu. Cross-granularity online optimization with masked compensated information for learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16514–16523, 2025.

Théo Ladune, Pierrick Philippe, Félix Henry, Gordon Clare, and Thomas Leguay. COOL-CHIC: Coordinate-based low complexity hierarchical image codec. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13515–13522, 2023.

Soonbin Lee, Fangwen Shu, Yago Sanchez, Thomas Schierl, and Cornelius Hellge. Compression of 3d gaussian splatting with optimized feature planes and standard video codecs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 25496–25505, 2025.

Tiantian Li, Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Jun Zhang, and Yan Wang. GaussianImage++: Boosted image representation and compression with 2d gaussian splatting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 6442–6449, 2026. doi: 10.1609/aaai.v40i8.37572.

Yuqi Li, Haotian Zhang, Li Li, and Dong Liu. Learned image compression with hierarchical progressive context modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18834–18843, 2025.

Jinming Liu, Heming Sun, and Jiro Katto. Learned image compression with mixed transformer–cnn architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14388–14397, 2023. doi: 10.1109/CVPR52729.2023.01383.

Jingbo Lu, Leheng Zhang, Xingyu Zhou, Mu Li, Wen Li, and Shuhang Gu. Learned image compression with dictionary-based entropy model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12850–12859, 2025a.

Jingbo Lu, Leheng Zhang, Xingyu Zhou, Mu Li, Wen Li, and Shuhang Gu. Learned image compression with dictionary-based entropy model. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12850–12859. IEEE, 2025b.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20654–20664, 2024.

Huidong Ma, Xinyan Shi, Hui Sun, Xiaofei Yue, Xiaoguang Liu, Gang Wang, and Wentong Cai. Learned image compression via sparse attention and adaptive frequency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41278–41287, 2026.

David Minnen, Johannes Ballé, and George D. Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, volume 31, pp. 10794–10803, 2018.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10349–10358, 2024.

Alexander G Ororbia, Ankur Mali, Jian Wu, Scott O'Connell, William Dreese, David Miller, and C Lee Giles. Learned neural iterative decoding for lossy image compression systems. In *2019 Data Compression Conference (DCC)*, pp. 3–12. IEEE, 2019.

Zixuan Pan, Kaiyuan Tang, Jun Xia, Yifan Qin, Lin Gu, Chaoli Wang, Jianxu Chen, and Yiyu Shi. SGI: Structured 2d gaussians for efficient and compact large image representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Shiyu Qin, Jinpeng Wang, Yimin Zhou, Bin Chen, Tianci Luo, Baoyi An, Tao Dai, Shu-Tao Xia, and Yaowei Wang. Cassic: Towards content-adaptive state-space models for learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15727–15736, 2025.

Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. In *European conference on computer vision*, pp. 74–91. Springer, 2022.

Kedar Tatwawadi, Parisa Rahimzadeh, Zhanghao Sun, Zhiqi Chen, Ziyun Yang, Sanjay Nair, Divija Hasteer, and Oren Rippel. What matters in practical learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12095–12105, 2026.

George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 5306–5314, 2017.

Yingying Wang, Xuanhua He, Yuhang Dong, Yunlong Lin, Yue Huang, and Xinghao Ding. Cross-modality interaction network for pan-sharpening. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–16, 2024a.

Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex C. Kot, and Bihan Wen. ContextGS: Compact 3d gaussian splatting with anchor level context model. In *Advances in Neural Information Processing Systems*, 2024b.

Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

Shuzhao Xie, Weixiang Zhang, Chen Tang, Yunpeng Bai, Rongwei Lu, Shijia Ge, and Zhi Wang. MesonGS: Post-training compression of 3d gaussians via efficient attribute transformation. In *European Conference on Computer Vision*, 2024.

Xianben Yang, Tao Wang, Yuxuan Li, Yi Jin, and Haibin Ling. GS^2: Graph-based spatial distribution optimization for compact 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33259–33268, 2026.

Zhaojie Zeng, Yuesong Wang, Tao Guan, Chao Yang, and Lili Ju. Instant GaussianImage: A generalizable and self-adaptive image representation via 2d gaussian splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27896–27905, 2025.

Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. GaussianImage: 1000 FPS image representation and compression by 2d gaussian splatting. In *Proceedings of the European Conference on Computer Vision*, 2024a. doi: 10.1007/978-3-031-72673-6_18.

Yunxiang Zhang, Alexandr Kuznetsov, Akshay Jindal, Kenneth Chen, Anton Sochenov, Anton Kaplanyan, and Qi Sun. Image-GS: Content-adaptive image representation via 2d gaussians, 2024b.

Lingting Zhu, Guying Lin, Jinnan Chen, Xinjie Zhang, Zhenchao Jin, Zhao Wang, and Lequan Yu. Large images are gaussians: High-quality large image representation with levels of 2d gaussian splatting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 10977–10985, 2025. doi: 10.1609/aaai.v39i10.33193.

Algorithm 1 Per-Image Encoding with CaliSplat

Require: RGB image I; Gaussian budget B
Ensure: Self-contained bitstream S

Partition I into supported spatial blocks

Stage 1: Sparse Gaussian candidate fitting Sec. 3.1; Appendix. A.2.1, A.2.2

for each block b **do**
  Compute the analytic $K = 1$ solution $(\boldsymbol{c}_b^\star, D_b(1))$ ▷ Eq. 5
end for
for each supported $K > 1$ **do**
  Select admitted blocks $\mathcal{A}_K$ using $D_b(1)$ ▷ Eq. 6
  for each $b \in \mathcal{A}_K$ **do**
    Initialize template geometry and $C^{(0)} = W^\dagger X$ ▷ Eq. 7
    for $t = 1, \ldots, T_c$ **do**
      Rasterize $\widehat{I}_{b,K}^{(t)} = \mathcal{R}(\Theta_{b,K}^{(t)})$ ▷ Eq. 4
      Compute $\mathcal{L}_{b,K}^{(t)} = \sum_{\mathbf{p}\in b} \|I_b(\mathbf{p}) - \widehat{I}_{b,K}^{(t)}(\mathbf{p})\|_2^2$
      Backpropagate $\nabla_{\boldsymbol{\rho}_{b,K}} \mathcal{L}_{b,K}^{(t)}$ and update $\boldsymbol{\rho}_{b,K}$ with Adam
    end for
    Store $\Theta_{b,K}^{\text{cheap}}$ and $D_b^{\text{cheap}}(K)$ ▷ Eq. 8
  end for
end for

Stage 2: Sampled teacher calibration Sec. 3.2; Appendix. A.2.3

for each $K > 1$ **do**
  Sample at most 128 admitted blocks across the $D_b^{\text{cheap}}(K)$ range
  for each sampled block b **do**
    Reinitialize from the same deterministic template
    Repeat the same rasterization–loss–backpropagation loop for T_t steps
    Record $D_b^{\text{teach}}(K)$
  end for
  Construct calibration features $\mathbf{x}_{b,K}$ ▷ Eq. 10
  Fit the per-K least-squares mapping and predict $\widetilde{D}_b(K)$ for all admitted candidates
end for
Set $\widetilde{D}_b(1) \leftarrow D_b(1)$
Discard teacher parameters and retain $\Theta_{b,K}^{\text{cheap}}$ ▷ **teachers calibrate scores only**

Stage 3: Budget-aware Gaussian allocation and coding Sec. 3.3; Appendix A.2.4

Search λ and solve the hierarchical allocation using calibrated distortions ▷ Eqs. 11–12
Obtain the selected topology $\mathcal{T}^\star$ and Gaussian counts $\{K_b^\star\}$
for each selected leaf $b \in \mathcal{T}^\star$ **do**
  Retrieve the selected $\Theta_{b,K_b^\star}^{\text{cheap}}$
  Encode template-relative geometry, relative weights, and palette-coded colors ▷ Eq. 13
end for
Quantize and range-code the selected topology, Gaussian counts and parameters into S
return S

A Appendix

A.1 More Results

A.1.1 Qualitative allocation analysis.

Fig. 7 examines where the available Gaussians are placed across different types of images. Each example contains the source image and its allocation map. As shonw in this visualization, blocks around locally detailed regions are often assigned larger K_b, whereas smooth backgrounds and nearly empty regions tend to use smaller K_b. For example, detailed building facades and tissue

Algorithm 2 Deterministic Gaussian Decoding

Require: CaliSplat bitstream S
Ensure: Reconstructed RGB image $\widehat{I}$

Stage 1: Recover the explicit Gaussian representation
Sec. 3.4; inverse details in Appendix A.2.5

Decode the selected topology, Gaussian counts, and quantized Gaussian symbols
for each selected leaf b **do**
  Reconstruct the deterministic template from (r_b, K_b)
  Recover coding-space geometry from the template and transmitted residual ▷ Eq. 14
  Recover relative weights and Gaussian colors
  Assemble the decoded Gaussian set $\widehat{\Theta}_b$
end for
Assemble $\widehat{\Theta} = \bigcup_b \widehat{\Theta}_b$ ▷ Eq. 1

Stage 2: Deterministic Gaussian rasterization
for each selected leaf b **do**
  Rasterize $\widehat{\Theta}_b$ using the local Gaussian mixture ▷ Eq. 4
end for
Assemble the reconstructed blocks and obtain $\widehat{I} = \mathcal{R}(\widehat{\Theta})$ ▷ Eq. 1
return $\widehat{I}$

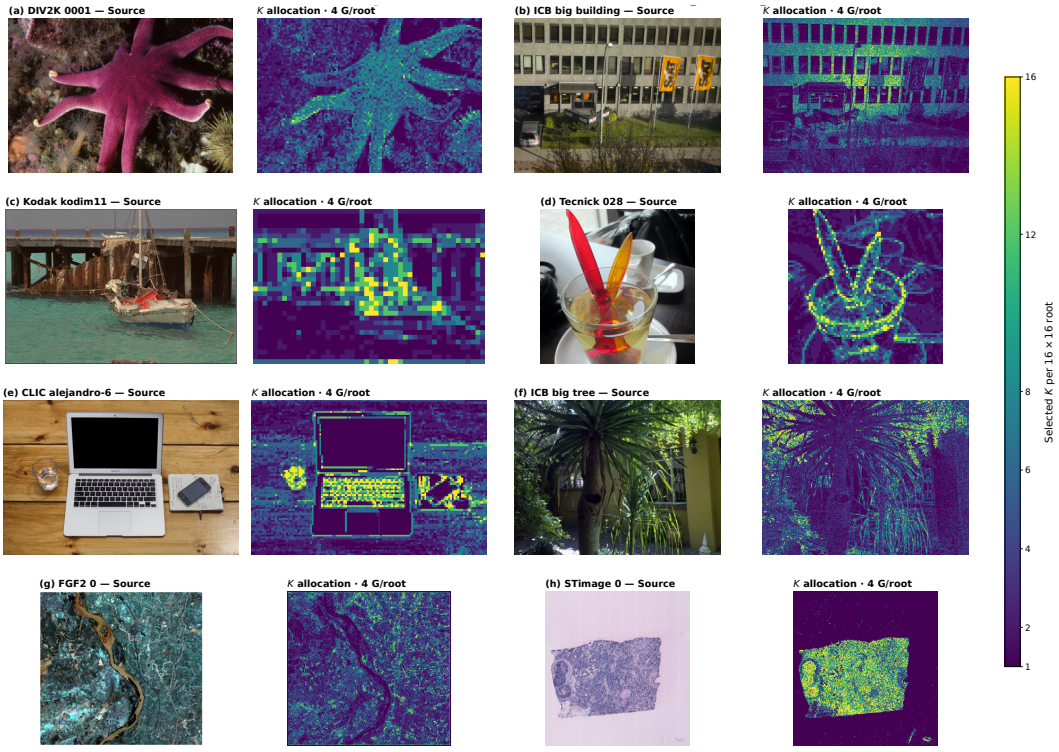


Figure 7: Qualitative visualization of Gaussian-budget allocation across eight images. Each example shows the source image and the selected Gaussian count K_b for each 16×16 root block under the same average budget $B/N = 4$. Larger K_b values are concentrated in regions with richer local structure, while smoother regions receive fewer Gaussians.

regions receive more Gaussians than nearby smooth or low-detail areas. These results show that the allocator does not distribute the Gaussian budget uniformly. Instead, it gives more capacity to regions where additional Gaussians are more useful for reconstruction. The visualization supports

Method	Low-rate				High-rate			
	PSNR (dB) ↑	MS-SSIM ↑	Training time (s) ↓	Decode speed (MP/s) ↑	PSNR (dB) ↑	MS-SSIM ↑	Training time (s) ↓	Decode speed (MP/s) ↑
GLIC	**33.34**	**0.9655**	N/A	2.67	**38.45**	0.9874	N/A	2.65
CMIC	<u>33.30</u>	<u>0.9654</u>	N/A	<u>3.44</u>	<u>38.43</u>	<u>0.9890</u>	N/A	<u>3.43</u>
DCAE	33.26	0.9593	N/A	3.03	38.23	**0.9891**	N/A	3.00
CaliSplat	27.71	0.8866	**22.03**	**215.82**	32.95	0.9655	**23.84**	**175.90**

Table 3: Quantitative comparison with representative learned image codecs GLIC (Chen et al., 2026a), CMIC (Chen et al., 2026c), and DCAE (Lu et al., 2025b). The best and second-best results for each metric are shown in bold and underlined, respectively.

that our method adapts both the amount and the spatial placement of Gaussian capacity to the image content. More results are in Fig. 8-11.

A.1.2 Comparison with learned image codecs.

Table 3 further compares CaliSplat with recent learned image codecs, including GLIC (Chen et al., 2026a), CMIC (Chen et al., 2026c), and DCAE (Lu et al., 2025b). These methods achieve clearly stronger rate–distortion performance, benefiting from dataset-level priors learned offline from large image corpora. In contrast, Gaussian-based codecs optimize an explicit representation for each individual image and do not directly exploit the same cross-image priors, making this RD gap a broader limitation of the current per-image Gaussian compression paradigm. CaliSplat nevertheless offers a substantially different decoding trade-off: it reaches 175.90–215.82 MP/s, compared with 2.65–3.44 MP/s for the learned codecs, corresponding to approximately $51.3\times$–$80.8\times$ higher throughput. Together with Table 1, these results show that CaliSplat primarily improves the efficiency of explicit Gaussian compression by reducing per-image fitting cost while preserving its lightweight and highly efficient decoding.

A.2 Additional Method Details

A.2.1 Gaussian Parameterization and Initialization

For a block of side length r_b, pixel indices $i, j = 0, \dots, r_b - 1$ are mapped to normalized local coordinates $\mathbf{p}_{ij} = (-1 + 2i/(r_b - 1),\ -1 + 2j/(r_b - 1))$. For a Gaussian centered at $\boldsymbol{\mu} = (\mu_y, \mu_x)$, the coordinates along its rotated principal axes are $d_1 = (p_y - \mu_y)\cos\phi + (p_x - \mu_x)\sin\phi$ and $d_2 = -(p_y - \mu_y)\sin\phi + (p_x - \mu_x)\cos\phi$. These normalized coordinates are used in Eq. 3 for all supported block sizes.

During gradient-based per-image fitting, Gaussian parameters are optimized through unconstrained raw variables $\boldsymbol{\rho}$ and mapped to valid physical values by

$$\boldsymbol{\mu} = 1.15\tanh(\boldsymbol{\rho}_\mu), \qquad \boldsymbol{s} = \mathrm{softplus}(\boldsymbol{\rho}_s) + 0.025, \qquad \phi = \rho_\phi, \quad a = \rho_a, \quad \boldsymbol{c} = 255\,\sigma(\boldsymbol{\rho}_c). \tag{15}$$

The extended center range allows slight motion beyond the normalized block boundary, while the scale floor avoids degenerate ellipses. Gradients are backpropagated through these mappings and the differentiable Gaussian rasterizer.

For each (r_b, K), we initialize the K Gaussians on the most balanced regular grid satisfying $n_y n_x = K$. Among valid factor pairs, we choose the one with aspect ratio closest to one; ties and Gaussian ordering follow fixed deterministic rules shared by the encoder and decoder. For grid location (i, j), the template is

$$\boldsymbol{\mu}_{i,j}^{(0)} = \left(\frac{i + \frac{1}{2}}{n_y^\star} \cdot 2 - 1,\ \frac{j + \frac{1}{2}}{n_x^\star} \cdot 2 - 1\right), \qquad \boldsymbol{s}^{(0)} = \left(\frac{1.35}{n_y^\star},\ \frac{1.35}{n_x^\star}\right), \qquad \phi^{(0)} = 0. \tag{16}$$

The factor 1.35 provides overlap between neighboring template Gaussians. Because the template is determined entirely by (r_b, K), it requires no additional side information.

With the template geometry fixed and initial log-weights set to zero, the rasterizer defines a responsibility matrix $W \in \mathbb{R}^{r_b^2 \times K}$. For a flattened RGB block $X \in \mathbb{R}^{r_b^2 \times 3}$, the initial Gaussian colors are

obtained in closed form:

$$C^{(0)} = \arg\min_{C} \|WC - X\|_F^2 = W^\dagger X. \tag{17}$$

Since W depends only on (r_b, K), its pseudoinverse is reused across blocks with the same configuration. For $K = 1$, the analytic mean is clipped to $[0.5, 254.5]$ only when converted to raw sigmoid coordinates, avoiding infinite inverse-sigmoid values; the exact $D_b(1)$ in Eq. 5 is computed from the unclipped mean.

A.2.2 Sparse Candidate Construction

For 16×16 blocks, we use

$$\mathcal{K}_{16} = \{1, 2, 4, 6, 8, 12, 16\}, \qquad (\rho_1, \rho_2, \rho_4, \rho_6, \rho_8, \rho_{12}, \rho_{16}) = (1.0, 1.0, 0.9, 0.8, 0.7, 0.6, 0.5), \tag{18}$$

where ρ_K denotes the fraction of blocks retained for candidate size K. For a scale containing N blocks, the corresponding admission count is

$$M_K = \begin{cases} 0, & \rho_K = 0, \\ \min\{N,\ \max(1,\ \lceil \rho_K N \rceil)\}, & \rho_K > 0. \end{cases} \tag{19}$$

Blocks are ranked by the analytic difficulty $D_b(1)$ as described in Eq. 6; candidates outside $\mathcal{A}_K$ are treated as unavailable by assigning $D_b^{\text{cheap}}(K) = +\infty$.

A.2.3 Teacher Sampling and Calibration

Let $N_K = |\mathcal{A}_K|$ and let $\sigma_K(0), \ldots, \sigma_K(N_K - 1)$ denote the admitted blocks sorted by increasing $D_b^{\text{cheap}}(K)$. For each $K > 1$, we select at most 128 teachers, uniformly along this sorted error axis. With $m_K = \min(128, N_K)$, the sampled set is

$$\mathcal{S}_K = \{\sigma_K(q_i)\}_{i=0}^{m_K - 1}, \qquad q_i = \begin{cases} 0, & m_K = 1, \\ \text{round}\left(\dfrac{i(N_K - 1)}{m_K - 1}\right), & m_K > 1. \end{cases} \tag{20}$$

This sampling covers the cheap-distortion range rather than concentrating teachers around its most populated region.

Each sampled candidate is reinitialized from the same deterministic template used by the cheap fit and optimized independently for $T_t = 160$ Adam steps. We use the same RGB squared-error objective, learning rate, and restart setting as in sparse fitting. Importantly, teacher optimization does not continue from $\Theta_{b,K}^{\text{cheap}}$; its purpose is to provide a longer-horizon reference $D_b^{\text{teach}}(K)$.

For each sampled block, the regression target is $y_{b,K} = \log(1 + D_b^{\text{teach}}(K))$. Stacking the features $\mathbf{x}_{b,K}$ from Eq. 10 and their targets gives X_K and $\mathbf{y}_K$. We fit a separate least-squares model for each K:

$$\boldsymbol{\beta}_K = \arg\min_{\boldsymbol{\beta}} \|X_K \boldsymbol{\beta} - \mathbf{y}_K\|_2^2. \tag{21}$$

The least-squares system is solved directly, without explicitly forming $(X_K^\top X_K)^{-1}$.

For every admitted candidate, the predicted log-distortion is mapped back to the original distortion domain:

$$\widetilde{D}_b(K) = \max\left\{\text{expm1}\left(\mathbf{x}_{b,K}^\top \boldsymbol{\beta}_K\right), 0\right\}. \tag{22}$$

For $K = 1$, we retain the exact analytic value $\widetilde{D}_b(1) = D_b(1)$, while candidates excluded by the sparse gate remain unavailable with infinite distortion.

A.2.4 BUDGET-AWARE HIERARCHICAL ALLOCATION

For a fixed multiplier λ, each tree node chooses between remaining unsplit and being replaced by its children. Let $\mathcal{K}_n$ denote the legal Gaussian counts at node n. The bottom-up dynamic program is

$$\begin{aligned} F_n^{\text{leaf}}(\lambda) &= \min_{K \in \mathcal{K}_n} \left[\widetilde{D}_n(K) + \lambda K \right], \\ F_n^{\text{split}}(\lambda) &= \sum_{c \in \text{child}(n)} F_c(\lambda), \\ F_n(\lambda) &= \min \left\{ F_n^{\text{leaf}}(\lambda), F_n^{\text{split}}(\lambda) \right\}. \end{aligned} \tag{23}$$

At an unsplit node, candidate counts are considered in increasing order so that local ties favor the smaller K; ties between split and unsplit configurations favor the unsplit node.

For a fixed candidate table and λ, this recursion gives the optimal legal configuration under the Lagrangian objective. The argument follows directly by induction on tree depth: terminal nodes choose their best legal candidate, while a split node combines optimal solutions from disjoint child regions and compares their total cost with its own best unsplit candidate.

To meet a target budget B, we search over λ using the total Gaussian count $G(\lambda) = \sum_{b \in \mathcal{T}^\star(\lambda)} K_b^\star(\lambda)$, which is non-increasing in λ. Starting from $\lambda_{\text{lo}} = 0$ and $\lambda_{\text{hi}} = 1$, we double the upper bound until $G(\lambda_{\text{hi}}) \le B$, then perform 20 bisection iterations. Among the feasible configurations near the final boundary, we retain the one with lower calibrated distortion. Because the candidate space is discrete, varying λ need not recover the exact constrained optimum for every integer budget.

A.2.5 DETERMINISTIC DECODING DETAILS

The decoder first recovers the selected topology, Gaussian counts, and quantized parameter symbols. For each leaf, the decoded pair (r_b, K_b) uniquely reconstructs the same deterministic geometric template used by the encoder. Let q_μ and q_Σ denote the quantization factors for center and log-covariance residuals. The coding-space geometry is recovered as

$$\widehat{\mathbf{g}}_{b,k} = \mathbf{g}_{b,k}^{(0)} + \left[\frac{z_{\mu_y}}{q_\mu}, \frac{z_{\mu_x}}{q_\mu}, \frac{z_m}{q_\Sigma}, \frac{z_u}{q_\Sigma}, \frac{z_v}{q_\Sigma} \right]^\top. \tag{24}$$

Writing $\alpha = \sqrt{\widehat{u}^2 + \widehat{v}^2}$, the anisotropic scale and orientation are recovered by

$$\widehat{\phi} = \begin{cases} \frac{1}{2} \operatorname{atan2}(\widehat{v}, \widehat{u}), & \alpha > 10^{-10}, \\ 0, & \text{otherwise}, \end{cases} \qquad \widehat{s}_1 = 2^{\widehat{m}+\alpha}, \qquad \widehat{s}_2 = 2^{\widehat{m}-\alpha}. \tag{25}$$

Relative weights and palette-coded colors are inverted analogously, yielding the explicit Gaussian representation $\widehat{\Theta}$. Each selected leaf is then rasterized using the same Gaussian formulation as in Eqs. 3–4. The reconstructed image is rounded and clipped to RGB8, assembled over all leaves, and cropped to the original $H \times W$ resolution.

A.3 IMPLEMENTATION DETAILS

The global Gaussian budget is determined using 20 bisection iterations over the Lagrangian multiplier. For parameter coding, we use quantization denominators of 20, 10, and 3 for Gaussian centers, log-covariance parameters, and relative weights, respectively, together with a color quantization step of 2. The color palette contains at most 768 entries and is optimized with up to four Lloyd iterations. Candidate construction at different scales is executed in parallel: higher-order 16×16 Gaussian fitting is performed on the GPU, while the analytic 8×8 $K = 1$ processing is handled by the CPU.

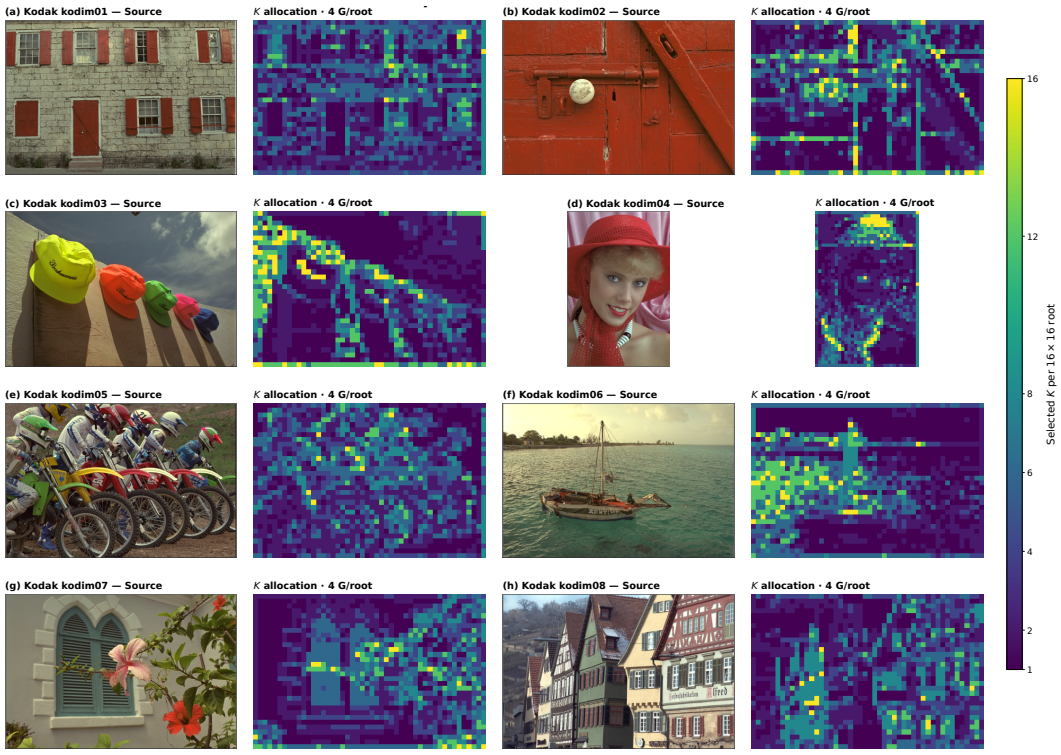


Figure 8: More qualitative visualization of Gaussian-budget allocation across eight images.

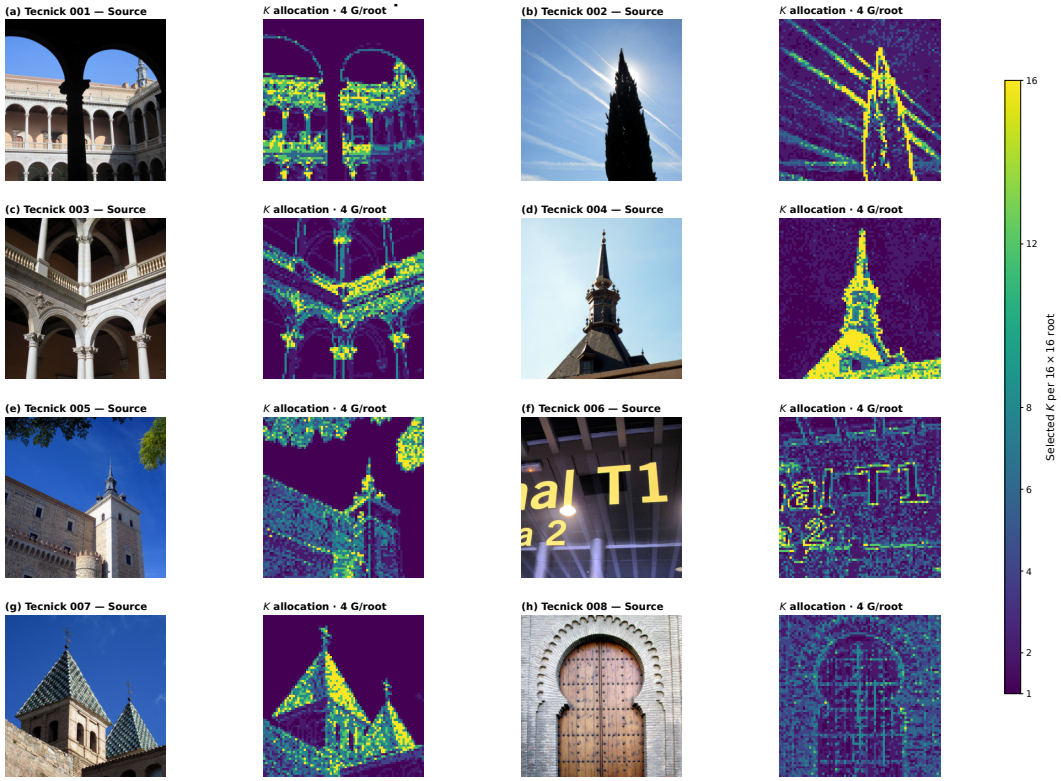


Figure 9: More qualitative visualization of Gaussian-budget allocation across eight images.

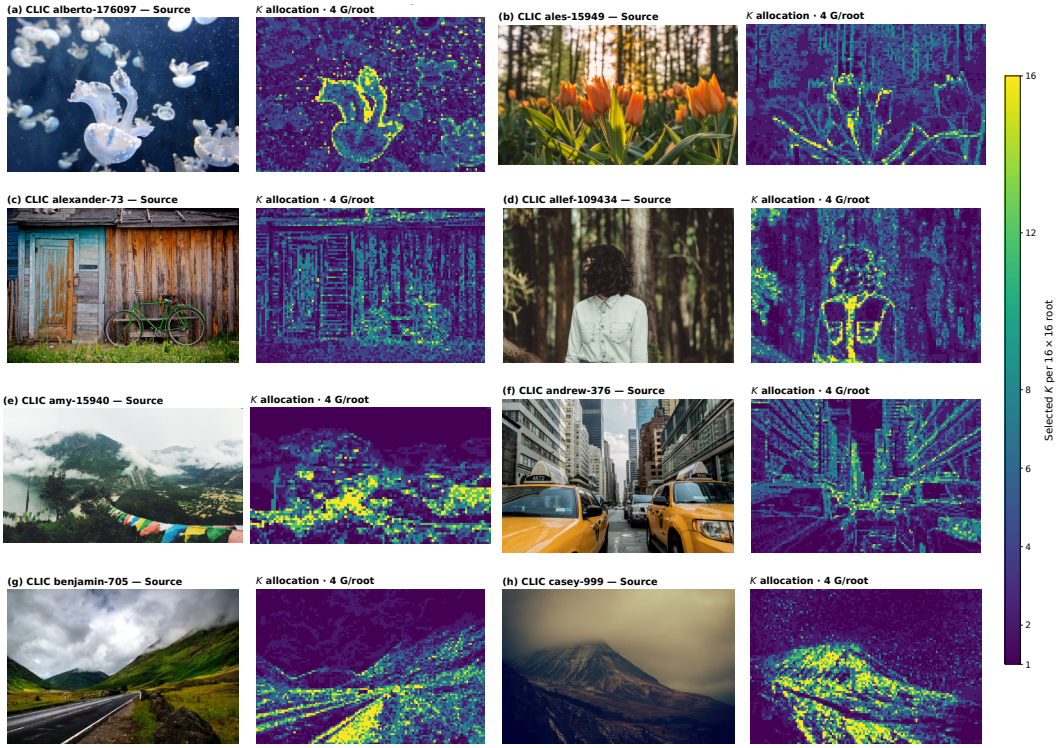


Figure 10: More qualitative visualization of Gaussian-budget allocation across eight images.

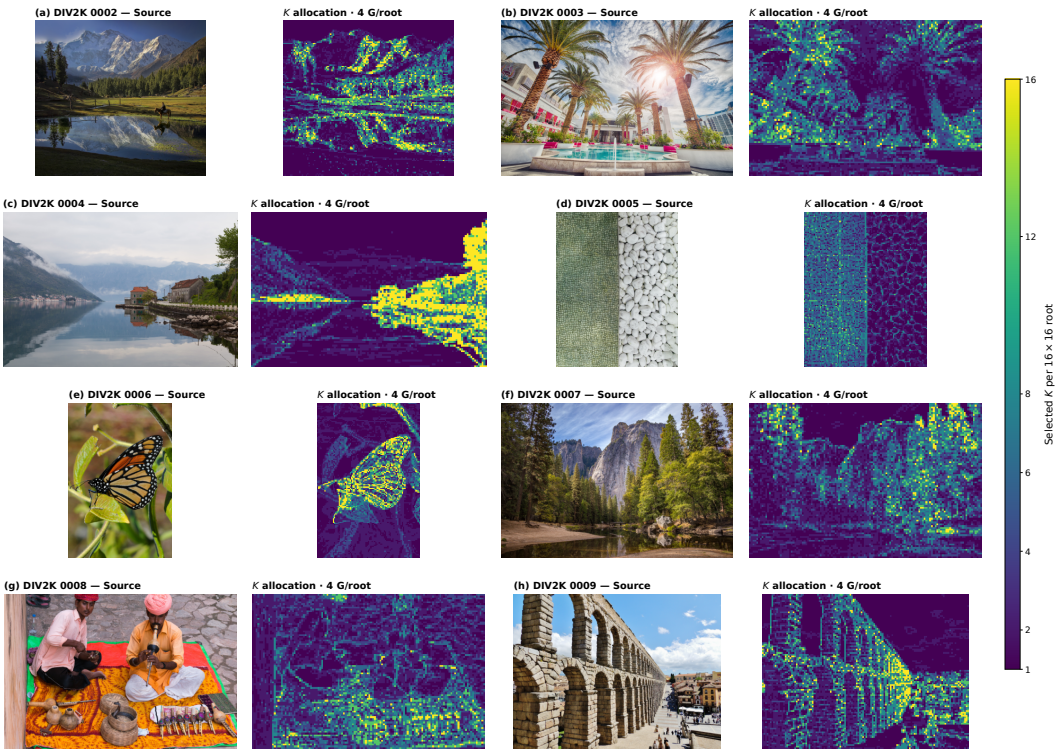


Figure 11: More qualitative visualization of Gaussian-budget allocation across eight images.

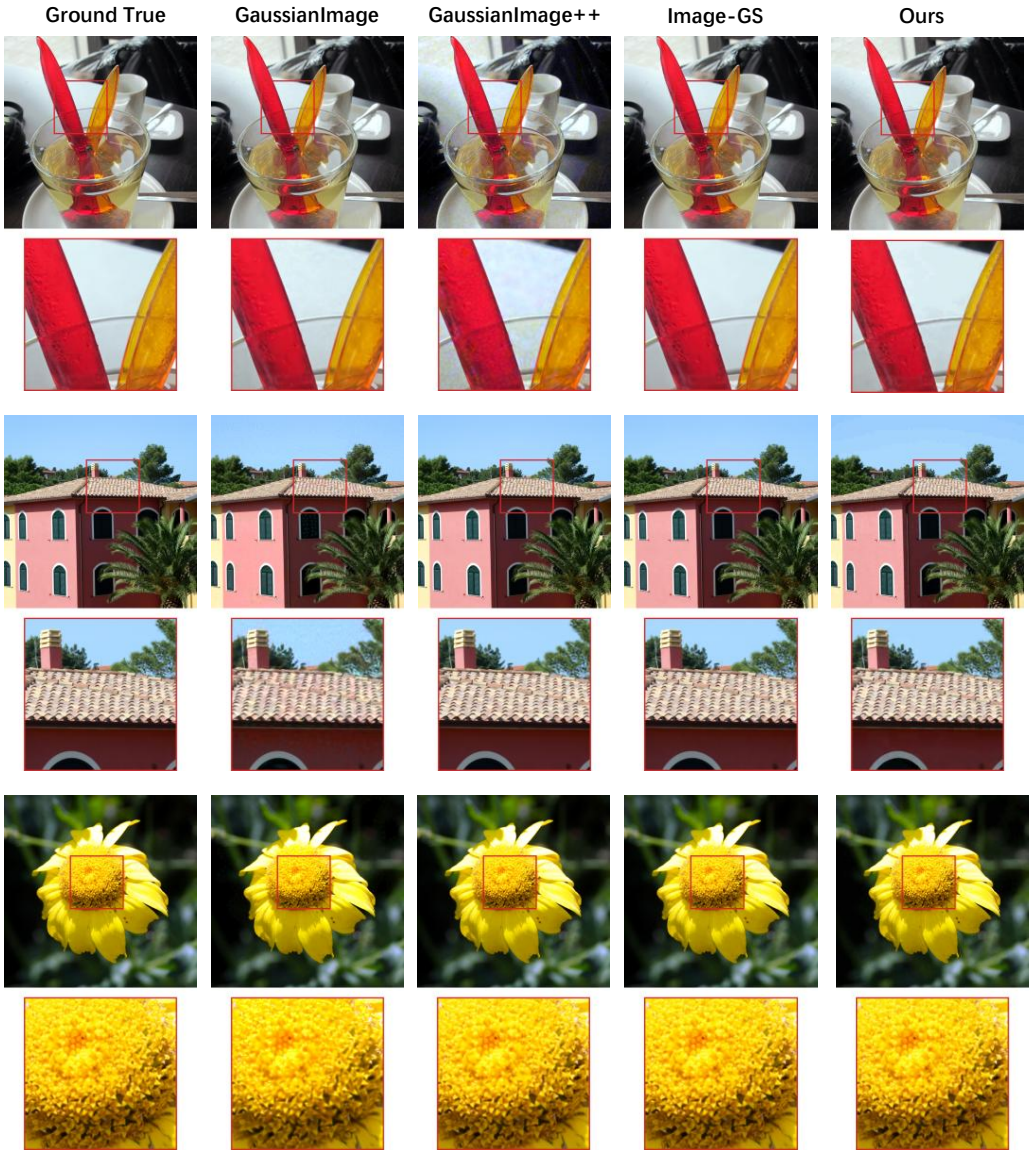


Figure 12: More qualitative reconstruction results of CaliSplat with other baselines.

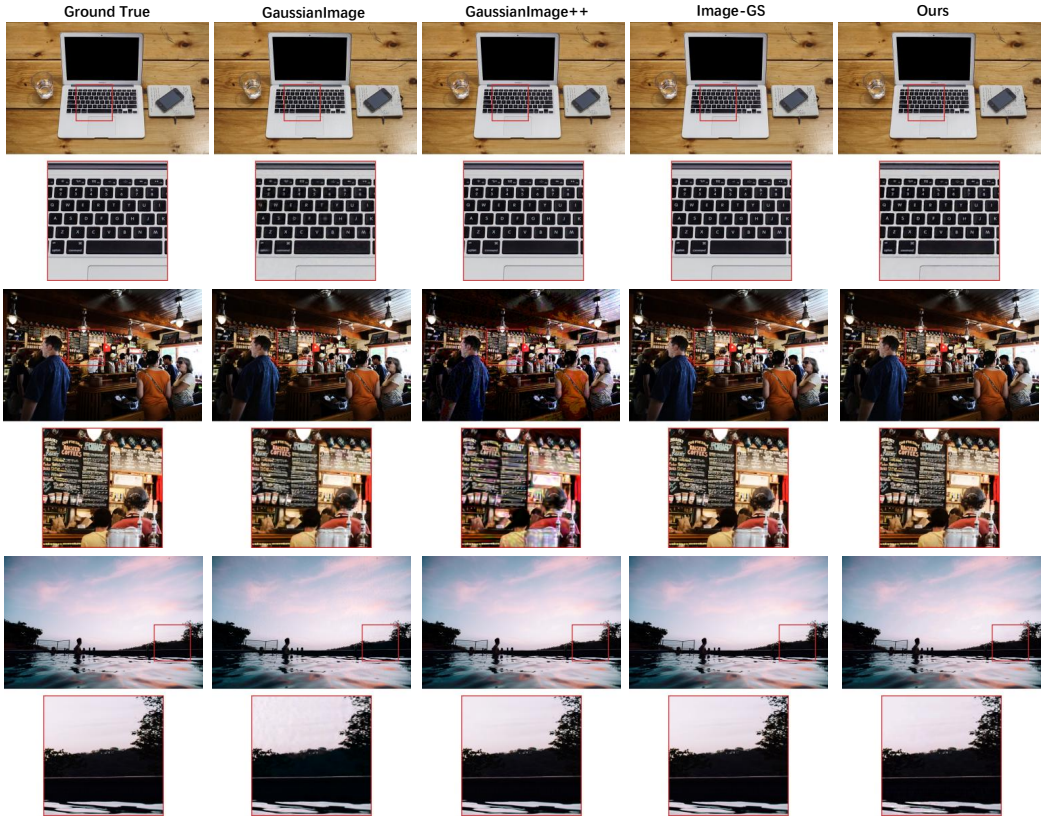


Figure 13: More qualitative reconstruction results of CaliSplat with other baselines.

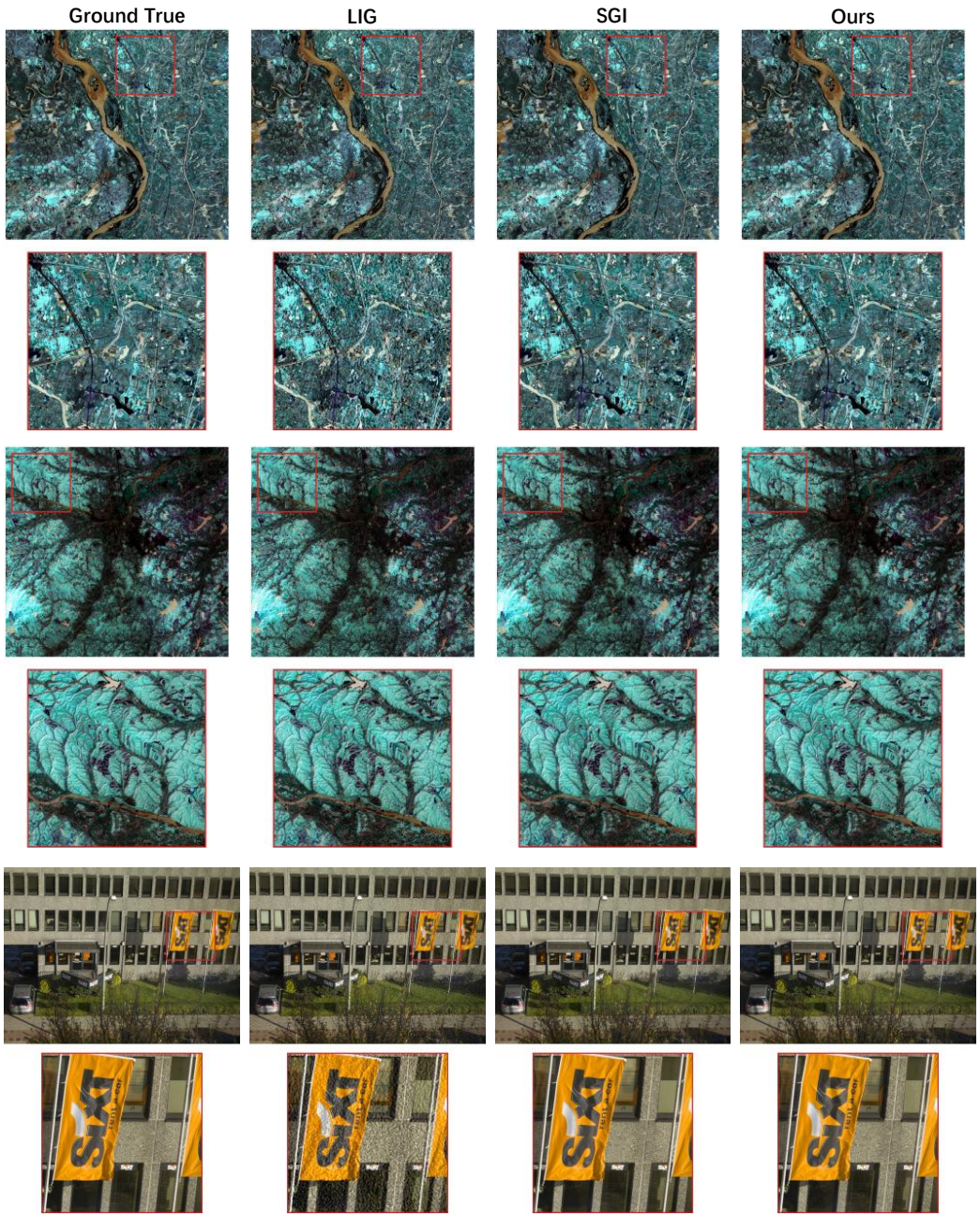


Figure 14: More qualitative reconstruction results of CaliSplat with other baselines.

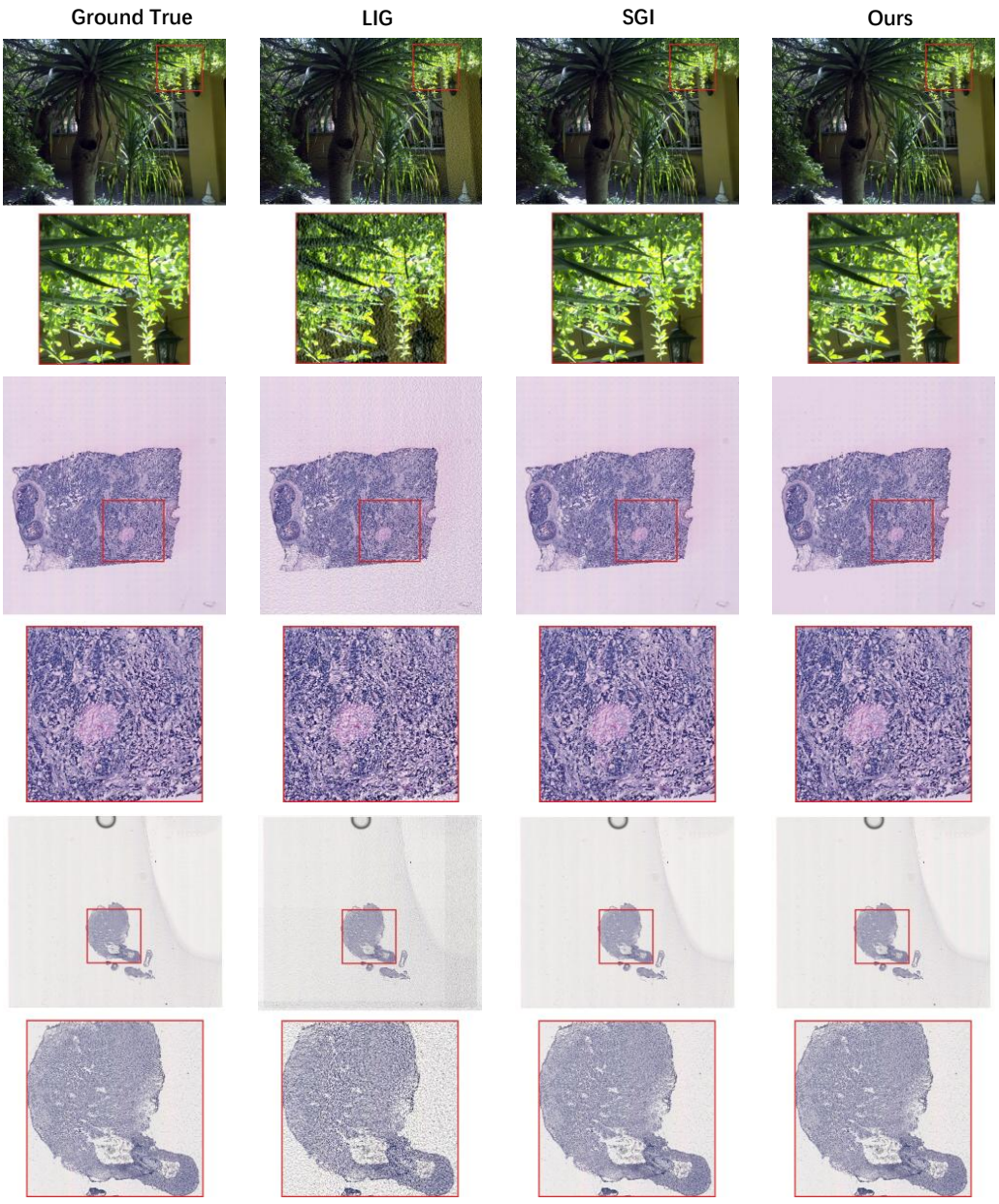


Figure 15: More qualitative reconstruction results of CaliSplat with other baselines.

Table 4: Main notations used in CaliSplat.

Notation	Description
$I, \widehat{I}$	Input and reconstructed RGB images.
S	Self-contained compressed bitstream.
b	Spatial image block.
r_b	Side length of block b.
K	Number of Gaussians used by a candidate representation.
k	Index of a Gaussian within a candidate.
$\theta_{b,k}$	Parameters of the k-th Gaussian in block b.
$\Theta_{b,K}$	K-Gaussian candidate representation for block b.
$\mathcal{R}(\cdot)$	Deterministic local Gaussian rasterization operator.
$D_b(1)$	Analytic $K = 1$ reconstruction error used as a block-difficulty signal.
$\mathcal{A}_K$	Set of blocks admitted for fitting a K-Gaussian candidate.
$C^{(0)}$	Closed-form color initialization for a Gaussian candidate.
T_c	Number of optimization steps used for cheap candidate fitting.
$\Theta_{b,K}^{\mathrm{cheap}}$	Gaussian parameters obtained from short-horizon candidate fitting.
$D_b^{\mathrm{cheap}}(K)$	Reconstruction error of the cheap-fitted candidate.
$D_b^{\star}(K)$	Reconstruction error associated with sufficiently long optimization.
$\varepsilon_{b,K}$	Residual optimization error caused by early stopping.
T_t	Number of optimization steps used for sampled teacher fitting.
$D_b^{\mathrm{teach}}(K)$	Longer-horizon reconstruction error produced by a sampled teacher fit.
$\mathbf{x}_{b,K}$	Feature vector used for per-K score calibration.
$\widetilde{D}_b(K)$	Calibrated candidate score used for global Gaussian allocation.
B	Global Gaussian budget.
$\mathcal{T}$	Selected block structure under the global budget.
K_b	Number of Gaussians assigned to selected block b.
λ	Cost assigned to each Gaussian during budget allocation.
$\mathbf{z}_{b,k}$	Coding-space geometry representation of a selected Gaussian.
$\mathbf{z}_{b,k}^{(0)}$	Decoder-known geometry of the deterministic template.
$\Delta\mathbf{z}_{b,k}$	Template-relative geometry residual, $\mathbf{z}_{b,k} - \mathbf{z}_{b,k}^{(0)}$.
$\widehat{\Theta}$	Explicit Gaussian representation recovered by the decoder.