SGI: Structured 2D Gaussians for Efficient and Compact Large Image Representation

Zixuan Pan[1]*, Kaiyuan Tang[1]*, Jun Xia[1], Yifan Qin[1],
Lin Gu[2], Chaoli Wang[1], Jianxu Chen[3], Yiyu Shi[1]
[1]University of Notre Dame [2]Tohoku University
[3]Leibniz-Institut für Analytische Wissenschaften – ISAS – e.V.

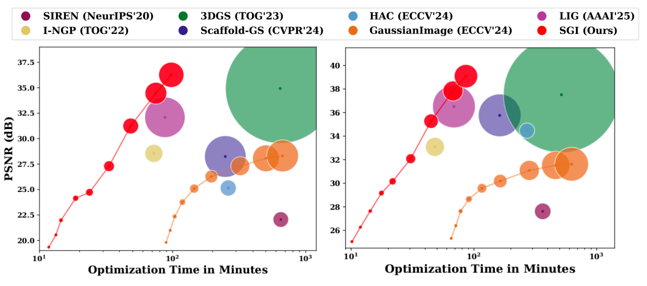
Abstract

2D Gaussian Splatting has emerged as a novel image representation technique that can support efficient rendering on low-end devices. However, scaling to high-resolution images requires optimizing and storing millions of unstructured Gaussian primitives independently, leading to slow convergence and redundant parameters. To address this, we propose structured Gaussian image (SGI), a compact and efficient framework for representing high-resolution images. SGI decomposes a complex image into multi-scale local spaces defined by a set of seeds. Each seed corresponds to a spatially coherent region and, together with lightweight multi-layer perceptrons (MLPs), generates structured implicit 2D neural Gaussians. This seed-based formulation imposes structural regularity on otherwise unstructured Gaussian primitives, which facilitates entropy-based compression at the seed level to reduce the total storage. However, optimizing seed parameters directly on high-resolution images is a challenging and non-trivial task. Therefore, we designed a multi-scale fitting strategy that refines the seed representation in a coarse-to-fine manner, substantially accelerating convergence. Quantitative and qualitative evaluations demonstrate that SGI achieves up to 7.5× compression over prior non-quantized 2D Gaussian methods and 1.6× over quantized ones, while also delivering 1.6× and 6.5× faster optimization, respectively, without degrading, and often improving, image fidelity. Code is available at https://github.com/zx-pan/SGI.

1. Introduction

Image representation is a fundamental problem in computer vision, with wide-ranging applications such as compression [19], editing [52], and super-resolution [49]. Traditional image representations, such as grid graphics or transform-based methods (e.g., discrete cosine transform [2] or wavelet transform [21]), are often limited in their ability to model visual signals with high fidelity. Recent work has explored implicit neural representations (INRs) [38, 41], which encode images using multi-layer perceptrons (MLPs) that learn the mapping from pixel coordinates to pixel values. Although INRs offer continuous, resolution-independent modeling, representing large, high-resolution images with INRs remains challenging, as faithfully capturing fine spatial details typically requires deep MLPs. This results in significant computation and memory overhead when processing millions of pixels, leading to slow encoding/decoding and limited feasibility on low-end devices. To address this issue, recent methods based on 2D Gaussian Splatting [54, 57] represent images as sets of explicit Gaussian primitives. This formulation enables fast encoding and decoding without requiring slow network inference. Moreover, the memory-efficient splatting rasterization enables high-resolution optimization even on resource-constrained devices.

Despite the efficiency of 2D Gaussian splatting, existing 2D Gaussian representation methods [54, 57] optimize each



(a) FGF2 (b) ICB

Figure 1. Image representation results on the FGF2 [28] and ICB [35] datasets. The x-axis (log scale) denotes optimization time in minutes, and the y-axis shows PSNR (dB). Each point represents a specific model configuration, with the area of the circle indicating its storage size. For GaussianImage and our SGI, we plot performance curves obtained by varying the number of Gaussian primitives. Our SGI consistently achieves a favorable trade-off between fidelity, compactness, and optimization time.

*Equal contribution.

Gaussian independently without leveraging spatial locality, the property that nearby pixels tend to share similar colors, textures, and structures, resulting in significant parameter redundancy across adjacent primitives. This issue becomes more severe in high-resolution settings, where millions of Gaussians are required to cover the spatial content, leading to (1) *large model sizes* due to storing vast numbers of primitives, and (2) *substantial optimization overhead*, especially when quantization-aware fine-tuning is needed for compression [54].

To address these limitations, we present structured Gaussian image (SGI), a compact representation tailored for high-resolution image modeling. SGI partitions the image into a collection of multi-scale local regions, each associated with a seed point. For every seed, a pair of lightweight MLPs predicts the attributes of the Gaussian primitives within its region, converting the formerly unstructured set of Gaussians into a coherent, seed-organized representation that preserves local details. The structural regularity introduced by seed-based 2D neural Gaussians allows us to further compress residual spatial redundancy at the seed level. Specifically, a context model composed of a binary hash grid estimates explicit distributions of seed attributes, enabling adaptive bit allocation and effectively compressing residual spatial redundancy. While seed-based compression alleviates the model-size issue, directly optimizing seed parameters at full resolution, however, is difficult and computationally costly. The adaptive bit allocation also further exacerbates the challenge of long optimization time. To mitigate this, we adopt a coarse-to-fine multi-scale fitting strategy, where SGI is first optimized on a low-resolution approximation and then progressively refined at higher resolutions. This hierarchical optimization significantly improves both convergence speed and stability.

As shown in Figure 1, SGI provides consistently better trade-offs between fidelity, compactness, and optimization efficiency compared to prior 2D Gaussian and INR-based methods. Our contributions are summarized as follows:

- We propose the first structured 2D Gaussian representation for high-resolution images by introducing seed-based 2D neural Gaussians and a context-guided entropy coding scheme, enabling effective elimination of spatial redundancy and significant model size reduction.
- We develop a multi-scale fitting strategy that enables coarse-to-fine optimization, substantially reducing optimization time while improving the reconstruction quality.
- Extensive experiments on megapixel-scale datasets show that SGI achieves up to 7.5× compression over non-quantized 2D Gaussian baseline methods and 1.6× over quantized ones, while providing 1.6× to 6.5× faster optimization and maintaining or improving image fidelity.

2. Related work

Implicit Neural Representation. By utilizing a neural network to fit continuous mappings from coordinates to signal values, INR [38, 41] has demonstrated its effectiveness in various types of signal representations, including 3D scenes [3, 4, 32], 3D data [20, 40, 44], 2D images [12, 16, 17], and videos [9, 10, 56]. For image representation, vanilla INRs [41] typically optimize an MLP to approximate the pixel-wise RGB values. However, for high-resolution images, such INRs must repeatedly forward pass the MLP numerous times to reconstruct the full image, which is computationally expensive. Recent grid-based INRs [6, 23, 33] move most network parameters to a parameteric grid, allowing efficient spatial information encoding and sampling. Nevertheless, as the grid resolution increases, the number of parameters in grid-based INRs grows rapidly, leading to substantial GPU memory consumption. Although some previous works introduce hierarchical structures [31, 37] to achieve efficient encoding, they still suffer from the slow feedforward process of the MLP network. To further improve efficiency, recent work has shifted from neural fields to point-based representations.

Differentiable Point-based Representation. Differentiable point-based representations [18, 24, 25, 27, 45, 46, 50, 54, 55] have recently been widely studied for their ability to efficiently and flexibly capture intricate structural details. Since Kerbl et al. [27] demonstrated the capability of 3D Gaussian representation in scene reconstruction, point-based representation has emerged as a trending topic. Extensive efforts have been made to extend the original 3DGS algorithm for image representation. GaussianImage [54] adapts Gaussian points to image space with fewer attribute parameters and optimizes weighted color attributes to approximate the alpha blending results, thereby reducing computation complexity. After quantization-aware optimization, the compact 2D Gaussian representation can achieve high-fidelity image representation with remarkable decoding speed. Following GaussianImage, other works have explored the applications of 2D Gaussian representation to realize arbitrary-scale image super-resolution [8, 22, 34], realistic image editing [47], and video representation [5, 29, 42]. Image-GS [55] proposes a content-adaptive image representation for better primitive allocation. In the context of large image representation, LIG [57] introduces a Level-of-Gaussian hierarchy to improve high-frequency fitting by sequentially refining residuals with additional Gaussians.

While achieving promising performance, these methods have limitations in exploiting spatial redundancy. Recent anchor-based frameworks [15, 30, 53] have demonstrated remarkable efficiency gains in 3D scene representations to reduce this redundancy. However, we empirically found that directly applying these 3D anchor representations to 2D image modeling does not yield comparable compression

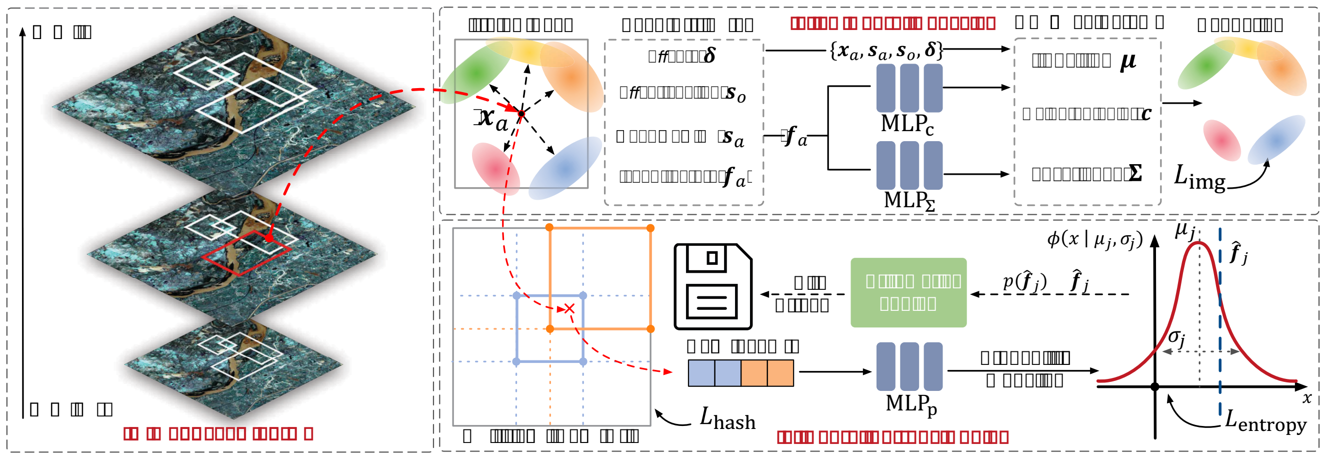


Figure 2. **The overall pipeline of SGI.** We first introduce **(a) seed-based 2D neural Gaussians**, where each seed predicts a group of 2D Gaussian primitives via two shared MLPs for decoding color and covariance. To accelerate optimization, we adopt a **(b) multi-scale fitting** strategy that progressively refines the representation from coarse to fine using a Gaussian pyramid. Finally, we leverage **(c) neural entropy coding** to further compress the explicit seed attributes for compact representation.

benefits. In contrast, we design 2D seed points to provide structural regularity for our *seed-based 2D neural Gaussians*, thereby establishing a compact, organized representation and enabling advanced neural entropy coding. By integrating a context model [13, 14, 48], our SGI encodes seed parameters based on their information entropy, leading to a significantly more compact image representation compared to other 2D Gaussian-based approaches. Unlike the Level-of-Gaussian strategy in LIG [57], which uses a subset of Gaussians solely for residual fitting, our *multi-scale fitting* strategy employs the full representation in a hierarchical manner, ensuring that optimization acceleration and representation compression do not come at the cost of reconstruction fidelity. Compared to 3DGS-based methods such as Scaffold-GS [30] and HAC [14], SGI achieves significantly faster optimization and lower GPU memory usage, owing to its use of *seed-based 2D neural Gaussians* and context model-driven *entropy coding*. In addition, the proposed *multi-scale fitting* strategy mitigates the extended optimization time typically due to the large number of parameters and the overhead of entropy estimation [13, 14, 48]. Further comparisons are provided in Section 4.2.

3. Methodology

Figure 2 shows the pipeline of SGI. Our objective is to encode the high-resolution image with a set of seed-based 2D neural Gaussians (Section 3.1). The structural regularity introduced by the seed-based representation allows us to extract a compact representation using neural entropy coding (Section 3.2). These seed attributes are optimized using a multi-scale strategy to improve convergence speed (Section 3.3). Finally, we describe how the seed parameters are encoded and decoded using entropy coding (Section 3.4).

3.1. Seed-based 2D Neural Gaussians

While 2D Gaussian representations have shown strong potential for image representation [54, 57], existing methods treat each Gaussian independently, thereby overlooking spatial locality among Gaussian primitives and leading to significant parameter redundancy. To address this limitation, we try to introduce *seed-based 2D neural Gaussians* into the 2D image representation domain for the first time, drawing inspiration from recent progress in *anchor-centered* methods for 3D reconstruction [15, 30, 53]. This approach aims to exploit the spatial locality by grouping Gaussians under a set of seeds and predicting their attributes through lightweight MLPs, rather than storing them directly. We introduce the detailed implementation of *seed-based 2D neural Gaussians* below.

Given a predefined number of seeds N, we initialize them to uniformly cover the image. Each seed located at position $\boldsymbol{x_a} \in \mathbb{R}^2$ is associated with a set of attributes:

$$\mathcal{A} = \left\{ \boldsymbol{f_a} \in \mathbb{R}^D,\ \boldsymbol{s_o} \in \mathbb{R}^2,\ \boldsymbol{s_a} \in \mathbb{R}^2,\ \boldsymbol{\delta} \in \mathbb{R}^{K\times 2} \right\}, \quad (1)$$

where $\boldsymbol{f_a}$ is the seed feature, $\boldsymbol{\delta}$ contains learned offsets for K associated Gaussians, $\boldsymbol{s_o}$ and $\boldsymbol{s_a}$ are per-seed scaling factors for offsets and scalings of the associated Gaussians. The positions of these Gaussians are computed as:

$$\left\{ \boldsymbol{\mu}^{(k)} \right\}_{k=0}^{K-1} = \boldsymbol{x_a} + \left\{ \boldsymbol{\delta}^{(k)} \right\}_{k=0}^{K-1} \cdot \boldsymbol{s_o}. \quad (2)$$

We then decode the opacity-weighted color coefficients $\mathbf{c}' \in \mathbb{R}^3$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{2\times 2}$ of each Gaussian from $\boldsymbol{f_a}$ using two MLPs: MLP$_c$ outputs the colors, and MLP$_\Sigma$ predicts base scales s_{base} and rotation angles θ. The final scale $\boldsymbol{s} \in \mathbb{R}^2$ of each Gaussian used for rendering is computed by $\boldsymbol{s} = s_{\text{base}} \cdot \boldsymbol{s_a}$. The covariance matrix $\boldsymbol{\Sigma}$ is obtained by constructing a positive semidefinite matrix via:

$$\mathbf{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^\top\mathbf{R}^\top, \tag{3}$$

$$\mathbf{R}(\theta) = \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2\end{bmatrix}. \tag{4}$$

Finally, the rendered pixel color is calculated by using all Gaussians contribute to the pixel via an accumulated summation [54]:

$$\boldsymbol{C} = \sum_{i\in I} \mathbf{c}'_i G_i(\mathbf{x}), \tag{5}$$

where I is the number of Gaussians contributing to the pixel, $\mathbf{c}'_i$ is an opacity-weighted color vector, and $G_i(\mathbf{x})$ is the spatial density of the i-th Gaussian at pixel $\mathbf{x}$, defined as:

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right). \tag{6}$$

This formulation enables fully differentiable rasterization. By adopting and extending differentiable rasterization techniques from prior work [27, 54], our *seed-based 2D neural Gaussians* achieves fast rendering and low memory consumption. Overall, the proposed *seed-based 2D neural Gaussians* models an image using N seed attributes $\{\mathcal{A}^i\}_0^{i=N-1}$ and two lightweight MLPs MLP$_c$ and MLP$_\Sigma$, which together predict 2D Gaussian primitives each defined by only three attributes (position, covariance, and weighted color), requiring just eight parameters per Gaussian.

3.2. Entropy Coding with Context Model

Theoretically, converting unstructured 2D Gaussians into seed-level attributes and shared MLPs should reduce the overall model size, similar to how Scaffold-GS [30] improves the storage efficiency of 3DGS [27]. However, this adaptation is far less straightforward in the 2D setting. As shown in Table 1 and Table 4, directly applying seed-based 2D neural Gaussians yields only about a 3% reduction compared to pure 2D Gaussian representations (e.g., LIG [57]). This limited gain arises because 2D Gaussian formulations already eliminate several parameters, such as opacity, that dominate storage in 3D counterparts. Nevertheless, the structural regularity introduced by seed-based neural Gaussians provides an organized representation that enables further compression by modeling and constraining the distribution of seed attributes, allowing fewer bits to be used during entropy coding. Specifically, given quantized codes $\hat{y}$ and a probability model $p_{\hat{y}}(\hat{y})$, entropy coding techniques, such as arithmetic coding [36], can losslessly compress them. Thus, to compress the seed attributes via entropy coding, we need quantization to discretize the data and probability modeling to estimate symbol likelihoods.

Quantization. Let $\boldsymbol{f}_j^{(i)}$ denote the j-th component of i-th seed's attributes $\mathcal{A}^{(i)}$. Following [14, 43], quantization is implemented using noise injection during training and rounding during testing:

$$\hat{\boldsymbol{f}}_j^{(i)} = \boldsymbol{f}_j^{(i)} + \mathcal{U}\left(-\tfrac{1}{2}, \tfrac{1}{2}\right)\cdot q_j^{(i)}, \quad \text{for training} \tag{7}$$

$$= \text{Round}(\boldsymbol{f}_j^{(i)}/q_j^{(i)})\cdot q_j^{(i)}. \quad \text{for testing} \tag{8}$$

The quantization step size $q_j^{(i)}$ is computed as:

$$q_j^{(i)} = Q_j \times \left(1 + \tanh(r_j^{(i)})\right), \tag{9}$$

where Q_j is the quantization step size, $r_j^{(i)}$ is a refinement factor predicted by the context model in Eq. (10).

Probability Modeling. Inspired by recent 3DGS work [13, 14, 48], we employ a context model MLP$_p$ to estimate the distributions of seed attributes. Rather than directly conditioning on $\boldsymbol{f}^{(i)} \in \mathcal{A}^{(i)}$, we introduce a learnable binary hash grid $\mathcal{H}$ [39] to capture the inherent spatial consistencies of the unorganized seeds for the context model:

$$\left\{\mu_j^{(i)}, \sigma_j^{(i)}, r_j^{(i)}\right\}_{j=0}^{3} = \text{MLP}_p(\mathcal{H}(\boldsymbol{x}_a^{(i)})), \tag{10}$$

$$p(\hat{\boldsymbol{f}}_j^{(i)}) = \int_{\hat{\boldsymbol{f}}_j^{(i)}-\frac{q_j}{2}}^{\hat{\boldsymbol{f}}_j^{(i)}+\frac{q_j}{2}} \phi(x \mid \mu_j^{(i)}, \sigma_j^{(i)})\, dx, \tag{11}$$

where ϕ is the probability density function of the Gaussian distribution, $j \in \{0,1,2,3\}$, $\hat{\boldsymbol{f}}_j^{(i)}$ is the quantized seed attribute for $\boldsymbol{f}_j^{(i)}$, and $\mu_j^{(i)}$ and $\sigma_j^{(i)}$ are the predicted mean and standard deviation for each attribute dimension. The context model and hash grid are trained by minimizing the entropy loss:

$$L_{\text{entropy}} = \sum_{i=0}^{N-1}\left[-\log_2\left[\prod_{\boldsymbol{f}^{(i)}\in\{\mathcal{A}^{(i)}\}} p(\hat{\boldsymbol{f}}^{(i)})\right]\right]. \tag{12}$$

The hash grid is binarized to $\{-1,+1\}$ and optimized via straight-through estimation (STE) [39], and its bit consumption is bounded by:

$$L_{\text{hash}} = -n_1\log_2\left(\frac{n_1}{n_1+n_0}\right) - n_0\log_2\left(\frac{n_0}{n_1+n_0}\right), \tag{13}$$

where n_1 and n_0 are total counts of $+1$ and -1 in the hash grid, respectively.

Overall Loss. The total optimization loss combines the rendering fidelity objective and bits consumption regularization:

$$L = L_{\text{img}} + \frac{\lambda}{N\cdot d_{\mathcal{A}}}(L_{\text{entropy}} + L_{\text{hash}}), \tag{14}$$

where L_{img} is the L_1 loss between rendered and target images, λ is a hyperparameter balancing the rate and fidelity, and $d_{\mathcal{A}} = D + 4 + 2K$ is the total number of attribute dimensions for each seed.

Table 1. **Quantitative results on FGF2 and ICB.** Optimization time is in minutes, and model size is in megabytes (MB). To fit INR-based methods (SIREN and I-NGP) on these large-scale datasets under limited GPU memory, we adopt a patch-based training strategy that loads randomly sampled coordinate and pixel batches from CPU to GPU on the fly. Our SGI is evaluated in two settings: **low-rate** (3.5M Gaussians) and **high-rate** (10M Gaussians), both demonstrating strong trade-offs between fidelity, compactness, and optimization efficiency. Darker blue and darker red highlights indicate the best-performing method within the low-rate and high-rate groups, respectively, in each metric. *Reported numbers are averaged per image across the dataset.*

Method	FGF2					ICB				
	PSNR↑	SSIM↑	LPIPS↓	Opt. Time↓	Size↓	PSNR↑	SSIM↑	LPIPS↓	Opt. Time↓	Size↓
SIREN [41] (NeurIPS'20)	22.05	0.8020	0.4831	649.71	15.79	27.62	0.8600	0.3817	363.34	15.79
I-NGP [33] (TOG'22)	28.55	0.9592	0.1043	72.32	21.07	33.09	0.9532	0.1176	48.11	21.07
HAC [14] (ECCV'24)	25.15	0.9226	0.1767	261.69	16.78	34.47	0.9801	0.0688	270.57	13.52
GaussianImage [54] (ECCV'24)	27.30	0.9457	0.1342	322.17	23.37	31.09	0.9330	0.1462	282.61	23.37
Our SGI (low-rate)	31.24	0.9863	0.0731	48.43	16.33	35.27	0.9853	0.0575	44.75	12.30
3DGS [27] (TOG'23)	34.93	0.9950	0.0246	642.85	787.73	37.52	0.9932	0.0248	515.99	787.73
Scaffold-GS [30] (CVPR'24)	28.25	0.9578	0.0973	248.83	112.61	35.76	0.9853	0.0509	162.11	105.81
LIG [57] (AAAI'25)	32.10	0.9879	0.0568	87.56	106.81	36.40	0.9888	0.0180	68.73	106.81
Our SGI (high-rate)	36.27	0.9961	0.0162	97.75	41.74	39.09	0.9949	0.0122	86.11	32.15

3.3. Multi-scale Fitting

Training on large, high-resolution images can be computationally intensive, even for Gaussian-based models. Besides, the quantization-aware training and probability modeling for the entropy coding introduce extra computation overhead for training. To accelerate optimization, we propose a *multi-scale fitting* strategy that gradually refines the representation by using the solution from a coarser scale as a warm start for the next finer scale. Given a target image I, we construct a Gaussian pyramid [1] with M levels, producing a sequence of downsampled images $\{I_0 = I, I_1, \ldots, I_{M-1}\}$ from fine to coarse, each downsampled by a factor of two. At each level l, we maintain a set of seed positions and attributes $\mathbb{A}^{(l)} = \{\boldsymbol{x_a}^{(i,l)}, \mathcal{A}^{(i,l)}\}_0^{i=N-1}$ and parameters $\theta^{(l)}$ of MLP$_c$ and MLP$_\Sigma$. Starting from the coarsest level $l = M-1$, we optimize:

$$\mathbb{A}_*^{(l)}, \theta_*^{(l)} = \arg\min_{\mathbb{A}^{(l)}, \theta^{(l)}} L_{\text{img}}(I_l, \hat{I}_l(\mathcal{G}^{(l)}(\mathbb{A}^{(l)}, \theta^{(l)}))), \tag{15}$$

where $\mathcal{G}^{(l)}(\mathbb{A}^{(l)}, \theta^{(l)})$ are the Gaussians inferred from seed parameters and MLPs, $\hat{I}_l$ is the rendered image at level l. The optimized parameters are transferred to the next finer level with adaptation:

$$\mathbb{A}^{(l-1)} \leftarrow \phi_{\text{init}}(\mathbb{A}_*^{(l)}), \quad \theta^{(l-1)} \leftarrow \theta_*^{(l)}, \tag{16}$$

where $\phi_{\text{init}}(\cdot)$ adapts seed attributes for the higher-resolution image. In particular, seed positions and scales are scaled by a factor of two to account for resolution doubling. This hierarchical fitting process continues iteratively until the finest level $l = 0$ is reached.

3.4. Bitstream Generation and Decoding

The encoding and decoding process involves the seed positions and attributes $\{\boldsymbol{x}_a^{(i)}, \mathcal{A}^{(i)}\}_{i=0}^{N-1}$, as well as the binary hash grid $\mathcal{H}$. The seed positions are compressed using geometric point cloud compression (GPCC) [7], while the hash grid is encoded using arithmetic coding. The seed attributes $\{\mathcal{A}^{(i)}\}_{i=0}^{N-1}$ are also entropy-coded via arithmetic coding, with probabilities for each component estimated by the context model defined in Eq. (10) and Eq. (11). In the end, we store the encoded seed parameters $\{\boldsymbol{x}_a^{(i)}, \mathcal{A}^{(i)}\}_{i=0}^{N-1}$, the binary hash grid $\mathcal{H}$, two MLPs (MLP$_c$ and MLP$_\Sigma$) used for decoding neural Gaussians, and the context model MLP$_p$.

At decoding time, the seed positions and hash grid are first decoded. The context model (with hash-based spatial features) is then used to reconstruct the probability distribution for each attribute component, which guides the arithmetic decoder in recovering the quantized seed attributes. These are subsequently passed through MLPs to reconstruct the full set of Gaussian parameters for rendering.

4. Experiments

4.1. Experiment Setup

Datasets, Metrics, and Baselines. We evaluate our method on three real-world datasets with high-resolution images spanning diverse domains, including natural, satellite, and biomedical images. The first contains four satellite images, each with approximately 51 megapixels (MP), from the Full-resolution Gaofen-2 (FGF2) dataset [28]. The second dataset includes two natural images from the Image Compression Benchmark (ICB) [35], with resolutions of 27.7 MP and 39.1 MP, and the biomedical dataset (STimage [11]) consists of three histopathology images averaging 76 MP. We compare our method with a wide range of existing image representation techniques, including both INR and Gaussian-based methods. Specifically, we include SIREN [41] and Instant-NGP (I-NGP) [33] as representative INR baselines, and compare against 3D Gaussian

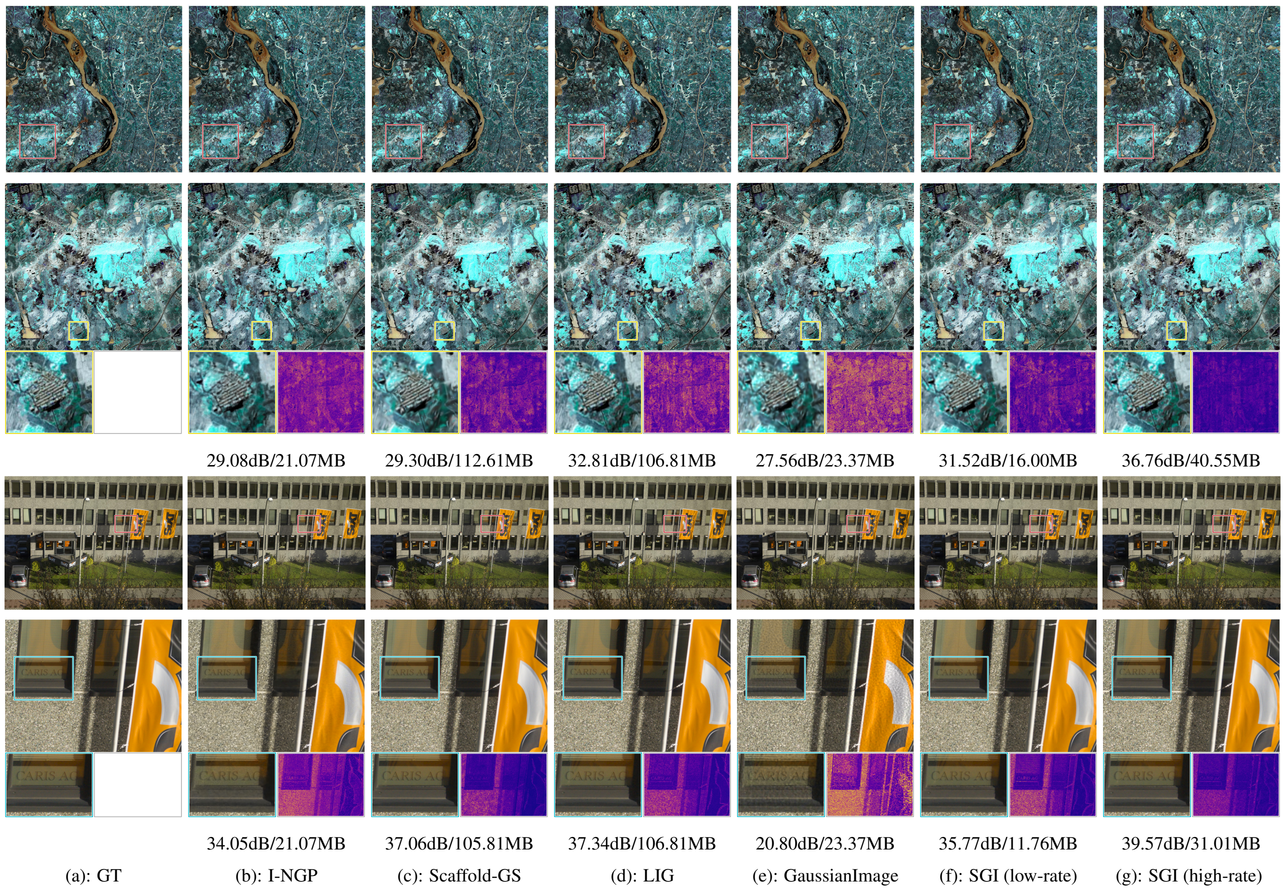


Figure 3. **Visual comparisons on the FGF2 and ICB (w/ zoom-in cases and error maps).** Qualitative results on representative examples from FGF2 (top) and ICB (bottom), comparing SGI with I-NGP [33], Scaffold-GS [30], LIG [57], and GaussianImage [54]. Zoom-in regions highlight perceptual differences. In the third row of each block, we visualize the per-pixel reconstruction error using heatmaps, where warmer colors (e.g., yellow) indicate higher deviation from the ground truth. PSNR (dB) and storage size (MB) for each method are shown below the visualizations.

methods (3DGS [27], Scaffold-GS [30], and HAC [14]) as well as 2D Gaussian approaches (LIG [57], GaussianImage [54]). To assess reconstruction quality, we report peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and learned perceptual image patch similarity (LPIPS). We also measure storage cost in MB and optimization time in minutes.

Implementation Details. Our SGI implementation builds on the 2D Gaussian CUDA kernels [54, 57] and the open source library `gsplat` [51]. All our models are trained and tested on NVIDIA A10 GPUs (24 GB memory). All Gaussian-based methods, including our SGI under the low-rate setting, use 3.5 million Gaussians. We also report results for a high-rate version of SGI that uses 10 million Gaussians. Due to the substantial memory consumption of INR-based methods, we adopt random sampling of image coordinates and signal values during training, with a batch size of 32,000. SGI is optimized using Adam for 15,000 steps, with loss weight $\lambda = 0.001$ and pyramid

Table 2. **Quantitative results on STimage.** Optimization time is in minutes, and model size is in megabytes (MB). To fit INR-based methods (SIREN and I-NGP) on these large-scale datasets under limited GPU memory, we adopt a patch-based training strategy that loads randomly sampled coordinate and pixel batches from CPU to GPU on the fly. Our SGI is evaluated in two settings: **low-rate** (3.5M Gaussians) and **high-rate** (10M Gaussians), both demonstrating strong trade-offs between fidelity, compactness, and optimization efficiency. Darker blue and darker red highlights indicate the best-performing method within the low-rate and high-rate groups, respectively, in each metric. *Reported numbers are averaged per image across the dataset.*

Method	PSNR↑	SSIM↑	LPIPS↓	Opt. Time↓	Size↓
SIREN [41] (NeurIPS'20)	28.09	0.8843	0.3654	983.48	15.79
I-NGP [33] (TOG'22)	35.57	0.9763	0.0518	129.96	21.07
GaussianImage [54] (ECCV'24)	25.75	0.6404	0.4006	407.70	23.37
Our SGI (low-rate)	33.96	0.9743	0.1196	103.43	10.05
3DGS [27] (TOG'23)	38.51	0.9957	0.0192	723.43	787.73
LIG [57] (AAAI'25)	34.33	0.9714	0.1386	106.82	106.81
Our SGI (high-rate)	38.72	0.9935	0.0208	136.26	22.03

Table 3. **Ablation studies on # of Gaussians K in each seed.**

K	FGF2					ICB				
	PSNR↑	SSIM↑	LPIPS↓	Opt. Time (Min)↓	Size (MB)↓	PSNR↑	SSIM↑	LPIPS↓	Opt. Time (Min)↓	Size (MB)↓
5	31.29	0.9861	0.0698	44.27	18.48	35.03	0.9841	0.0626	44.70	13.64
10	31.24	0.9863	0.0731	48.43	16.33	35.27	0.9853	0.0575	44.75	12.30
15	30.61	0.9836	0.0828	55.33	15.32	34.88	0.9842	0.0599	50.60	11.48
20	30.62	0.9834	0.0809	56.02	14.83	34.57	0.9824	0.0668	53.37	10.87

level $M = 3$. Other hyperparameters follow the official implementations of the respective baselines unless otherwise specified. *More implementation details are in the Appendix.*

4.2. Comparison with Baselines

As shown in Table 1, when evaluated on FGF2 and ICB, SGI significant reduces storage size compared to both INR and Gaussian-based baselines, while maintaining or surpassing image fidelity in PSNR, SSIM, and LPIPS. In the low-rate setting (3.5M Gaussians), SGI achieves a strong trade-off between accuracy and compactness, outperforming the prior 2D Gaussians baseline GaussianImage by a large margin on both datasets. In the high-rate configuration (10M Gaussians), SGI achieves the best overall fidelity across all metrics, outperforming the second-best 3DGS with much lower storage cost. In terms of optimization efficiency, INR method SIREN suffer from slow convergence due to patch-wise optimization imposed by limited GPU memory, while 3DGS, Scaffold-GS and HAC incur significantly longer optimization times due to the complexity of 3D Gaussian rendering. Compared to the next-fastest I-NGP, our SGI achieves both higher fidelity and lower storage cost. This optimization efficiency is particularly important, as image representation requires optimization a separate model for each image. Figure 3 further demonstrates that SGI better preserves fine-grained textures and high-frequency details compared to existing baselines.

The results on biomedical dataset STimage are presented in Table 2. Scaffold-GS leverages 3D Gaussian primitives to represent images and optimize MLPs during training, resulting in out-of-memory (OOM) issues when facing the STimage dataset, which has a higher resolution than FGF2 dataset. Therefore, we exclude Scaffold-GS [30] from this evaluation. In the low-rate setting, SGI achieves competitive or superior fidelity compared to other baselines while using the least optimization time and the smallest model size. In the high-rate setting, SGI obtains the highest PSNR across all methods, demonstrating that increased Gaussian capacity leads to further fidelity improvement while maintaining a favorable balance between efficiency and compactness. We show the qualitative comparisons on STimage in the Appendix due to page limit.

Table 4. **Ablation studies on λ in L.**

λ	FGF2		ICB	
	PSNR↑	Size (MB)↓	PSNR↑	Size (MB)↓
0	32.36	104.08	36.13	102.35
0.0005	31.30	18.93	35.48	14.90
0.001	31.24	16.33	35.27	12.30
0.003	30.21	11.93	33.51	8.34

Table 5. **Ablation studies on multi-scale fitting.**

M	FGF2		ICB	
	PSNR↑	Opt. Time (Min)↓	PSNR↑	Opt. Time (Min)↓
1	30.58	71.59	35.02	70.63
2	30.60	55.96	35.06	49.90
3	31.24	48.43	35.27	44.75
4	30.56	42.27	33.78	37.67

Table 6. **The storage size (MB) of each component** of our method evaluated on the FGF2 dataset.

Number of Gaussians	$\boldsymbol{x_a}$	$\boldsymbol{f_a}$	$\{\boldsymbol{s_o}, \boldsymbol{s_a}\}$	$\boldsymbol{\delta}$	$\mathcal{H}$	MLPs	Total Size
3.5e6	0.9995	7.8689	1.5427	5.7646	0.0293	0.1273	16.3324
7.0e6	1.8360	15.5719	2.4960	11.0664	0.0292	0.1273	31.1268
1.0e7	2.4442	20.5720	3.3549	15.2157	0.0298	0.1273	41.7439

4.3. Ablation Studies and Discussions

Evaluation of the Number of Gaussians per Seed. To assess the effectiveness of our *seed-based neural Gaussians*, we evaluate SGI on FGF2 and ICB by varying the number of Gaussians K assigned to each seed, as shown in Table 3. To maintain a consistent total number of Gaussians, we adjust the number of seeds accordingly. The dimension D of the seed feature $\boldsymbol{f_a}$ and the latent size of the MLPs are also adapted based on K. As observed in the table, larger K values lead to smaller overall model sizes, as more Gaussian attributes are compactly encoded within each seed's feature. However, when K becomes too large, a slight drop in fidelity and an increase in optimization time are observed due to the larger MLP size and seed features. We choose $K = 10$ in this paper as a trade-off between representation quality and model compactness.

Ablation on the Hyperparameter λ in L. We evaluate the impact of the rate-distortion trade-off parameter λ in the loss function L in Table 4. When $\lambda = 0$, the entropy coding is completely deprecated, leaving only the seed-based 2D neural Gaussians. Notably, using seed-based 2D neural Gaussians alone yields less redundancy reduction than in 3D cases, highlighting the importance of our entropy mod-

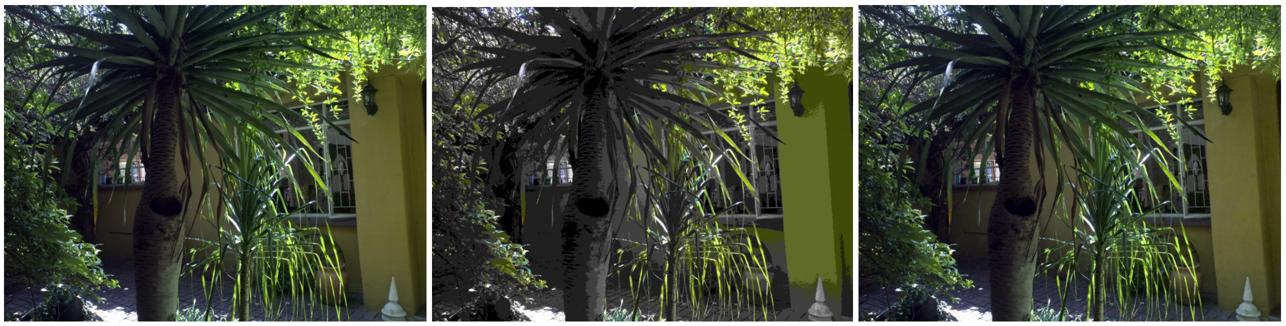


Figure 4. **Visual comparison with traditional image codec method JPEG on the ICB dataset at low Bpp.** We report the bit per pixel (bpp) and PSNR (dB) for each method below the visualizations.

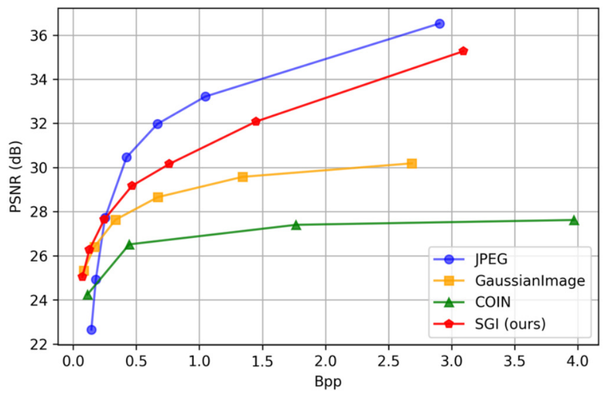


Figure 5. **Rate-distortion curves** of our approach and different compression baselines on ICB dataset in PSNR.

eling in achieving effective 2D Gaussian compression. As λ increases, the storage size reduces due to improved entropy modeling, but overly large values may slightly degrade reconstruction quality. We empirically set $\lambda = 0.001$.

Ablation on Multi-scale Fitting. Table 5 shows the optimization efficiency and representation quality of SGI under different numbers of Gaussian pyramid levels. We exclude the model size metric here as *multi-scale fitting* theoretically will not affect the final model size. With the total number of optimization steps fixed, we report the actual optimization time. The results show that our *multi-scale fitting* significantly accelerates optimization while maintaining or even improving image fidelity. When $M = 4$, we observe a slight PSNR drop, as the optimization budget is distributed across more levels, leaving fewer steps for the final high-resolution stage. We thus use $M = 3$ in main experiments.

Component-wise Storage Analysis. Table 6 breaks down the storage cost of each component in SGI under different numbers of Gaussians. The hash grid $\mathcal{H}$ is stored in a compact binary format, while the MLPs are lightweight by design, contributing only a small number of parameters. Together, they impose minimal storage overhead and scale well with increasing Gaussians, enabling efficient high-resolution image representation.

Comparisons with Image Compression Baselines. Figure 5 presents the rate-distortion (RD) curves of various codecs on the ICB dataset. Our method significantly outperforms the 2DGS-based method GaussianImage and the INR-based method COIN by a large margin. It achieves competitive performance across the entire bit-rate range and outperforms the conventional JPEG codec in the low bit-rate regime. As illustrated in Figure 4, JPEG suffers from severe color shifts and visual artifacts at low bitrates, resulting in noticeably degraded PSNR. In contrast, our SGI method produces visually faithful reconstructions with superior PSNR while using fewer bits, demonstrating the potential of structured 2D Gaussian representations for next-generation image compression.

5. Conclusion

In this work, we present SGI, an efficient and compact framework for high-resolution image representation using structured 2D Gaussians. By organizing unstructured Gaussian primitives under seeds and decoding their attributes through shared lightweight MLPs, our *seed-based 2D neural Gaussians* establish a structured representation that enables further compression. Building on this structure, we introduce an *entropy coding scheme* with a learnable context model guided by a binary hash grid, enabling effective probabilistic modeling of seed attributes. To mitigate the optimization overhead introduced by the large number of Gaussians and entropy estimation, we develop a *multi-scale fitting strategy* that progressively refines the representation from coarse to fine, accelerating optimization and enhancing its stability. Experiments on three megapixel-scale datasets show that SGI achieves competitive or superior fidelity with significantly reduced model size and optimization time, highlighting its potential as an effective solution for efficient and compact large-scale image representation.

Acknowledgment

This work was supported in part by the U.S. National Science Foundation under grants IIS-2101696, OAC-2104158, and IIS-2401144. J. Chen was supported by the Bundesministerium für Forschung, Technologie und Raumfahrt, BMFTR under the funding reference 161L0272 and the "Ministerium für Kultur und Wissenschaft des Landes Nordrhein-Westfalen" and "Der Regierende Bürgermeister von Berlin, Senatskanzlei Wissenschaft und Forschung."

References

[1] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41, 1984. 5

[2] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1): 90–93, 2006. 1

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of IEEE International Conference on Computer Vision*, pages 5855–5864, 2021. 2

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *Proceedings of IEEE International Conference on Computer Vision*, pages 19697–19705, 2023. 2

[5] Andrew Bond, Jui-Hsien Wang, Long Mai, Erkut Erdem, and Aykut Erdem. GaussianVideo: Efficient video representation via hierarchical Gaussian splatting. *arXiv preprint arXiv:2501.04782*, 2025. 2

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *Proceedings of European Conference on Computer Vision*, pages 333–350. Springer, 2022. 2

[7] Anthony Chen, Shiwen Mao, Zhu Li, Minrui Xu, Hongliang Zhang, Dusit Niyato, and Zhu Han. An introduction to point cloud compression standards. *GetMobile: Mobile Computing and Communications*, 27(1):11–17, 2023. 5

[8] Chuan Chen, Weiwei Wang, Xixi Jia, Xiangchu Feng, and Hanjia Wei. Local Gaussian ensemble for arbitrary-scale image super-resolution. *Computer Vision and Image Understanding*, 257:104372, 2025. 2

[9] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. NeRV: Neural representations for videos. In *Proceedings of Advances in Neural Information Processing Systems*, pages 21557–21568, 2021. 2

[10] Hao Chen, Matthew Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. HNeRV: A hybrid neural representation for videos. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 10270–10279, 2023. 2

[11] Jiawen Chen, Muqing Zhou, Wenrong Wu, Jinwei Zhang, Yun Li, and Didong Li. STimage-1K4M: A histopathology image-gene expression dataset for spatial transcriptomics. In *Proceedings of Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. 5

[12] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 8628–8638, 2021. 2

[13] Yihang Chen, Qianyi Wu, Mehrtash Harandi, and Jianfei Cai. How far can we compress Instant-NGP-based NeRF? In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 20321–20330, 2024. 3, 4

[14] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-grid assisted context for 3D Gaussian splatting compression. In *Proceedings of European Conference on Computer Vision*, pages 422–438, 2024. 3, 4, 5, 6

[15] Woong Oh Cho, In Cho, Seoha Kim, Jeongmin Bae, Youngjung Uh, and Seon Joo Kim. 4D scaffold Gaussian splatting for memory efficient dynamic scene reconstruction. *arXiv preprint arXiv:2411.17044*, 2024. 2, 3

[16] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. COIN: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021. 2

[17] Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Golinski, Yee Whye Teh, and Arnaud Doucet. COIN++: Neural compression across modalities. *Transactions on Machine Learning Research*, 2022, 2022. 2

[18] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian compression with 15× reduction and 200+ FPS. In *Proceedings of Advances in Neural Information Processing Systems*, 2024. 2

[19] Zongyu Guo, Gergely Flamich, Jiajun He, Zhibo Chen, and José Miguel Hernández-Lobato. Compression with Bayesian implicit neural representations. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1938–1956, 2023. 1

[20] Jun Han and Chaoli Wang. CoordNet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):4951–4963, 2023. 2

[21] Christopher E Heil and David F Walnut. Continuous and discrete wavelet transforms. *SIAM review*, 31(4):628–666, 1989. 1

[22] Jintong Hu, Bin Xia, Bin Chen, Wenming Yang, and Lei Zhang. GaussianSR: High fidelity 2D Gaussian splatting for arbitrary-scale image super-resolution. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 3554–3562, 2025. 2

[23] Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. Tri-MipRF: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *Proceedings of IEEE International Conference on Computer Vision*, pages 19774–19783, 2023. 2

[24] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *Proceedings of ACM SIGGRAPH Conference*, pages 32:1–32:11, 2024. 2

[25] Han Huang, Yulun Wu, Chao Deng, Ge Gao, Ming Gu, and Yu-Shen Liu. FatesGS: Fast and accurate sparse-view surface reconstruction using Gaussian splatting with depth-feature consistency. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 3644–3652, 2025. 2

[26] Andrey Ignatov, Radu Timofte, et al. Pirm challenge on perceptual image enhancement on smartphones: report. In *European Conference on Computer Vision (ECCV) Workshops*, 2019. 2

[27] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):139:1–139:14, 2023. 2, 4, 5, 6

[28] Ming Li, Teng Pan, H Guan, H Liu, and J Gao. Gaofen-2 mission introduction and characteristics. In *Proceedings of International Astronautical Congress*, pages 12–16, 2015. 1, 5

[29] Mufan Liu, Qi Yang, Miaoran Zhao, He Huang, Le Yang, Zhu Li, and Yiling Xu. D2GV: Deformable 2D Gaussian splatting for video representation in 400FPS. *arXiv preprint arXiv:2503.05600*, 2025. 2

[30] 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 IEEE Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 2, 3, 4, 5, 6, 7

[31] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. ACORN: Adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics*, 40(4):58:1–58:13, 2021. 2

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of European Conference on Computer Vision*, pages 405–421, 2020. 2

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41 (4):102:1–102:15, 2022. 2, 5, 6

[34] Long Peng, Anran Wu, Wenbo Li, PeizheXia, Xinjie Zhang, Xueyuan Dai, Xin Di, Haoze Sun, Renjing Pei, Yang Wang, Yang Cao, and Zheng-Jun Zha. Pixel to gaussian: Ultra-fast continuous super-resolution with 2d gaussian modeling. In *Proceedings of International Conference on Learning Representations*, 2026. 2

[35] Rawzor. Image Compression Benchmark. `https://imagecompression.info/test_images/`, 2015. Accessed: 2025-05-05. 1, 5

[36] Jorma Rissanen and Glen Langdon. Universal modeling and coding. *IEEE Transactions on Information Theory*, 27(1): 12–23, 1981. 4

[37] Vishwanath Saragadam, Jasper Tan, Guha Balakrishnan, Richard G Baraniuk, and Ashok Veeraraghavan. MINER: Multiscale implicit neural representation. In *Proceedings of European Conference on Computer Vision*, pages 318–333, 2022. 2

[38] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. WIRE: Wavelet implicit neural representations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 18507–18516, 2023. 1, 2

[39] Seungjoo Shin and Jaesik Park. Binary radiance fields. In *Proceedings of Advances in Neural Information Processing Systems*, pages 55919–55931, 2023. 4

[40] Vincent Sitzmann, Eric Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-learning signed distance functions. In *Proceedings of Advances in Neural Information Processing Systems*, pages 10136–10147, 2020. 2

[41] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proceedings of Advances in Neural Information Processing Systems*, pages 7462–7473, 2020. 1, 2, 5, 6

[42] Weronika Smolak-DyĹĹ'ewska, Dawid Malarz, Kornel Howil, Jan Kaczmarczyk, Marcin Mazur, PrzemysĹ Spurek, et al. VeGaS: Video Gaussian splatting. *arXiv preprint arXiv:2411.11024*, 2024. 2

[43] Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. In *Proceedings of International Conference on Learning Representations*, 2021. 4

[44] Kaiyuan Tang and Chaoli Wang. ECNR: Efficient compressive neural representation of time-varying volumetric datasets. In *Proceedings of IEEE Pacific Visualization Conference*, pages 72–81, 2024. 2

[45] Kaiyuan Tang, Siyuan Yao, and Chaoli Wang. iVR-GS: Inverse volume rendering for explorable visualization via editable 3D Gaussian splatting. *IEEE Transactions on Visualization and Computer Graphics*, 31(6):3783–3795, 2025. 2

[46] Luyang Tang, Jiayu Yang, Rui Peng, Yongqi Zhai, Shihe Shen, and Ronggang Wang. Compressing streamable free-viewpoint videos to 0.1 Mb per frame. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 7257–7265, 2025. 2

[47] Joanna Waczyńska, Tomasz Szczepanik, Piotr Borycki, Sławomir Tadeja, Thomas Bohné, and Przemysław Spurek. MiraGe: Editable 2D images using Gaussian splatting. *arXiv preprint arXiv:2410.01521*, 2024. 2

[48] Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex Kot, and Bihan Wen. ContextGS: Compact 3D Gaussian splatting with anchor level context model. In *Proceedings of Advances in Neural Information Processing Systems*, pages 51532–51551, 2024. 3, 4

[49] Jingyu Yang, Sheng Shen, Huanjing Yue, and Kun Li. Implicit transformer network for screen content image continuous super-resolution. In *Proceedings of Advances in Neural Information Processing Systems*, pages 13304–13315, 2021. 1

[50] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024. 2

[51] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025. 6

[52] Hao Zhang, Tianyuan Dai, Yanbo Xu, Yu-Wing Tai, and Chi-Keung Tang. FaceDNeRF: Semantics-driven face reconstruction, prompt editing and relighting with diffusion models. In *Proceedings of Advances in Neural Information Processing Systems*, pages 55647–55667, 2023. 1

[53] Jiahui Zhang, Fangneng Zhan, Ling Shao, and Shijian Lu. SOGS: Second-order anchor for advanced 3D Gaussian splatting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 11167–11176, 2025. 2, 3

[54] Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. GaussianImage: 1000FPS image representation and compression by 2D Gaussian splatting. In *Proceedings of European Conference on Computer Vision*, pages 327–345. Springer, 2024. 1, 2, 3, 4, 5, 6

[55] Yunxiang Zhang, Bingxuan Li, Alexandr Kuznetsov, Akshay Jindal, Stavros Diolatzis, Kenneth Chen, Anton Sochenov, Anton Kaplanyan, and Qi Sun. Image-GS: Content-adaptive image representation via 2D Gaussians. In *Proceedings of ACM SIGGRAPH Conference*, pages 1–11, 2025. 2

[56] Qi Zhao, M Salman Asif, and Zhan Ma. DNeRV: Modeling inherent dynamics via difference neural representation for videos. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2031–2040, 2023. 2

[57] 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 AAAI Conference on Artificial Intelligence*, pages 10977–10985, 2025. 1, 2, 3, 4, 5, 6

SGI: Structured 2D Gaussians for Efficient and Compact Large Image Representation

Supplementary Material

A. More Implementation Details

The learning rates used for different components are listed in Table A1. We apply learning rate decay to all components except for the seed features and scaling parameters. For optimization, we adopt the Adam optimizer with default momentum parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and set the numerical stability term to $\epsilon = 1 \times 10^{-15}$. Each of the three MLPs (i.e., $\mathrm{MLP}_c, \mathrm{MLP}_\Sigma, \mathrm{MLP}_p$) is composed of two fully connected (linear) layers with a ReLU activation function applied between them.

Table A1. Details of the learning rates in SGI.

Component	Learning rate
Seed position $\boldsymbol{x_a}$	0
Offsets $\boldsymbol{\delta}$	0.01
Seed feature $\boldsymbol{f_a}$	0.0075
Scaling $\{\boldsymbol{s_o}, \boldsymbol{s_a}\}$	0.007
MLP_c	0.008
MLP_Σ	0.004
MLP_p	0.005
Hash grid $\mathcal{H}$	0.005

B. Pseudocode

Algorithm 1 presents the full training pipeline of SGI. The input image is first downsampled to form a multi-level Gaussian pyramid, enabling coarse-to-fine optimization over M levels. At each level l, seed parameters and MLP weights are either initialized or adapted from the coarser level $l{+}1$. During training, we inject quantization-aware noise into seed attributes and decode their corresponding Gaussian primitives using shared MLPs. The image reconstruction loss L_{img} is computed with the SGI rendered images and ground truth images. In parallel, a context model guided by a binary hash grid estimates the probability distribution of seed attributes for entropy coding. After optimization, seed positions are compressed using GPCC, and seed attributes along with the hash grid are entropy-coded using arithmetic coding.

Algorithm 1 Structured Gaussian Image (SGI)

Input: Image I

Hyperparameters: Pyramid levels M, seed number N, Gaussians per seed K

```
1:  Initialize seeds $\mathbb{A} = \{\boldsymbol{x}_a^{(i)}, \mathcal{A}^{(i)}\}_{i=0}^{N-1}$
2:  Initialize learnable parameters $\theta = \{\theta_c, \theta_\Sigma, \theta_p, \theta_\mathcal{H}\}$ for MLPs and binary hash grid $\mathcal{H}$
3:  for $l = M-1, \ldots, 0$ do
4:      Downsample $I$ to $I_l$
5:      Initialize or adapt $\mathbb{A}^{(l)}, \theta^{(l)}$ from level $l{+}1$    ▷ Eq. (16)
6:      repeat
7:          Inject noise to $\mathbb{A}^{(l)}$ for quantization-aware training    ▷ Eq. (8)
8:          Decode Gaussians: $\{\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)}, \boldsymbol{c}^{(k)}\}_{k=0}^{K-1}$    ▷ Section 3.1
9:          Render image $\hat{I}_l$, compute $L_{\text{img}}(I_l, \hat{I}_l)$
10:         Estimate seed attributes probability via $\mathrm{MLP}_p$ and $\mathcal{H}$    ▷ Eqs. (10), (11)
11:         Compute $L_{\text{entropy}}$ and $L_{\text{hash}}$    ▷ Eqs. (12), (13)
12:         Update $\mathbb{A}^{(l)}, \theta^{(l)}$ using total loss    ▷ Eq. (14)
13:     until convergence
14: Compress seed positions via GPCC
15: Entropy-code seed attributes and hash grid via arithmetic coding
16: return seed positions, entropy-code attributes, and model parameters $\theta$
```

Table A2. **Unsupervised super-resolution performance on a DIV2K sample.** Quantitative results are measured in PSNR (dB).

Method	$\times 2$	$\times 4$	$\times 8$
Bilinear interpolation	28.40	27.24	26.86
SGI	**28.88**	**27.58**	**27.17**

Table A3. **Comparisons on DIV2K.** Optimization time is in minutes, and model size is in MB. *Numbers are averaged per image.*

Method	PSNR (dB)↑	SSIM↑	LPIPS↓	Opt. Time↓	Size↓
SIREN (NeurIPS'20)	24.71	0.6432	0.5019	28.37	0.45
GaussianImage (ECCV'24)	35.22	0.9327	0.1469	21.91	3.07
LIG (AAAI'25)	44.87	0.9904	0.0113	9.97	15.26
Our SGI (low-rate)	28.69	0.8206	0.3163	3.31	0.33
Our SGI (med-rate)	37.72	0.9643	0.0778	6.80	1.82
Our SGI (high-rate)	44.03	0.9872	0.0298	15.64	5.40

C. Additional Experimental Results

C.1. Continuous Attribute of SGI Image Representation

We investigate the continuous nature of the SGI-based image representation in Table A2. Specifically, we downsample a 2K image from the DIV2K dataset [26] to $2\times$, $4\times$, and $8\times$ resolutions. Our SGI model is trained using 50,000 Gaussians only on the $8\times$ downsampled image. The learned low-resolution representation is then directly used to generate higher-resolution outputs, leveraging the continuous property of the SGI representation. SGI achieves approximately a 0.5 dB gain in PSNR over bilinear interpolation on the low-resolution image. These results not only serve as the foundation for our multi-scale fitting strategy, but also demonstrate that the 2D Gaussian representation used in this work enables arbitrary-scale super-resolution, similar to [8, 22].

C.2. Evaluations on the DIV2K Dataset

While our main focus is on large, megapixel-scale images, where prior Gaussian- and INR-based representations face severe scalability and optimization challenges, the results on DIV2K [26] in Table A3 show that SGI also achieves strong RD performance on a standard image compression benchmark.

C.3. Analysis of Inference Time

Without optimization, arithmetic coding (AC) adds $\sim$80 ms latency for a 2K image with 50K Gaussians on an A10 GPU. The total latency ($\sim$88 ms) remains competitive with GS-based methods and much faster than INR-based methods (e.g., SIREN, $\sim$250 ms). Latency can be further reduced via parallel decoding across various attributes using tailored CUDA kernel designs.

D. Future Directions

While SGI establishes a robust baseline for seed-based Gaussian image representation, several promising avenues exist to further push the boundaries of compression efficiency and reconstruction fidelity.

Content-Adaptive Representation. Our current framework utilizes a fixed number of Gaussians K per seed to ensure optimization stability and implementation simplicity. However, the SGI architecture is inherently compatible with content-adaptive strategies. A natural extension would be to dynamically optimize K or the primitive allocation based on local texture complexity, allocating more expressive power to high-frequency details while maintaining sparsity in smooth regions. Such directions are orthogonal to our SGI and could be integrated with concurrent ideas, such as those in Image-GS [55], to further enhance fidelity at extremely low bitrates.

Advanced Optimization and Quantization. To maintain a lightweight and efficient training pipeline, SGI currently employs additive uniform noise to simulate quantization. Future research could explore more sophisticated quantization-aware training (QAT) techniques, such as stochastic Gumbel annealing (SGA) or learned rounding-to-nearest estimators.

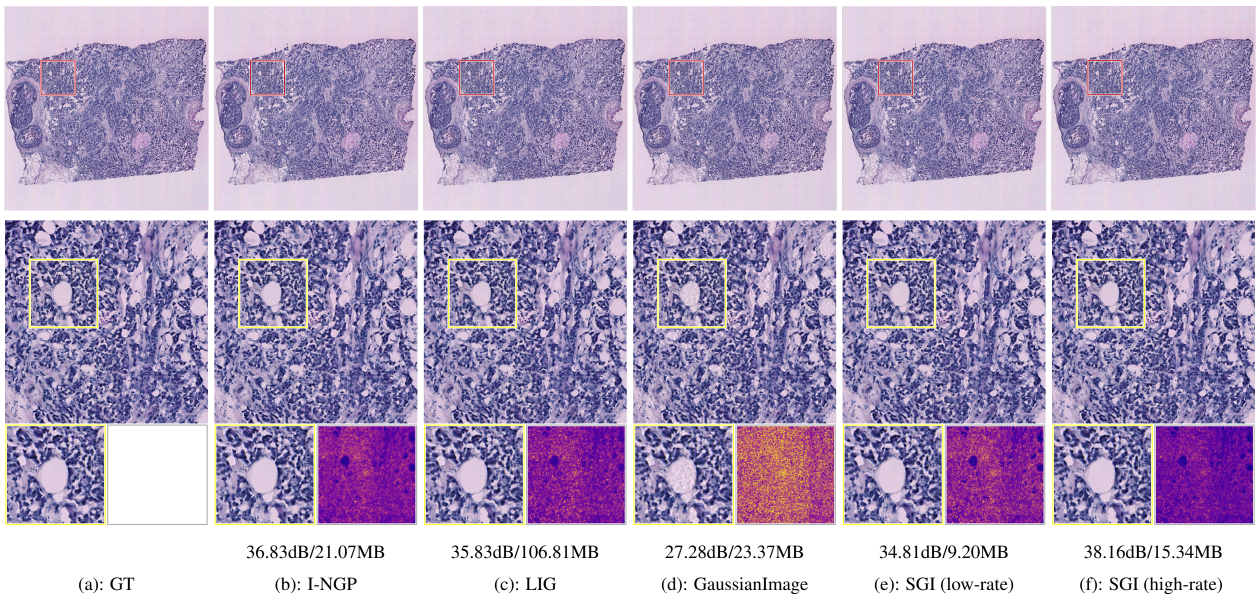


Figure A1. **Visual comparisons on STimage (w/ zoom-in cases and error maps).** Zoom-in regions highlight perceptual differences. The third row shows per-pixel reconstruction error heatmaps, where warmer colors (e.g., yellow) indicate larger deviations from the ground truth. PSNR (dB) and storage size (MB) for each method are shown below the visualizations.