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., $\text{MLP}_c, \text{MLP}_\Sigma, \text{MLP}_p$) 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

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

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

Method	×2	×4	×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

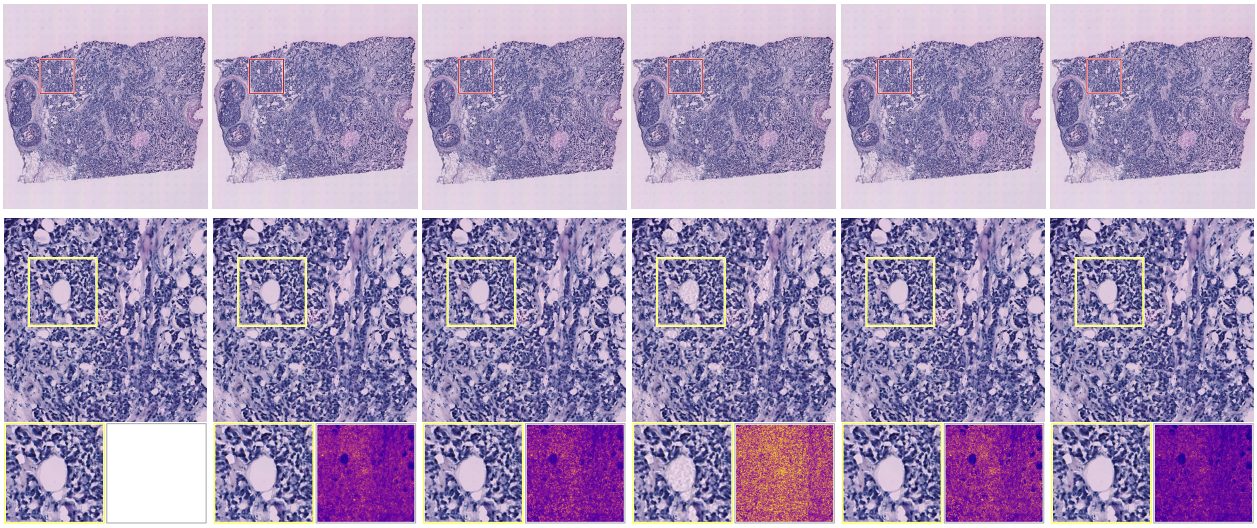
Without optimization, arithmetic coding (AC) adds $\sim$80 ms latency for a 2K image with 50K Gaussians on an A10 GPU. The total latency ($\sim$88 ms) remains competitive with GS-based methods and much faster than INR-based methods (e.g., SIREN, $\sim$250 ms). Latency can be further reduced via parallel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



	36.83dB/21.07MB	35.83dB/106.81MB	27.28dB/23.37MB	34.81dB/9.20MB	38.16dB/15.34MB
(a): GT	(b): I-NGP	(c): LIG	(d): GaussianImage	(e): SGI (low-rate)	(f): SGI (high-rate)

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.