FROM PIXELS TO COMPACT GAUSSIANS: GENERALIZABLE 2D GAUSSIAN IMAGE REPRESENTATION VIA ERROR-AWARE GRAPH COARSENING

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing 2D Gaussian image representations often learn to place primitives from scratch or optimize them separately for each image. We observe that an image itself already provides an analytic starting point: each pixel supplies a Gaussian center and color, while its local structure tensor determines an anisotropic covariance. Rasterizing this dense field achieves approximately 25–30 dB PSNR without training or per-image optimization. This high-fidelity but overcomplete initialization reframes compact Gaussian representation as a budget-constrained simplification problem. This overcomplete foundation shifts the core objective from synthesizing primitives to strategically pruning them under reconstruction feedback. To this end, we introduce **Error-aware Pixel-Inductive Gaussians** (EPIG), a bottom-up framework that alternates error-aware graph coarsening with reconstruction-driven refinement. A merging network selects neighboring primitives for moment-matched contraction, and a refinement network updates the survivors using the rendering residual. Repeating these steps yields a compact representation in a few feed-forward stages without test-time optimization. Across three budgets on DIV2K and zero-shot Kodak, EPIG improves PSNR over the evaluated feed-forward baselines.

1 INTRODUCTION

Representing an image with a compact set of 2D Gaussian primitives has recently emerged as an attractive alternative to implicit neural image representations (Sitzmann et al., 2020; Saragadam et al., 2023; Müller et al., 2022; Dupont et al., 2021; 2022). Inherited from the success of 3D Gaussian Splatting (Kerbl et al., 2023), GaussianImage (Zhang et al., 2024) first demonstrated that images can be represented by a sparse collection of anisotropic 2D Gaussians while retaining high visual fidelity and enabling extremely fast rasterization. Since then, Gaussian-based image representations have rapidly expanded to image compression (Li et al., 2026b; Zhu et al., 2025), video coding (Wang et al., 2025b; Lee et al., 2025), super-resolution (Hu et al., 2024; Chen et al., 2025), and downstream low-level vision tasks (Chen et al., 2026c;d).

Despite this progress, constructing the Gaussian primitives themselves remains a major bottleneck. Most existing methods optimize Gaussian positions, covariances, colors, and opacities independently for each image, often requiring thousands of gradient-descent iterations (Zhang et al., 2024; Li et al., 2026b; Zhu et al., 2025). Such per-image fitting is computationally expensive and, more importantly, the representation learned does not naturally generalize to unseen images. Recent feed-forward approaches (Wang et al., 2025a; Zeng et al., 2026) alleviate this problem by directly predicting Gaussian primitives or progressively increasing their capacity. Nevertheless, they largely follow a *top-down* paradigm: starting from a small or weakly grounded set of primitives and learning where Gaussians should be placed, split, or densified. Consequently, the network must learn not only how to produce a compact representation, but also the more fundamental question of where image primitives should exist in the first place.

We revisit this assumption and ask a different question: *Do Gaussian primitives really need to be synthesized from scratch?*

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.

Our key observation is that **every pixel induces a 2D Gaussian in an analytic initialization**. A pixel provides its position and color, and the local structure tensor's eigensystem supplies an analytic orientation and anisotropic covariance. Direct rasterization of the resulting HW-primitive field achieves approximately 25–30 dB PSNR, where W and H are the image width and height. Starting from this, we aim to use much fewer Gaussians at a prescribed budget N. The central problem is thus to remove redundant primitives and correct the reconstruction errors introduced by their removal.

Based on this insight, we propose Error-aware Pixel-Inductive Gaussians (EPIG), a generalizable bottom-up framework that converts pixel-induced Gaussians into a compact representation through error-aware graph coarsening. A compatibility-driven Merging Prediction Network selects neighboring Gaussians for contraction under the current primitive budget. A reconstruction-driven Refinement Network then adjusts the surviving attributes according to the rendering residual. The merging network decides *where capacity can be removed*, while the refinement network determines *how the remaining capacity should be used*. Passing the refined state to the next stage lets subsequent contractions account for earlier corrections. After a few feed-forward stages, the output contains the requested number of Gaussians rather than one Gaussian per pixel, with no test-time optimization.

Our contributions are summarized as follows:

- **Analytic pixel-induced initialization.** We use pixel attributes and local structure tensors to construct an initial dense anisotropic Gaussian field that achieves approximately 25–30 dB PSNR without learning or per-image optimization. The field provides a grounded starting point for subsequent compression.
- **Error-aware graph coarsening.** We introduce a closed-loop coarsening mechanism that couples graph edge contraction with rendering feedback. Reconstruction residuals guide attribute refinement, and the refined Gaussian state informs subsequent merge decisions.
- **Generalizable compact representation.** With fixed network weights, EPIG converts unseen images into a user-specified number of Gaussians using a feed-forward network without test-time optimization. It outperforms the evaluated feed-forward baselines in PSNR at all tested budgets on DIV2K and zero-shot Kodak.

2 Related Work

Optimization-based 2D Gaussian image representation. GaussianImage (Zhang et al., 2024) introduced Gaussian splatting for single-image representation, where Gaussian parameters are optimized separately for each image. Early methods typically initialize primitives randomly or on regular grids and rely on gradient descent to recover the image structure (Zhang et al., 2024; Zhu et al., 2025). Subsequent work has explored more informed initialization strategies. For example, structure-guided methods place Gaussians according to image gradients, superpixels, or local color variations (Liang et al., 2025; Li et al., 2026a), while other approaches exploit segmentation boundaries (Takabe et al., 2026), frequency information (Wan et al., 2025), or local entropy (Chen et al., 2026a). Beyond initialization, several methods adapt the Gaussian set during fitting through distortion-driven densification or merging (Li et al., 2026b; Chen et al., 2026b). Although these designs improve fitting quality or efficiency, they still require image-specific optimization of Gaussian parameters, often for thousands of iterations. As a result, the representation learned for one image cannot be directly reused for another, and encoding remains relatively expensive.

Feed-forward 2D Gaussian image representation. Recent work has explored feed-forward alternatives to reduce the cost of per-image fitting. Some methods directly predict Gaussian parameters from image features (Zeng et al., 2025; Kim et al., 2026), while others introduce learned priors (Wang et al., 2025a), cross-image eigenspace initialization (Tai et al., 2025), or hybrid schemes that alternate prediction and optimization (Zeng et al., 2026). Despite amortizing compute time, these methods still adhere to a top-down paradigm: they rely on networks to directly synthesize or parameterize primitives, inherently coupling primitive placement and geometry with predictor capacity, thereby introducing significant learning complexity. We depart from this top-down formulation by recognizing that an image naturally induces a dense, analytic Gaussian field. This shifts the core challenge from primitive synthesis to *bottom-up* representation simplification, where learning focuses on error-aware coarsening and reconstruction-driven refinement rather than blind generation from scratch.

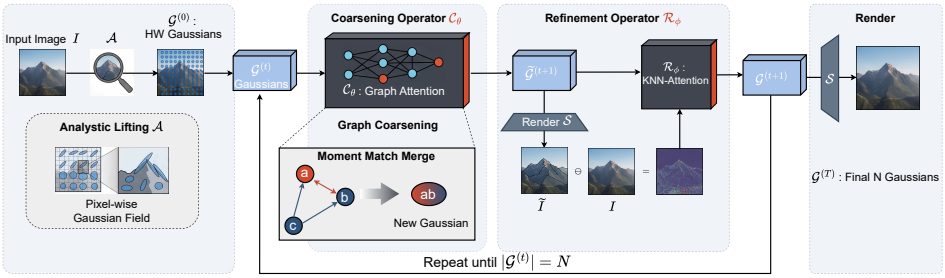

Figure 1: The EPIG pipeline. An analytic structure-tensor lifting initializes a dense Gaussian field. Successive stages alternate between graph edge contraction and rendering-guided refinement to reach target budget N, enabling feed-forward inference without per-image adaptation.

3 METHOD

3.1 OVERVIEW AND PROBLEM FORMULATION

Given an image $I \in \mathbb{R}^{H\times W\times 3}$ and a primitive budget N, we seek a compact set of N 2D Gaussians whose rendering reconstructs I. Each Gaussian has a center μ_i, covariance Σ_i, color c_i, and opacity α_i. We denote the differentiable renderer adopted from GaussianImage (Zhang et al., 2024) by $\mathcal{S}$. EPIG begins with an analytically constructed, pixel-wise Gaussian field and progressively reduces it to the requested budget. Two networks predict which primitives to merge and how to update the surviving attributes for accurate reconstruction.

As illustrated in Figure 1, the pipeline comprises three steps. First, an analytic lifting operator $\mathcal{A}$ converts every pixel into an anisotropic Gaussian, yielding an overcomplete initial state $\mathcal{G}^{(0)} = \mathcal{A}(I)$ with $|\mathcal{G}^{(0)}| = HW$. Centers and colors are inherited from pixels, while covariances are derived from local structure tensors. This training-free construction already provides a faithful image representation (Section 3.2). Second, a learned coarsening operator $\mathcal{C}_\theta$ predicts compatible edges on the current region-adjacency graph. Selected pairs are contracted through moment matching, reducing the primitive count toward N. Prediction thus focuses on which existing primitives can be combined while preserving image structure (Section 3.3). Third, a reconstruction-driven operator $\mathcal{R}_\phi$ renders the contracted set and uses its discrepancy from the input image to predict attribute updates for the surviving Gaussians. The refined state supplies the input to the next coarsening stage. Coarsening and refinement are executed in this order at every stage and are trained jointly through reconstruction supervision. The complete reduction process is

$$\mathcal{G}^{(0)} = \mathcal{A}(I), \qquad \underbrace{\boxed{\mathcal{G}^{(t)} \xrightarrow{\mathcal{C}_\theta} \widetilde{\mathcal{G}}^{(t+1)} \xrightarrow{\mathcal{R}_\phi(\cdot\,;I)} \mathcal{G}^{(t+1)}}}_{\text{repeat until } |\mathcal{G}^{(T)}|=N}. \tag{1}$$

Coarsening thus decides where representation capacity can be removed, whereas refinement decides how the remaining capacity should be redistributed. The two components are coupled by joint training through the sequential pipeline, and at inference the same finite-stage pipeline converts an unseen image to a requested budget using only feed-forward passes, without per-image optimization (Section 3.4).

3.2 ANALYTIC PIXEL-TO-GAUSSIAN INITIALIZATION

Our starting point is a deterministic lifting from a discrete image to a dense, continuous Gaussian field. At initialization, every pixel p supplies a primitive center and color, $\mu_p = p$ and $c_p = I(p)$, with initial opacity $\alpha_p = 1$. Its covariance is derived from local image structure so that the primitive follows nearby edges instead of spreading uniformly across them.

Let g_x and g_y denote the horizontal and vertical gradients of image luminance. Their local directional variation is summarized by the structure tensor (Harris & Stephens, 1988),

$$J(p) = \begin{pmatrix} (k * g_x^2)(p) & (k * g_x g_y)(p) \\ (k * g_x g_y)(p) & (k * g_y^2)(p) \end{pmatrix}, \tag{2}$$

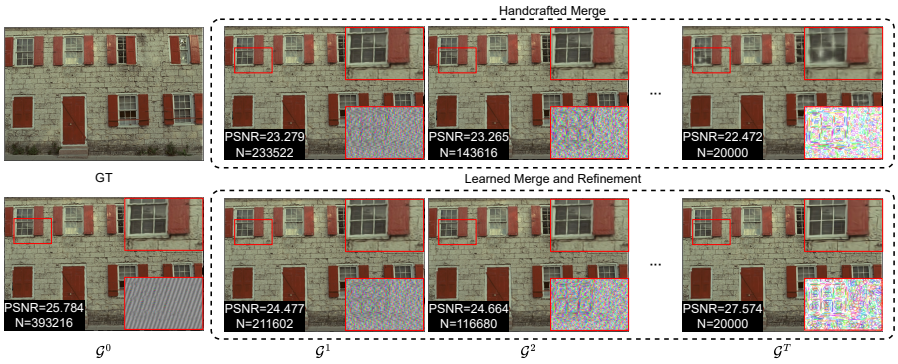

Figure 2: Qualitative progression from $\mathcal{G}^{(0)}$ to $\mathcal{G}^{(T)}$ on a representative image. Top: handcrafted merging; bottom: our learned pipeline (refinement is bypassed when the primitive count exceeds 150k during training for efficiency). Even without refinement, the learned merge policy achieves higher fidelity than handcrafted rules (e.g., col. 2). With reconstruction-aware refinement, the final representation (last col.) achieves higher PSNR than the dense initialization (first col.) despite using over $10\times$ fewer Gaussians, while exhibiting better structural alignment.

where k is a fixed 9×9 Gaussian kernel and $*$ denotes convolution. Let $\nu_{1,p} \geq \nu_{2,p}$ be its eigenvalues and $R(\theta_p) = [v_{1,p}, v_{2,p}]$ the corresponding orthonormal eigenvectors. The dominant direction $v_{1,p}$ follows maximum intensity change, or the local edge normal, while $v_{2,p}$ follows the edge tangent. The ratio $\rho_p = \sqrt{\nu_{1,p}/\nu_{2,p}}$ measures local anisotropy; degenerate cases use a bounded ratio, with isotropic support in flat regions. We construct the covariance as

$$\Sigma_p = R(\theta_p)\ \mathrm{diag}(\rho_p^{-1}, \rho_p)\, R(\theta_p)^\top, \tag{3}$$

Assigning the smaller variance to the edge normal compresses the primitive across an edge and elongates it along the tangent. Regions with similar directional variation in both axes induce approximately isotropic primitives. Each primitive also carries an additive mass $w_p = \sqrt{\det \Sigma_p}$, which initially serves as an area proxy for its spatial support. This mass is accumulated during contraction and retained during refinement, recording the amount of original support represented by a surviving primitive.

The resulting HW-primitive field is fully determined by the image and fixed operators. Direct rasterization already achieves approximately 25–30 dB PSNR without training or per-image optimization. This fidelity makes the field a useful starting point for reduction; subsequent coarsening and refinement replace it with a compact N-primitive representation.

3.3 Error-Aware Graph Coarsening and Refinement

Reducing the overcomplete field to a prescribed budget N requires identifying redundant local primitives while preserving reconstruction quality. Even a locally plausible contraction can introduce rendering error, so we alternate graph coarsening with reconstruction-guided refinement; the updated survivors then inform subsequent merging decisions. To define candidate contractions, we organize the field as a region-adjacency graph. At stage t, each node stores a Gaussian and a region Ω_i inherited from its source pixels; an edge connects two nodes whose regions share a boundary. A merging operation unions the associated regions, while refinement updates Gaussian attributes without changing pixel ownership. Below, we omit stage indices when describing a single iteration.

Handcrafted merging prior. To preserve image structure during reduction, we define a handcrafted merging cost that penalizes color mismatch, high-gradient content and geometric separation:

$$C(a,b) = \underbrace{1 - e^{-\|c_a - c_b\|_2}}_{\text{color}} + \underbrace{\kappa(g_a + g_b; \tau_g)}_{\text{frequency}} + \underbrace{\kappa(d_{ab}; \tau_m)}_{\text{geometry}}, \tag{4}$$

where g_i is the input-image gradient energy accumulated over Ω_i (with i denoting adjacent Gaussians a and b), $d_{ab} = (\mu_a - \mu_b)^\top \Sigma_{ab}^{-1} (\mu_a - \mu_b)$ measures separation under the candidate's moment-matched covariance Σ_{ab} in Eq. 6, and $\kappa(x;\tau) = \mathrm{sigmoid}(\log(x+1) - \log\tau)$. Greedy contraction

under this local rule yields reasonable reconstruction fidelity (Figure 2, top row) and region-based pseudo-labels for pretraining the merging predictor. However, being a purely local heuristic, it lacks data-driven adaptability and cannot optimize for end-to-end rendering fidelity; as demonstrated in our ablation study (Section 4), incorporating learning and joint optimization substantially improves reconstruction quality. We therefore use the handcrafted rule to initialize the merging predictor and retain $C(a, b)$ as a structural regularizer during joint training (Section 3.4).

Learned merging prediction and contraction. To enable feed-forward merging prediction beyond the static handcrafted rule, a graph network is proposed to score candidate edges from the current Gaussian state and target budget. Node descriptors encode Gaussian attributes, additive mass, and regional statistics, whereas edge descriptors e_{ab} incorporate the individual cost components of $C(a, b)$ (color, frequency, and geometry). The merging predictor $\mathcal{C}_\theta$ comprises graph-attention Transformer layers that aggregate neighboring information into node embeddings h_i and a shared head that predicts edge scores:

$$\pi_{ab} = \mathrm{sigmoid}(\mathcal{C}_\theta([h_a,\, h_b,\, h_a \odot h_b,\, e_{ab},\, \log(n_t/N)])) \,. \tag{5}$$

Here $n_t = |\mathcal{G}^{(t)}|$, and $\log(n_t/N)$ conditions the same predictor on the remaining reduction across stages and budgets. Edge endpoints follow a canonical order.

We select mutual-best edges, each attaining the highest score at both endpoints, with deterministic tie-breaking. This yields node-disjoint pairs that can be contracted in parallel. We retain at most $n_t - N$ pairs to avoid overshooting the target budget. Each selected pair is replaced by its moment-matched Gaussian:

$$\begin{aligned} x_{ab} &= \frac{w_a x_a + w_b x_b}{w_a + w_b}, && x \in \{\mu, c, \alpha\}, \\ \Sigma_{ab} &= \frac{\sum_{i\in\{a,b\}} w_i[\Sigma_i + (\mu_i - \mu_{ab})(\mu_i - \mu_{ab})^\top]}{w_a + w_b}. \end{aligned} \tag{6}$$

The merged node inherits mass $w_a + w_b$ and region $\Omega_a \cup \Omega_b$. The learned policy thus amortizes merging selection across images, while analytic contraction preserves mass-weighted first and second spatial moments. The resulting contracted set is $\widetilde{\mathcal{G}}^{(t+1)}$.

Reconstruction-aware refinement. Although moment matching preserves the spatial moments of a merged pair, it does not guarantee optimal rendering fidelity. We render the contracted set as $\widetilde{I} = \mathcal{S}(\widetilde{\mathcal{G}}^{(t+1)})$ and use its discrepancy from I to predict corrections for the surviving primitives. A shared frozen encoder $\mathcal{B}$ extracts multi-layer features of the input and of the rendered image. Their feature-space difference and the pixel-space rendering residual are projected and pooled over each Gaussian's spatial footprint:

$$r_i = \mathcal{P}_i\Big[\phi_f\Big(\mathcal{B}(\widetilde{I}) - \mathcal{B}(I)\Big) + \phi_p\Big(\widetilde{I} - I\Big)\Big], \tag{7}$$

where ϕ_f and ϕ_p are learned projections and $\mathcal{P}_i$ denotes normalized Gaussian-weighted pooling. This descriptor supplies the feature-space and pixel-space reconstruction errors localized to each primitive.

A learned input projection combines r_i with the contracted Gaussian attributes, a recurrent state, and $\log(n_{t+1}/N)$ into a unified primitive token. A graph transformer then exchanges information among spatial k-nearest neighbors, with attention biased by appearance and orientation similarity. Let z_i denote the resulting token of Gaussian i. A shared update head predicts its attribute increments:

$$(\Delta\mu_i, \Delta\boldsymbol{\ell}_i, \Delta\vartheta_i, \Delta c_i, \Delta\alpha_i) = \mathcal{R}_\phi^{\mathrm{update}}(z_i), \qquad \Delta\boldsymbol{\ell}_i = (\Delta\ell_{1,i}, \Delta\ell_{2,i}). \tag{8}$$

The recurrent state is initialized at each stage from pooled encoder features of the input image. When a stage uses multiple refinement steps, the transformer updates this state residually to accumulate corrections. Let $\nu_{1,i}, \nu_{2,i}$ and ϑ_i denote the current covariance's eigenvalues and orientation, initialized by the moment-matched contraction. The attribute updates are formulated as

$$\begin{aligned} \mu_i' &= \mu_i + \Delta\mu_i, \quad \nu_{k,i}' = \nu_{k,i} e^{\Delta\ell_{k,i}}, \quad k \in \{1, 2\}, \\ \vartheta_i' &= \vartheta_i + \Delta\vartheta_i, \quad \Sigma_i' = R(\vartheta_i')\,\mathrm{diag}(\nu_{1,i}', \nu_{2,i}')R(\vartheta_i')^\top. \end{aligned} \tag{9}$$

Color and opacity receive additive residuals and are clipped to $[0, 1]$. The exponential eigenvalue update strictly preserves positive definiteness, while region assignments and additive mass remain fixed. The resulting $\mathcal{G}^{(t+1)}$ supplies the state for the next merging prediction. Thus, although the recurrent state is reinitialized at each stage, the refined Gaussian attributes successfully carry rendering feedback into subsequent contractions.

3.4 Joint Training and Feed-Forward Inference

From local supervision to reconstruction feedback. Training proceeds in three phases. We first pretrain the merging predictor on candidate edges $\mathcal{E}_t$ using binary targets $y_{ab} \in \{0, 1\}$ that indicate whether a pair belongs to the same region after the handcrafted merging rollout:

$$\mathcal{L}_{\text{pre}} = -\frac{1}{|\mathcal{E}_t|} \sum_{(a,b)\in\mathcal{E}_t} [y_{ab} \log \pi_{ab} + (1 - y_{ab}) \log(1 - \pi_{ab})]. \tag{10}$$

The refinement network is then pretrained on states produced by the merging predictor. Finally, both networks are trained jointly on sampled images and budgets. At a sampled stage, a rollout supplies the current Gaussian state; contraction, refinement, and rendering then determines the reconstruction loss. Merging scores are thereby trained against the fidelity achievable after refinement, beyond their initial structural supervision.

Gradients through contraction. Hard matching makes merging selection nondifferentiable. Let $b_{ab} \in \{0, 1\}$ indicate the edges it selects. To propagate rendering gradients to the edge scores, we use a straight-through surrogate

$$m_{ab} = \text{sg}(b_{ab} - \widetilde{m}_{ab}) + \widetilde{m}_{ab}, \qquad \widetilde{m}_{ab} = \text{sigmoid}((s_{ab} + \varepsilon_{ab})/\tau), \tag{11}$$

where $s_{ab} = \text{logit}(\pi_{ab})$, ε_{ab} is a training-time Gumbel perturbation, τ is a temperature, and sg denotes stop-gradient. In the forward pass $m_{ab} = b_{ab}$, so selected pairs are contracted exactly. In the backward pass, gradients follow $\widetilde{m}_{ab}$; for a selected pair, the surrogate interpolates its two slots toward the analytic merged attributes. This biased approximation lets rendering supervision update the merging scores.

Training objective. Given the rendering after refinement $\widehat{I} = \mathcal{S}(\mathcal{G}^{(t+1)})$, image fidelity is measured by

$$\mathcal{L}_{\text{rec}} = \frac{\|I - \widehat{I}\|_2^2}{3HW}, \qquad \mathcal{L}_{\text{grad}} = \frac{\sum_{d\in\{h,v\}} \|\nabla_d I - \nabla_d \widehat{I}\|_1}{3HW}. \tag{12}$$

where ∇_h and ∇_v are the horizontal and vertical Sobel operators. During joint training, a regression term anchors predicted scores to the handcrafted cost:

$$\mathcal{L}_{\text{prior}} = \frac{1}{|\mathcal{E}_t|} \sum_{(a,b)\in\mathcal{E}_t} \left(s_{ab} - [\eta_1 - \eta_2 \ \text{sg}(C(a, b))]\right)^2. \tag{13}$$

The cost is evaluated before contraction and detached from the gradient, so this term favors the handcrafted ranking without allowing attribute updates to reduce the target cost. Both this loss and pretraining cover candidate edges $\mathcal{E}_t$; pretraining uses binary region labels, whereas the joint-stage regularizer uses a continuous cost-derived target.

To penalize vanishing covariance support, we additionally use

$$\mathcal{L}_{\text{shape}} = \frac{1}{n_{t+1}} \sum_i \left(e^{-\nu_{\min,i}/2} + e^{-\nu_{\max,i}/2}\right), \tag{14}$$

where $\nu_{\min,i}$ and $\nu_{\max,i}$ are the extreme eigenvalues of Σ'_i. The complete objective is

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_g \mathcal{L}_{\text{grad}} + \lambda_p \mathcal{L}_{\text{prior}} + \lambda_s \mathcal{L}_{\text{shape}}. \tag{15}$$

The image-space losses train merging prediction and refinement together through the surrogate, while the prior and shape terms regularize the learned policy and resulting Gaussians. Error awareness comes from supervising the refined rendering and passing the refined state to subsequent merging stages. More implementation details are shown in appendix.

Feed-forward inference. For an unseen image, we repeatedly apply merging prediction, moment-matched contraction, and reconstruction-aware refinement to its analytic Gaussian field. Each matching is capped at $n_t - N$ pairs, and stages continue until $n_t = N$, yielding exactly the requested budget without overshooting it. Network weights remain fixed; Gaussian attributes change only through analytic contraction and predicted residual updates. The target budget conditions the same trained model across stages, without test-time gradient descent.

4 EXPERIMENTS

We train on DIV2K's 800 training images and evaluate on its 100-image validation set (Agustsson & Timofte, 2017). After bicubic $2\times$ downsampling, image resolutions range from 408×1020 to 1020×1020. We also test transfer without adaptation on Kodak's 24 images of size 768×512 (Kodak) using the same trained model.

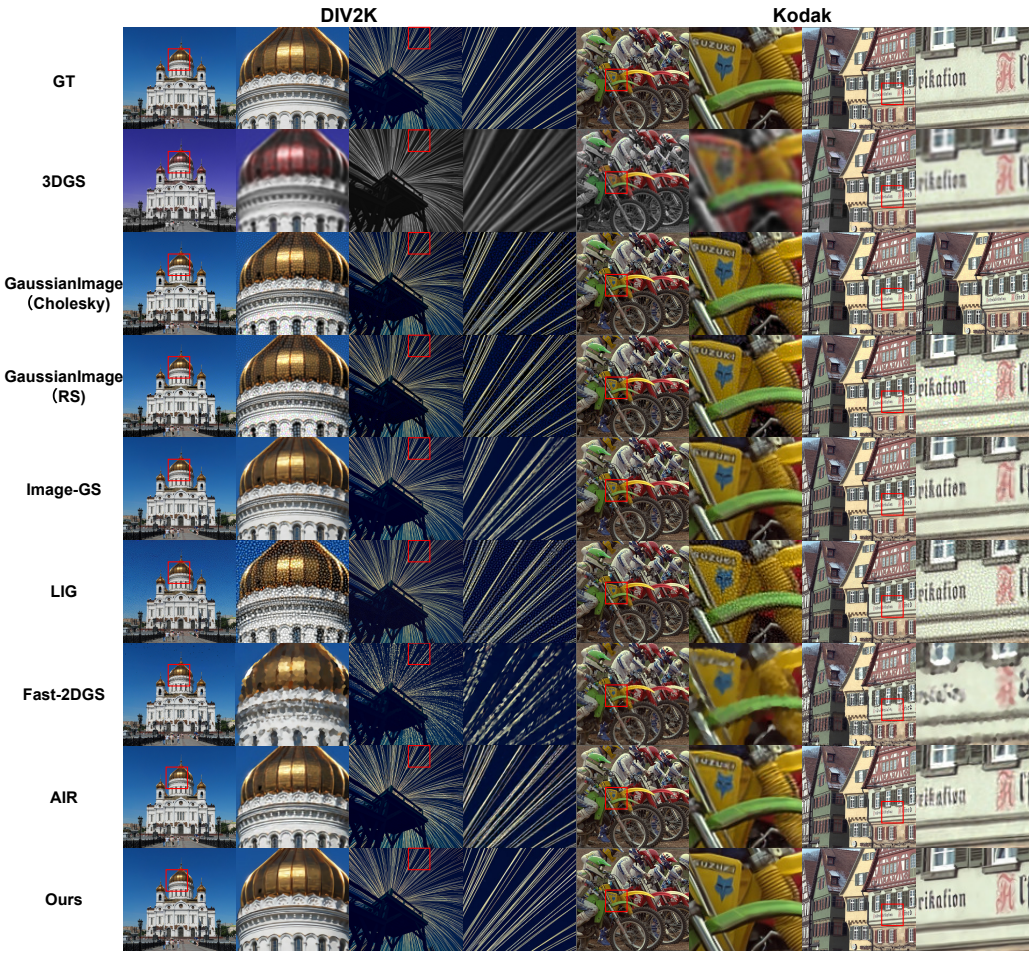

Figure 3: Representative DIV2K and Kodak reconstructions with enlarged regions.

We group baselines by whether they require image-specific optimization at inference. The *optimization-based* group includes 3D Gaussian Splatting adapted to single-image fitting (Kerbl et al., 2023), GaussianImage with Cholesky and rotation–scale covariance parameterizations (Zhang et al., 2024), Image-GS (Zhang et al., 2025), and LIG (Zhu et al., 2025). The *feed-forward* group includes AIR (Zeng et al., 2026) and Fast-2DGS (Wang et al., 2025a); for the latter, we report its output before optional image-specific fine-tuning. All baselines use their official implementations. Since AIR cannot guarantee an exact final primitive count, we select its output closest to each target budget. We report PSNR and MS-SSIM (Wang et al., 2003), LPIPS (Zhang et al., 2018), and per-image runtime on one NVIDIA GeForce RTX 3090. For time-constrained evalu-

ation, following AIR (Zeng et al., 2026), we run GaussianImage for 1,000 iterations on DIV2K and 800 on Kodak, and Image-GS for 600 iterations on both datasets, with densification after iteration 300. We also report 50,000-iteration DIV2K results for optimization-based methods as a higher-compute reference. All experiments use fixed hyperparameters: $(\tau_g, \tau_m, \tau) = (20, 2, 1)$, $(\lambda_{\text{rec}}, \lambda_g, \lambda_p, \lambda_s) = (15, 1, 0.5, 0.1)$, and $(\eta_1, \eta_2) = (2.5, 1.5)$. The merging predictor is pretrained for 12 epochs using pseudo-labels generated by the handcrafted merge rule.

Table 1: Quantitative results on DIV2K and zero-shot Kodak dataset at three Gaussian budgets. Time is in seconds.

		DIV2K				Kodak				50k iterations (DIV2K)	
Method	N	PSNR ↑	MS-SSIM ↑	LPIPS ↓	time	PSNR ↑	MS-SSIM ↑	LPIPS ↓	time	PSNR ↑	time
Per-image optimization											
3DGS	10k	20.62	0.7930	0.7068	3.8	21.06	0.8327	0.7189	2.4	26.65	187.1
	20k	22.17	0.8566	0.6038	4.3	22.72	0.8869	0.5903	2.6	29.33	192.1
	50k	24.81	0.9230	0.4451	5.7	24.94	0.9383	0.4363	3.6	33.72	279.6
Gaussian-Image (Cholesky)	10k	13.05	0.5690	0.7584	2.4	15.95	0.5741	0.8143	1.6	29.61	150.3
	20k	19.67	0.7439	0.6616	2.4	23.68	0.8418	0.5238	1.7	32.50	136.7
	50k	27.72	0.9348	0.3106	2.8	30.57	0.9654	0.1922	2.0	37.42	130.0
Gaussian-Image (RS)	10k	13.10	0.5706	0.7595	2.5	16.04	0.5778	0.8172	1.8	29.67	133.6
	20k	19.76	0.7466	0.6606	2.6	23.85	0.8468	0.5220	1.8	32.59	200.4
	50k	27.95	0.9367	0.3128	2.9	30.93	0.9667	0.2046	2.0	**37.65**	188.1
Image-GS	10k	**27.62**	**0.9528**	**0.2526**	2.3	**29.00**	**0.9552**	**0.2328**	2.0	**31.02**	55.9
	20k	28.86	0.9687	0.1966	2.5	30.17	0.9685	**0.1742**	2.2	**33.62**	62.8
	50k	30.44	0.9810	0.1263	3.2	31.83	0.9807	0.1030	2.9	37.21	79.7
LIG	10k	8.87	0.1770	0.7765	2.9	11.04	0.2416	0.7804	1.8	28.04	161.6
	20k	14.03	0.4589	0.7230	2.9	17.64	0.5857	0.6642	1.9	31.00	161.8
	50k	22.38	0.8191	0.5362	3.2	26.10	0.9086	0.4825	2.2	36.23	140.6
Feed-forward methods											
Fast-2DGS	10k	15.17	0.5314	0.5357	0.04	20.52	0.7278	0.4958	0.02	—	—
	20k	20.07	0.7608	0.4438	0.04	23.42	0.8427	0.4070	0.02	—	—
	50k	23.43	0.8801	0.3158	0.04	25.38	0.9026	0.3524	0.02	—	—
AIR	10k	25.38	0.9382	0.3617	0.43	25.59	0.9239	0.4172	0.43	—	—
	20k	26.28	0.9446	0.3176	0.43	29.27	0.9628	0.1857	0.43	—	—
	50k	29.45	0.9772	0.1480	0.44	31.67	0.9827	**0.0904**	0.43	—	—
EPIG	10k	26.73	0.9408	0.2609	2.70	28.79	0.9535	0.2561	2.04	—	—
	20k	**29.02**	**0.9690**	**0.1908**	2.40	**30.75**	**0.9735**	0.1950	1.80	—	—
	50k	**31.73**	**0.9853**	**0.1113**	1.92	**33.51**	**0.9864**	0.1172	1.52	—	—

Quantitative results. Table 1 demonstrates the strong performance of EPIG across datasets and representation budgets. Among feed-forward methods, EPIG consistently achieves the highest PSNR and MS-SSIM at all evaluated budgets on both DIV2K and Kodak. Notably, the improvements transfer directly to Kodak using the model trained solely on DIV2K, demonstrating strong zero-shot generalization without any per-image adaptation. Compared with optimization-based methods, EPIG exhibits a favorable capacity-scaling behavior. While Image-GS performs better under the most constrained budget (10k), EPIG surpasses it at 20k and further widens the margin at 50k in both PSNR and MS-SSIM, while also achieving the lowest LPIPS on DIV2K. This crossover validates our bottom-up coarsening-and-refinement pipeline across budgets.

Per-image optimization with 50,000 iterations still establishes a higher numerical PSNR ceiling, but requires substantial optimization time for every input image. In contrast, EPIG produces high-fidelity reconstructions through a fixed sequence of lightweight forward passes, entirely eliminating test-time optimization. Overall, these results show that EPIG achieves a strong trade-off among reconstruction quality, representation capacity, inference efficiency, and cross-dataset generalization.

Qualitative comparison. Figure 3 compares reconstruction quality on DIV2K and illustrates zero-shot generalization to Kodak. In the DIV2K image (cols. 3–4), the radiating lines are thin, continuous, and well separated in the reference. Both GaussianImage variants recover their directions

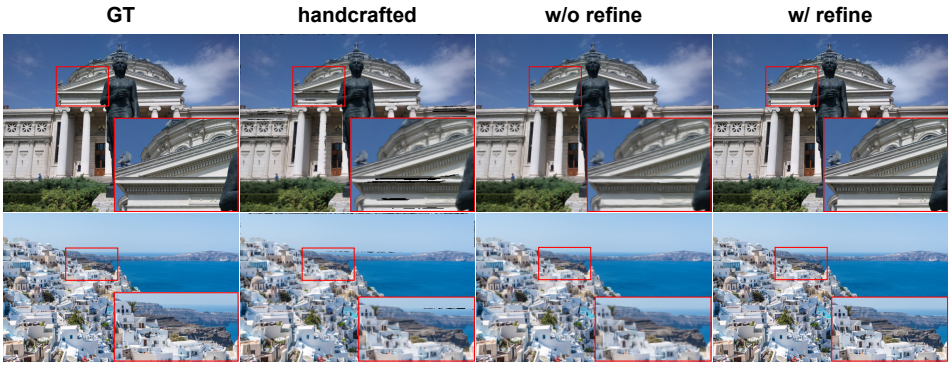

Figure 4: Ablation results. Handcrafted merging produces dark gaps along structural boundaries. The "w/o Refine" column uses the jointly trained merging predictor with refinement disabled at inference; the complete model restores sharper, more continuous contours.

Table 2: DIV2K validation PSNR (dB) at four budgets. The handcrafted merging result uses the greedy cost from Section 3.3; the merge-only variant disables refinement at inference after joint training; the complete model alternates merging and refinement.

Variant	N=10k	N=20k	N=50k	N=100k
handcrafted merging only	23.42	24.42	24.99	25.15
learned merging only	23.24	24.48	25.92	26.79
full model	**26.73**	**29.02**	**31.73**	**33.11**

but introduce granular artifacts along the lines and in the dark background. Image-GS produces smoother backgrounds but broadens and blurs the lines, whereas AIR retains sharper strokes with uneven widths and local streaking. EPIG more closely preserves their continuity and separation. In the Kodak facade (cols. 7–8), GaussianImage introduces colored speckle on the pale wall around the lettering. Image-GS and AIR blur the sign and markedly degrade its narrow character strokes and window-frame details, whereas our result preserves clearer lettering and architectural boundaries against a cleaner wall. More qualitative results are shown in appendix.

Ablation. Table 2 compares the handcrafted merging rule with two inference configurations of the jointly trained model on DIV2K: merging only, with the refinement network disabled at every stage, and the complete pipeline. The merging-only configuration is close to the handcrafted rule at 10k (23.24 versus 23.42 dB) and exceeds it at 20k, 50k, and 100k. Enabling refinement raises PSNR by 3.49, 4.54, 5.81, and 6.32 dB, respectively. Because both configurations use the same trained merging predictor, these gains reflect inference-time attribute corrections and their influence on subsequent merging decisions. Figure 4 shows how these variants differ locally. In the upper example, handcrafted merging creates dark horizontal gaps across the building's cornice; in the lower example, similar gaps interrupt the distant shoreline. The jointly trained merging predictor removes these gaps even when refinement is disabled at inference, but the cornice ornament and distant white buildings remain blurred. With refinement enabled, the repeated architectural details and small building boundaries become more distinct while the long edges remain continuous.

5 Conclusion

We presented EPIG, which compresses an analytic pixel-induced Gaussian field to a prescribed budget through learned contraction and rendering-guided refinement. It improves PSNR and MS-SSIM over feed-forward baselines at every tested budget on DIV2K and zero-shot Kodak. At 50k, it also leads the time-constrained comparison in all three DIV2K metrics and in Kodak PSNR and MS-SSIM. It remains slower than other feed-forward methods, while longer per-image optimization can reach higher DIV2K PSNR. Future work will reduce the cost of coarsening and refinement.

AI USE STATEMENT

In this work, we used generative AI tools (LLM-based assistants) for code development support and manuscript drafting. All AI-assisted content—including text, code, and experimental analysis—was reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide implementation details, training settings, datasets, and evaluation protocols in the main paper. The code will be publicly released upon acceptance.

REFERENCES

Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1122–1131, 2017. doi: 10.1109/CVPRW.2017.150.

Du Chen, Liyi Chen, Zhengqiang Zhang, and Lei Zhang. Generalized and efficient 2D Gaussian splatting for arbitrary-scale super-resolution. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Jiacong Chen, Qingyu Mao, Shuai Liu, Chao Li, Junhao Lin, Xiandong Meng, Fanyang Meng, and Yongsheng Liang. Entropy-aware image representation via 2D Gaussian splatting. *Signal Processing*, 244:110521, 2026a.

Jiacong Chen, Qingyu Mao, Xiandong Meng, Shuai Liu, Chao Li, Fanyang Meng, Youneng Bao, and Yongsheng Liang. Locality-aware density control for efficient gaussian-based image representation. In *ACM Multimedia (ACMMM)*, 2026b.

Yuhan Chen, Wenxuan Yu, Guofa Li, Kunyang Huang, Ying Fang, Yicui Shi, Wenbo Chu, and Keqiang Li. Dehaze-GaussianImage: Zero-shot dehazing via efficient 2D Gaussian splatting representation. *IEEE Transactions on Computational Imaging*, 2026c.

Yuhan Chen, Wenxuan Yu, Guofa Li, Yijun Xu, Ying Fang, Yicui Shi, Long Cao, Wenbo Chu, and Keqiang Li. LL-GaussianImage: Efficient image representation for zero-shot low-light enhancement with 2D Gaussian splatting. *arXiv preprint*, 2026d.

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whyu Teh, and Arnaud Doucet. COIN: Compression with implicit neural representations. In *ICLR Workshop on Neural Fields*, 2021.

Emilien Dupont, Muhammad Gondal, Auguste Gonin, Abraham Itzhakov, Yuxuan Zhang, Christos Louizos, and Max Welling. COIN++: Data agnostic neural data compression. In *CVPR Workshop*, 2022.

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Fourth Alvey Vision Conference*, pp. 147–151, 1988.

Jintong Hu, Bin Xia, Bin Chen, Wenming Yang, and Lei Zhang. GaussianSR: High fidelity 2D Gaussian splatting for arbitrary-scale image super-resolution. In *NeurIPS*, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (SIGGRAPH)*, 42(4), 2023.

MyoungGon Kim, JeongHyeon Ahn, Seohyeon Park, Hyemi Kim, Seunghyun Park, Jung Ho Hwang, and JungHyun Han. EllipssianNet: Image-guided sampling of 2D gaussians for Gaussian splatting. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2026.

Kodak. Kodak lossless true color image suite. `https://r0k.us/graphics/kodak/`, 1999.

Inseo Lee, Youngyoon Choi, and Joonseok Lee. GaussianVideo: Efficient video representation and compression by Gaussian splatting. In *CVPR Workshops*, 2025.

Linfei Li, Lin Zhang, Zhong Wang, and Ying Shen. SmartSplat: Feature-smart gaussians for scalable compression of ultra-high-resolution images. In *AAAI Conference on Artificial Intelligence*, 2026a.

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 *AAAI Conference on Artificial Intelligence*, 2026b.

Huanxiong Liang, Yunuo Chen, Yicheng Pan, Sixian Wang, Jincheng Dai, Guo Lu, and Wenjun Zhang. Structure-guided allocation of 2D gaussians for image representation and compression, 2025.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 2022.

Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G. Baraniuk. WIRE: Wavelet implicit neural representations. In *ICML*, 2023.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.

Lo-Wei Tai, Ching-En Li, Cheng-Lin Chen, Chih-Jung Tsai, Hwann-Tzong Chen, and Tyng-Luh Liu. EigenGS representation: From eigenspace to Gaussian image space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Masaya Takabe, Hiroshi Watanabe, Sujun Hong, Tomohiro Ikai, Zheming Fan, Ryo Ishimoto, Kakeru Sugimoto, and Ruri Imichi. Contour information-aware 2D Gaussian splatting for image representation. In *International Conference on Consumer Electronics (ICCE)*, 2026.

Zunian Wan, Jiancheng Zhao, Lingfeng Zhang, et al. Spectrum-adaptive distribution of 2D gaussians for image representation and compression. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2025.

Hao Wang, Ashish Bastola, Chaoyi Zhou, Wenhui Zhu, Xiwen Chen, Xuanzhao Dong, Siyu Huang, and Abolfazl Razi. Fast 2DGS: Efficient image representation with deep Gaussian prior, 2025a.

Longan Wang, Yuang Shi, and Wei Tsang Ooi. GSVC: Efficient video representation and compression through 2D Gaussian splatting. In *ACM Multimedia Systems Conference (MMSys) / NOSSDAV*, 2025b.

Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multi-scale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems, and Computers*, volume 2, pp. 1398–1402, 2003.

Zhaojie Zeng, Yuesong Wang, Chao Yang, Tao Guan, and Lili Ju. Instant GaussianImage: A generalizable and self-adaptive image representation via 2D Gaussian splatting. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Zhaojie Zeng, Yuesong Wang, Yawei Luo, and Tao Guan. AIR: Amortized image reconstruction framework for self-supervised feed-forward 2D Gaussian splatting, 2026.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

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 *European Conference on Computer Vision (ECCV)*, 2024.

Yunxiang Zhang, Bingxian 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 *ACM SIGGRAPH Conference Papers*, 2025. doi: 10.1145/3721238.3730596.

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, 2025.

A Implementation Details

A.1 Network architecture

Merging predictor. The merging predictor follows the graph-attention design of Section 3.3. Node descriptors are embedded by a linear projection and processed by two graph-attention layers (width 96, four attention heads), where attention is computed over an eight-neighbor table with self-loops, implemented as segment softmax over the edge list. The readout is a two-layer perceptron (hidden width 64) over the concatenation $[h_a, h_b, h_a \odot h_b, e_{ab}]$. The network has 0.17M parameters, about $600\times$ fewer than the refinement network, and is evaluated at every coarsening stage.

Refinement network. The frozen encoder $\mathcal{B}$ combines the first three stages of ResNet-18 (64, 64, and 128 channels) with a DINOv2 ViT-B/14 encoder (768-dim patch tokens, patch size 14). Multi-scale features of the input image are pooled with Gaussian weights over each primitive's footprint to initialize the recurrent state. The error features of Eq. 7 use the multi-layer ResNet difference, projected by ϕ_f, and the pixel-space residual, rearranged into 2×2 neighborhoods and projected by ϕ_p; both projections output 256 channels. Each primitive receives one token that combines its rotation-equivariant parameterization, the error features, and the recurrent state. Tokens are updated by an eight-layer transformer (width 512, four heads, feed-forward width 2048) whose attention is restricted to k=16 spatial nearest neighbors and biased by appearance and orientation similarity (weight 0.5). A zero-initialized update head maps each token to the attribute increments of Eq. 9. The refinement network has 109M parameters, of which the frozen encoder accounts for the majority; only the projections, transformer, and update head are trained.

A.2 Input descriptors

The node descriptor of primitive i has 12 dimensions: normalized center coordinates, $\log \lambda_1$, $\log \lambda_2$, $\cos 2\theta$, $\sin 2\theta$, the three color channels, $\log(\text{area}+1)$, $\log(\text{grad}+1)$, and $\log w_i$, where the first eight encode Gaussian attributes and the last four encode additive mass and regional statistics. The edge descriptor e_{ab} has 21 dimensions: the center offset, the covariance parameterizations and colors of both endpoints, their logarithmic statistics, and the budget scalar $\log(n_t/N)$, which together cover the individual cost components of $C(a, b)$ (color difference, gradient energy, and the ingredients of the Mahalanobis offset).

A.3 Training procedure

Training follows the three stages of Section 3.4. First, the handcrafted rule is run on the training images, and candidate edges are labeled by whether the rule leaves the pair in the same region at convergence; the merge predictor is distilled on these binary targets for 12 epochs. Second, the refinement network is pretrained for 20,000 steps (learning rate 2×10^{-4}) on initializations produced by the distilled predictor at a fixed budget, followed by 20,000 adaptation steps (learning rate 5×10^{-5}) on a mixture of variable-budget initializations. Third, both networks are trained jointly for 60,000 steps with the hyperparameters listed in the experimental setup; each step samples a random image and budget, descends to the sampled stage without gradients while replicating the deployment trajectory, and applies one gradient-updated refinement step through the straight-through surrogate. All stages use the 800 DIV2K training images without augmentation. The joint stage takes about one and a half days on a single RTX 3090 GPU.

A.4 Inference and numerical details

At inference, coarsening proceeds without refinement above an entry count of 150k primitives and interleaves exactly one refinement step per stage below it; stages repeat until the requested budget is reached (4–8 rounds in practice). Merged covariances are re-projected onto the positive-definite cone by an eigendecomposition with eigenvalues clamped at 10^{-6}. Refinement updates are clamped to keep eigenvalues within $[10^{-4}, 0.1 \max(H, W)]$, opacities within $[0.1, 1]$, and colors within $[0, 1]$. Rendering normalizes accumulated opacity with an epsilon of 10^{-6}.

B Additional Qualitative Results

Figures 5 and 6 show additional reconstructions at a budget of 50k primitives on DIV2K validation images and Kodak images, respectively. The optimization-based methods are fitted separately to each test image on both datasets. In contrast, the feed-forward methods (Fast-2DGS, AIR, and EPIG) are trained on DIV2K and evaluated on Kodak without further training or image-specific optimization. Across both datasets, EPIG preserves fine structures and sharp boundaries while producing cleaner, more coherent regions than the compared feed-forward methods. The Kodak examples further show that these visual improvements carry over to unseen images without test-time optimization.

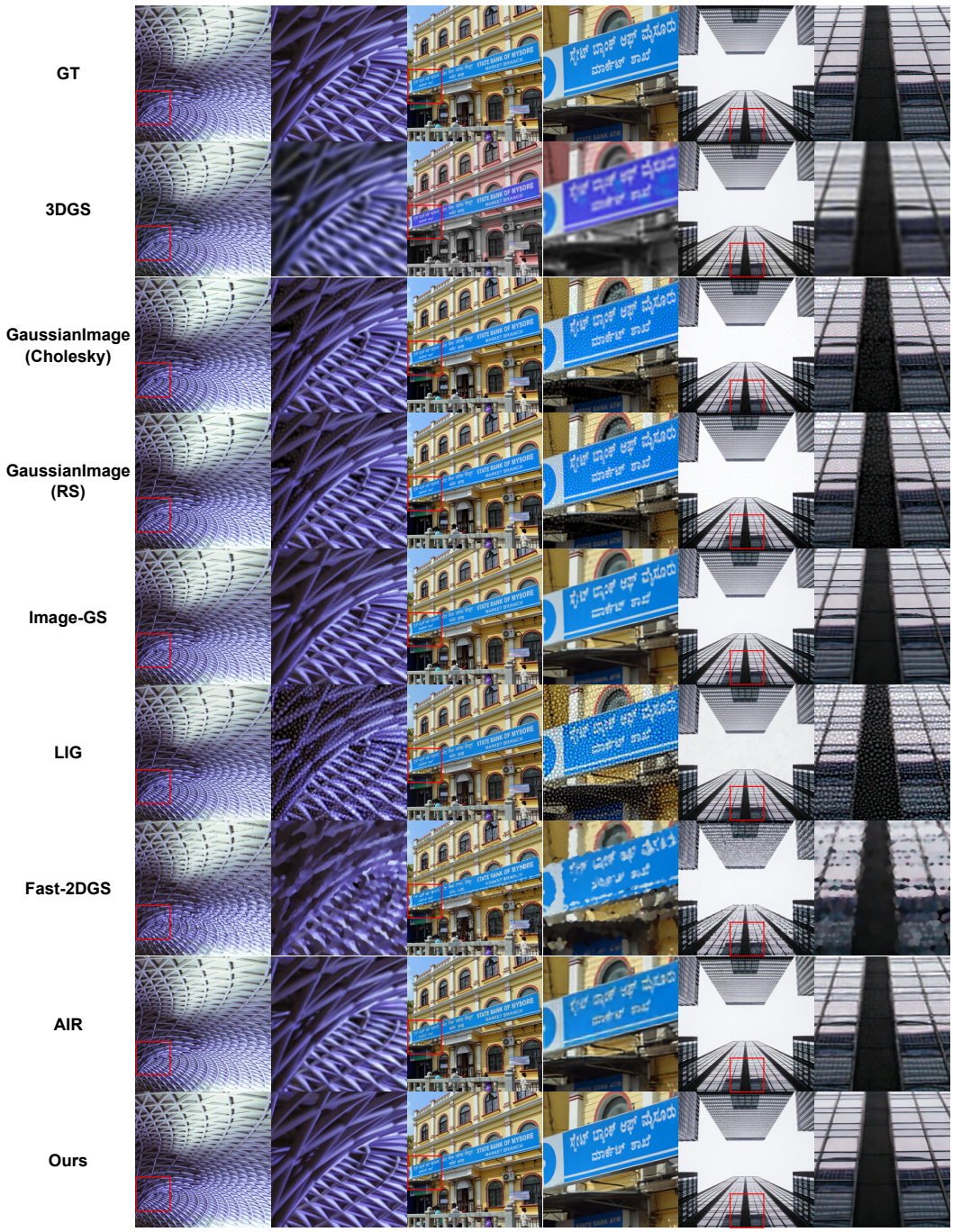

Figure 5: Additional qualitative comparison on DIV2K validation at N=50k.

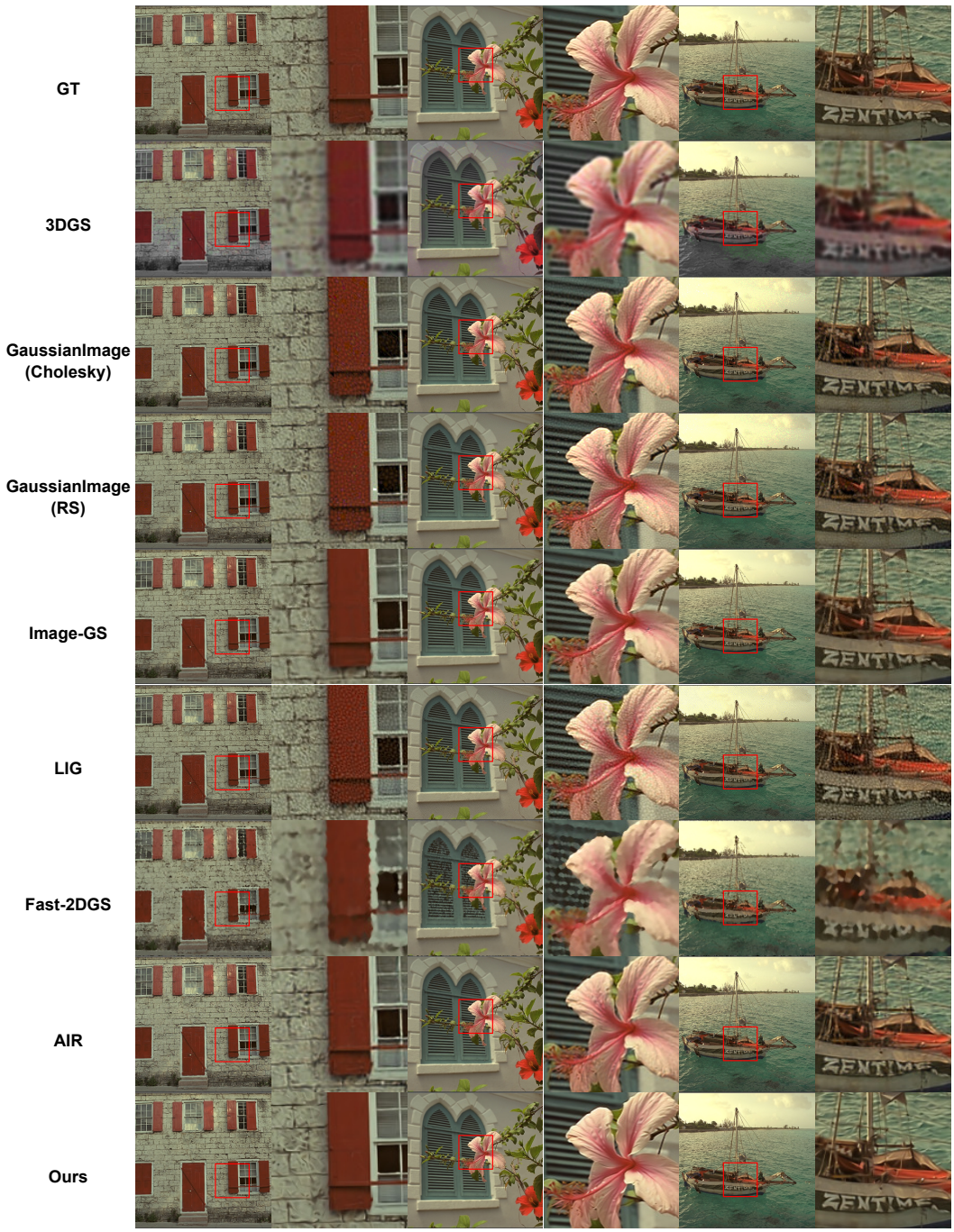

Figure 6: Additional zero-shot qualitative comparison on Kodak at N=50k.