Faster 3D Gaussian Splatting Convergence via Structure-Aware Densification

LINJIE LYU, Max-Planck-Institut für Informatik, Germany
AYUSH TEWARI, Cambridge University, United Kingdom
JIANCHUN CHEN, Max-Planck-Institut für Informatik, Germany
THOMAS LEIMKÜHLER, Max-Planck-Institut für Informatik, Germany
CHRISTIAN THEOBALT, Max-Planck-Institut für Informatik, Germany and Saarbrücken Research Center for Visual Computing, Interaction, and Artificial Intelligence (VIA), Germany

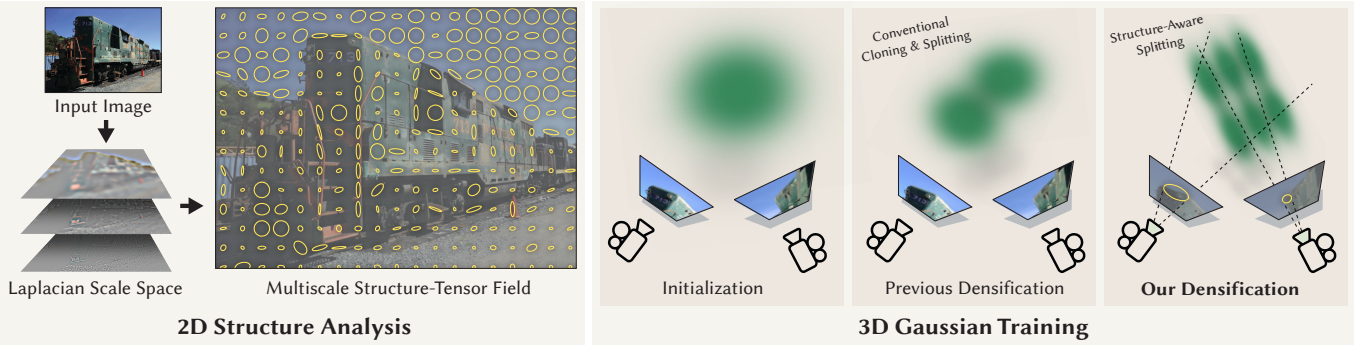

Fig. 1. Our approach addresses population control in the 3D Gaussian Splatting representation. Instead of conventional split or clone operations for Gaussians, we analyze multiscale image structure in the input views (left) and leverage this information for structure-aware anisotropic densification (right). Our method leads to significant acceleration of convergence (53 s on the Mip-NeRF360 and 41 s on the Deep Blending dataset) and achieves the best perceptual quality.

3D Gaussian Splatting has emerged as a powerful scene representation for real-time novel-view synthesis. However, its standard adaptive density control relies on screen-space positional gradients, which do not distinguish between geometric misplacement and frequency aliasing, often leading to either over-blurred high-frequency textures or inefficient over-densification. We present a structure-aware densification framework. Our key insight is that the decision to subdivide a Gaussian should be driven by an explicit comparison between its projected screen-space extent and the local structure of the texture it seeks to represent. We introduce a multi-scale frequency analysis combining structure tensors with Laplacian scale space analysis to estimate the dominant frequency at each pixel, enabling robust supervision across varying texture scales. Based on this analysis, we define η, a per-Gaussian, per-axis frequency violation metric that indicates when a primitive may be under-resolving local texture details. Unlike methods that perform isotropic splitting (e.g., splitting each Gaussian into two smaller ones with uniform shape), our approach performs anisotropic splitting. For each axis with high η, we compute a split factor to better resolve the local frequency content. We further introduce a multiview consistency criterion that aggregates η observations across multiple views. By performing densification early and faster, we skip the lengthy iterative densification phases required by baseline methods and achieve significantly faster convergence. Experiments on standard benchmarks demonstrate that our method also achieves superior reconstruction quality, particularly in high-frequency regions.

CCS Concepts: • **Computing methodologies** → **Rasterization**; **Machine learning**.

ACM Reference Format:
Linjie Lyu, Ayush Tewari, Jianchun Chen, Thomas Leimkühler, and Christian Theobalt. 2026. Faster 3D Gaussian Splatting Convergence via Structure-Aware Densification. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3799902.3811212

Authors' addresses: Linjie Lyu, llyu@mpi-inf.mpg.de, Max-Planck-Institut für Informatik, Germany; Ayush Tewari, at2164@cam.ac.uk, Cambridge University, United Kingdom; Jianchun Chen, jchen@mpi-inf.mpg.de, Max-Planck-Institut für Informatik, Germany; Thomas Leimkühler, thomas.leimkuehler@mpi-inf.mpg.de, Max-Planck-Institut für Informatik, Germany; Christian Theobalt, theobalt@mpi-inf.mpg.de, Max-Planck-Institut für Informatik, Germany and Saarbrücken Research Center for Visual Computing, Interaction, and Artificial Intelligence (VIA), Germany.

SIGGRAPH Conference Papers '26, July 19–23, 2026, Los Angeles, CA, USA

1 INTRODUCTION

3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] has emerged as an effective approach for novel-view synthesis, achieving real-time rendering with high-fidelity reconstruction. Unlike implicit neural representations such as NeRF [Mildenhall et al. 2020], 3DGS explicitly represents scenes as collections of anisotropic 3D Gaussians, enabling efficient tile-based rasterization. However, despite its rendering efficiency, training a 3DGS model typically requires 30,000 iterations that can take 10–20 minutes even on modern GPUs.

A common assumption is that the training bottleneck lies in the optimization process itself: the learning rate schedule, gradient

computation [Mallick et al. 2024], or convergence properties of the optimizer [Höllein et al. 2025; Lan et al. 2025]. However, we argue that *one of the main bottlenecks is densification*. The standard adaptive density control in 3DGS relies on a reactive, gradient-based heuristic that splits primitives only after significant rendering error has accumulated. Furthermore, an isotropic split strategy is applied, where each split operation divides one Gaussian into just two children. Regions with high-frequency details require a dense sampling of Gaussians, which can only be achieved by repeating this process many times, each round requiring hundreds of training iterations to accumulate sufficient gradient statistics before the next split can occur. As an example, consider a Gaussian that needs to resolve a texture requiring 16 times its current sampling density. Under the traditional scheme, this requires at least four successive split-and-train cycles ($2^4 = 16$), each spanning hundreds of iterations. If we could determine the required resolution analytically and densify accordingly in a single step, we could bypass the lengthy iterative densification entirely.

The gradient-based splitting criterion is fundamentally *uninformed* about the local scene structure. A Gaussian that perfectly covers a textured region but is too large to resolve fine details produces a blurry reconstruction, yet may generate small positional gradients because its *center* is already well-placed. The standard heuristic struggles to distinguish between geometric misalignment (where the Gaussian should move) and inadequate resolution (where it should split). In practice, finding an appropriate gradient threshold to trigger splitting for thin structures while avoiding over-densification elsewhere remains a challenging balancing act. Previous methods have explored advanced adaptive densification criteria [Mallick et al. 2024; Ren et al. 2026] to reduce the Gaussian count and speed up individual iterations; however, they remain fundamentally limited by the traditional isotropic splitting strategy, which fails to efficiently capture thin geometric structures and bottlenecks convergence speed through gradual densification schedules.

In this work, we propose a structure-aware densification framework that directly addresses the densification bottleneck. We introduce a supervision signal based on *structure tensors* [Di Zenzo 1986; Förstner and Gülch 1987] combined with *Laplacian scale space* analysis [Lindeberg 1994], which together enable robust estimation of the dominant frequency and orientation of image features at each pixel. This multi-scale approach identifies the characteristic frequency across multiple octaves, making the method robust to textures of varying scales.

By projecting each Gaussian's 3D axes into screen space and comparing the resulting extents against the local texture wavelength, we derive a per-axis *frequency violation metric*, η. When $\eta > 1$, the primitive may be too large to faithfully represent the local texture, indicating that subdivision could improve detail reconstruction.

Unlike previous methods that perform stochastic sampling-based splitting (typically generating children via random offsets and shrinking all axes uniformly), our approach utilizes η to perform *early analytic densification*. We calculate the number of subdivisions n required along each axis independently, generating $n_x \times n_y \times n_z$ children arranged in a regular grid. This allows our method to skip the gradual and long densification periods required by reactive gradient-based methods, ensuring that the new primitives are immediately capable of resolving the local frequency content. This leads to faster convergence and better detail reconstruction.

Furthermore, we introduce a *multiview consistency mechanism* that aggregates η-observations across multiple training views. Rather than reacting to errors from a single viewpoint, we classify each Gaussian's observations and trigger densification only when a significant fraction of views report consistent frequency violations. This prevents over-densification for non-surface Gaussians.

In summary, our contributions are summarized as follows:

(1) A **Multi-scale Frequency Analysis** utilizing structure tensors and Laplacian scale space to compute the dominant frequency at each pixel, providing robust supervision across texture scales.
(2) A **Frequency Violation Metric** that quantifies potential under-resolution by comparing projected Gaussian extent to local texture wavelength.
(3) An **Structure-aware Densification** strategy that determines the optimal per-axis subdivision factor to resolve frequency violations efficiently, enabling the model to skip lengthy densification periods.

Experiments on standard benchmarks demonstrate that our method achieves superior reconstruction quality, particularly in high-frequency regions such as fine structures, while also improving training efficiency over baseline methods. All code and data are available on our project webpage: https://vcai.mpi-inf.mpg.de/projects/SAD-GS.

2 RELATED WORK

Here, we review related works on 3DGS, focusing on acceleration techniques, densification strategies, and frequency analysis.

3DGS Acceleration. A key bottleneck of 3DGS [Kerbl et al. 2023] is the large number of Gaussians required for high-fidelity reconstruction, which increases both training time and rendering cost. Several works focus on optimizing the rasterization pipeline itself. Taming-3DGS [Mallick et al. 2024] replaces per-pixel backpropagation with per-splat parallel computation, significantly speeding up optimization and serving as a strong baseline for subsequent research. StopThePop [Radl et al. 2024] and FlashGS [Feng et al. 2025] use precise tile intersection to reduce Gaussian–tile pairs and accelerate rasterization. 3DGS-LM [Höllein et al. 2025] replaces the Adam [Kingma and Ba 2015] optimizer with the Levenberg-Marquardt algorithm for faster convergence, and 3DGS^2 [Lan et al. 2025] achieves near second-order convergence via prioritized per-kernel updates. These methods primarily target computational efficiency rather than addressing the fundamental question of *which* Gaussians to create or remove.

Gaussian Densification and Pruning. The quality of 3DGS reconstructions depends heavily on adaptive density control, which clones or splits Gaussians based on view-space positional gradients. Recent works refine this densification strategy to control primitive count. Taming-3DGS [Mallick et al. 2024] employs a budgeting mechanism using importance scores based on Gaussian-associated properties. DashGaussian [Chen et al. 2025] introduces a resolution-guided scheduler that progressively reconstructs the scene. Compact-3DGS [Lee et al. 2024] uses learnable masks to reduce redundancy. Similarly, Kheradmand et al. [2024] reformulate 3DGS optimization as a

Markov Chain Monte Carlo process to dynamically adjust Gaussian density. Nevertheless, these methods still require a prolonged densification phase when relying on the gradient-based heuristic.

Beyond densification, several methods focus on pruning to reduce Gaussian count. Mini-Splatting [Fang and Wang 2024] removes Gaussians through intersection-preserving simplification and sampling. PUP 3DGS [Hanson et al. 2025b] and Speedy-Splat [Hanson et al. 2025a] remove redundant Gaussians by computing Hessian approximations across training views. LightGaussian [Fan et al. 2024] and Compact3D [Navaneet et al. 2024] design importance scores to guide pruning. While effective at reducing primitive count, these methods may discard Gaussians needed to capture high-frequency details, degrading rendering quality. FastGS [Ren et al. 2026] introduces a multiview consistent importance score for densification and pruning. We consider this work to be our state-of-the-art baseline. FastGS is limited by a gradual densification schedule that bottlenecks convergence speed; in contrast, we demonstrate that structure-aware early densification enables significantly faster convergence.

Frequency Analysis in Radiance Fields. Aliasing is a fundamental challenge in rendering, arising when the sampling rate is insufficient to capture high-frequency scene content. In NeRF-based methods, Mip-NeRF [Barron et al. 2021] introduced cone tracing and integrated positional encoding to pre-filter the representation.

For 3D Gaussian Splatting, recent works utilize frequency analysis to guide the structural distribution of primitives. Leveraging spectral cues for efficient initialization, Zhang et al. [2025] and Meuleman et al. [2025] utilize frequency-based edge detection to guide the initial placement of primitives. Similarly, Mallick et al. [2024] integrate a frequency-based term directly into the densification logic to prioritize high-frequency details with limited memory. While this approach improves the decision of *when* to split, it still follows a conservative, uniform split into two children that does not address *how many* splits are needed to adequately resolve local structure. Conversely, methods like Mip-Splatting [Yu et al. 2024] focus on anti-aliasing by introducing low-pass filters that constrain the minimum size of Gaussian primitives based on their screen-space footprint. While effective at eliminating artifacts, this approach tends to suppress high-frequency details rather than reconstruct them.

Our approach differs fundamentally by addressing not only *when* to densify, but *how* to densify based on local structure. While Mip-Splatting *suppresses* high frequencies to match the representation, we *actively densify* the geometry to *resolve* those frequencies. By computing the dominant local frequency using structure tensor analysis [Di Zenzo 1986; Förstner and Gülch 1987] with Laplacian scale space [Lindeberg 1994] and comparing it against each Gaussian's projection, we derive an explicit *frequency violation metric*. This enables more principled densification decisions, producing sharper reconstructions in textured regions without information loss.

3 METHOD

Our approach addresses the limitations of heuristic densification in 3DGS by integrating explicit frequency analysis into the densification process. After recalling preliminaries on 3DGS and multiscale image structure analysis (Section 3.1), we introduce a structure-aware supervision signal derived from multi-scale frequency analysis (Section 3.2), a frequency violation metric (Section 3.3), and a structure-aware densification strategy that proactively subdivides under-resolved Gaussians (Section 3.5). A multiview consistency mechanism (Section 3.4) ensures robust densification decisions.

3.1 Preliminaries

3.1.1 3D Gaussian Splatting. 3DGS represents a scene using a set of anisotropic 3D Gaussians. Each Gaussian is parameterized by its position $\boldsymbol{\mu} \in \mathbb{R}^3$, covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$, opacity $\alpha \in [0,1]$, and view-dependent color. The covariance Σ is decomposed into a scaling vector $\mathbf{s} = (s_x, s_y, s_z) \in \mathbb{R}^3$ and rotation quaternion $\mathbf{q} \in \mathbb{R}^4$ (converted to a rotation matrix $R \in \mathbb{R}^{3\times3}$), yielding $\Sigma = RSS^TR^T$ where $S = \mathrm{diag}(\mathbf{s})$. This parameterization allows independent control over the size of each Gaussian along its three principal axes.

For rendering, each 3D Gaussian is projected to 2D screen space via the view transformation $W \in \mathbb{R}^{3\times3}$ and projection Jacobian $J \in \mathbb{R}^{2\times3}$. We denote the projected mean by $\boldsymbol{\mu}_{2D} \in \mathbb{R}^2$. The three principal axes of the 3D Gaussian also project to 2D, yielding axis vectors

$$\mathbf{v}_k = JWRS_k, \quad k \in \{x, y, z\}, \tag{1}$$

where S_k is the k-th column of S. Each $\mathbf{v}_k \in \mathbb{R}^2$ represents the projected extent of the Gaussian along its k-th principal axis. These vectors will be used in Section 3.3 to compare Gaussian resolution against local texture frequency.

The color of a pixel is computed by alpha blending the depth-ordered projected Gaussians that overlap the pixel. The loss $\mathcal{L}$ used to optimize the representation from posed image observations is a weighted combination of ℓ_1 and structural similarity [Wang et al. 2004] losses.

3DGS periodically densifies the representation by splitting Gaussians with high view-space positional gradients $\nabla_{\boldsymbol{\mu}_{2D}}\mathcal{L}$. However, this heuristic is reactive and lacks awareness of the local signal frequency.

3.1.2 Structure Tensor & Local Frequency Analysis. The principle of our method is to ensure that the frequency content induced by the Gaussian primitives matches the local frequency content of the scene.

To reason about local image structure, we make use of the structure tensor [Di Zenzo 1986; Förstner and Gülch 1987]. Given an image I pre-smoothed with a Gaussian kernel G_σ, the structure tensor at scale σ is a 2×2 per-pixel symmetric matrix that summarizes the local gradient distribution:

$$S_\sigma = G_\rho * (\nabla I_\sigma \nabla I_\sigma^T) \tag{2}$$

where $I_\sigma = G_\sigma * I$ is the pre-smoothed image and ρ is the integration scale. In component form, this yields

$$S_{xx} = \sum_c I_{x,c}^2, \quad S_{xy} = \sum_c I_{x,c} I_{y,c}, \quad S_{yy} = \sum_c I_{y,c}^2,$$

where $I_{x,c} = \nabla_x(G_\sigma * I_c)$ and $I_{y,c} = \nabla_y(G_\sigma * I_c)$ are the smoothed gradients of color channel c, following the multi-channel formulation of Di Zenzo [1986]. S_{xx} measures gradient energy in the horizontal direction, S_{yy} in the vertical direction, and S_{xy} captures

the cross-correlation between directions, encoding orientation. The eigenvalues λ_1, λ_2 of S_σ represent gradient energy along principal directions of local orientation: high eigenvalues indicate regions with significant high-frequency content such as edges.

At a fixed analysis scale σ, the energy of the structure tensor reflects both image contrast and spatial frequency content. The dependence on contrast is immediate, since S_σ is constructed from image gradients. To illustrate the role of frequency, consider a sinusoidal signal $\sin(2\pi\omega x)$: its derivative is $2\pi\omega\cos(2\pi\omega x)$, so the gradient magnitude scales as $|I_x| \propto \omega$, and the corresponding tensor entry satisfies $S_{xx} \propto \omega^2$. In practice, pre-smoothing with G_σ acts as a low-pass filter that attenuates frequencies above $\omega \sim 1/(2\pi\sigma)$. Consequently, the trace $\mathrm{tr}(S) = S_{xx} + S_{yy}$ captures the band-limited gradient energy, aggregating contributions from a range of spatial frequencies with higher frequencies contributing more strongly (up to the cutoff induced by σ). This relationship underlies our multi-scale analysis (Section 3.2).

3.2 Multi-scale Structure Tensor Analysis

To estimate local frequency, we leverage the structure tensor (Section 3.1.2) in combination with Laplacian scale-space analysis [Lindeberg 1994; Scheunders 2002]. The structure tensor captures local gradient orientation and magnitude, but by itself does not indicate the spatial scale at which these gradients are most prominent. Since real scenes contain structures at multiple frequencies – ranging from fine texture to coarse edges – a single-scale tensor cannot distinguish between them. We therefore introduce scale explicitly by analyzing image structure across scales.

Specifically, given an RGB image I, we construct a Gaussian scale space using Gaussian blur kernels G_{σ_l} with increasing standard deviations $\sigma_l = 1.5^l$, for $l \in 0, \ldots, L$ (we use $L = 4$ in practice). We maintain full spatial resolution at all levels, yielding a set of images $\{I_l\}$. Unlike image pyramids that perform additional downsampling [Lowe 2004], this formulation preserves pixel correspondences across scales. At each level l, we compute per-pixel structure tensors S_l using an integration scale $\rho_l = 3\sigma_l$. The structure tensor S_l encodes the combined effects of image contrast, band-limited frequency content, as well as orientation and anisotropy (Section 3.1.2), which we aim to disentangle.

First, to remove the influence of overall signal magnitude, we normalize the tensor:

$$\hat{S}_l = \frac{S_l}{\mathrm{tr}(S_l) + \epsilon}, \tag{3}$$

where $\epsilon > 0$ ensures numerical stability. This normalization suppresses contrast- and frequency-dependent scaling captured by $\mathrm{tr}(S_l)$, and emphasizes orientation and anisotropy.

Next, we estimate per-scale texture energy using a Laplacian scale-space formulation. The difference between adjacent blur levels acts as a band-pass filter that isolates image content within a specific frequency band. We define the corresponding energy as

$$E_l = \|I_{l-1} - I_l\|_2, \tag{4}$$

computed per pixel. High values of E_l indicate that significant texture energy is present at the corresponding scale.

Finally, we aggregate the structure tensors, weighted by their corresponding energy and modulated by the squared frequency:

$$\bar{S} = \frac{\sum_{l=0}^{L} E_l^\gamma \, \omega_l^2 \, \hat{S}_l}{\sum_{l=0}^{L} E_l^\gamma + \epsilon}, \tag{5}$$

where γ controls the sharpness of the weighting, emphasizing scales with higher energy; it was chosen empirically and set to 3.0. This aggregation emphasizes scales at which the local structure exhibits strong responses, reflecting both the dominant frequency content and the corresponding orientation and anisotropy. The resulting tensor map $\bar{S}$ provides scale-adaptive guidance for structure-aware densification. Fig. 1 (left panel) visualizes $\bar{S}^{-1}$ as ellipses.

3.3 Frequency Violation Metric

We propose a per-Gaussian frequency violation metric η that compares the size of projected scene Gaussians to the local texture wavelength observed in the input images, as estimated by the multi-scale structure tensor $\bar{S}$ introduced above, thereby providing a more principled densification criterion than previous heuristics.

To associate each scene Gaussian with a structure tensor from a training image, we uniformly sample $\bar{S}$ within the Gaussian's image-space footprint, average the samples to obtain a representative tensor, and then compute its principal eigenvalue λ_1. The minimum local wavelength is given by $\Lambda_{\min} = 1/(\sqrt{\lambda_1} + \epsilon)$, representing the smallest spatial scale of detail that can be reliably captured within the Gaussian's region. Higher eigenvalues correspond to more rapid spatial variation and thus finer texture.

Our frequency violation metric is defined as the ratio of the magnitudes of the Gaussian's projected principal axis vectors $\mathbf{v}_k$ (Eq. (1)) to $\Lambda_{\min}$:

$$\boxed{\eta_k = \frac{\|\mathbf{v}_k\|_2}{\Lambda_{\min}}.} \tag{6}$$

When $\eta > 1$, the Gaussian's projected extent exceeds the local texture wavelength, indicating potential under-resolution. Geometrically, this implies that a single Gaussian cannot accurately reconstruct higher-frequency spatial variations. This frequency violation directly correlates with reconstruction error and guides our adaptive densification strategy. The three-dimensional violation vector $\boldsymbol{\eta} = (\eta_x, \eta_y, \eta_z)$ enables anisotropic densification.

An alternative formulation projects the aggregated structure tensor onto each axis direction:

$$\eta_k^{(\mathrm{proj})} = \sqrt{S_{xx}u_k^2 + 2S_{xy}u_k v_k + S_{yy}v_k^2}, \tag{7}$$

where (u_k, v_k) are the components of $\mathbf{v}_k$ from Eq. (1). This approach provides a stricter anisotropic criterion where each axis is evaluated against its directional frequency content. However, as shown in Section 4.3, this alternative yields lower-quality view synthesis results.

3.4 Multiview Consistency

Single-view η measurements may be noisy due to occlusion or view-dependent effects. To avoid densifying Gaussians that are occluded or located far from visible surfaces, we aggregate observations across training views, maintaining per-axis counts N for high ($\eta > 1$) and

low ($\eta < 0.1$) responses, along with the maximum observed response $\eta_{\max}$.

Densification is triggered only when a significant fraction of views report consistent violations, i.e., $\frac{N_{\text{high}}}{N_{\text{total}}} > \tau_{\text{split}}$.

This ensures decisions are robust to transient errors. Similarly, we prune Gaussians that consistently report low η across views (indicating they cover smooth regions where smaller primitives are not needed) and have low opacity, i.e., $\frac{N_{\text{low}}}{N_{\text{total}}} > \tau_{\text{prune}}$ and $\alpha < \tau_\alpha$.

We set the thresholds $\tau_{\text{split}} = \tau_{\text{prune}} = 0.8$ and $\tau_\alpha = 0.1$ in all our experiments.

3.5 Structure-aware Densification

Most previous methods [Kerbl et al. 2023; Mallick et al. 2024; Ren et al. 2026] split each Gaussian into two children with uniform shrinkage, requiring multiple iterations to achieve fine resolution. In addition, uniform shrinkage struggles with thin structures, as it fails to adapt the shape of anisotropic Gaussians. Our structure-aware densification instead determines the split factor directly from $\eta_{\max}$ (we use η throughout this subsection to avoid notational clutter).

Recall that η_k quantifies the factor by which the Gaussian's extent exceeds the local wavelength along the k-th principal direction. Ideally, the Gaussian should be split into approximately η_k children along axis k so that each child matches the local frequency content. However, directly setting the split factor to $n = \lceil \eta \rceil$ can lead to excessive splitting when η is large or noisy. To improve robustness and to maintain memory efficiency, we instead employ a concave mapping that compresses η in the high-value regime. As analyzed in Section 4.3, we found empirically that the concave function

$$n = \lceil \sqrt{\eta} \rceil \tag{8}$$

performs best among alternatives. We compute n independently for each dimension and generate $n_x \times n_y \times n_z$ Gaussian children arranged on a regular grid with positions

$$\boldsymbol{\mu}_{\text{child}}^{(i,j,k)} = \boldsymbol{\mu}_{\text{parent}} + R_{\text{parent}} \cdot (\mathbf{s}_{\text{parent}} \odot \mathbf{g}^{(i,j,k)}), \tag{9}$$

where R_{parent} is the parent primitive's rotation matrix, $\mathbf{s}_{\text{parent}}$ denotes the parent's scaling vector, $\mathbf{g}^{(i,j,k)}$ represents the coordinates corresponding to indices i, j, k on a regular grid in the unit cube with resolution $n_x \times n_y \times n_z$, and $\odot$ denotes element-wise multiplication. The children's scaling vectors are defined as $\mathbf{s}_{\text{child}} = \mathbf{s}_{\text{parent}} \oslash (n_x, n_y, n_z)$, where $\oslash$ denotes element-wise division. Our approach allows reaching the necessary resolution in fewer densification cycles while avoiding excessive memory growth, thus handing over the job to gradient-based optimizers at an earlier stage for faster convergence. Figure 3 illustrates the difference between conventional densification and our approach.

3.6 Training

Our method integrates into the standard 3DGS training loop with minimal overhead. Beyond the standard structure-from-motion initialization, we additionally place Gaussians on the faces of the scene's bounding box to ensure complete geometric coverage. The multi-scale structure tensors $\bar{S}$ are precomputed for each training image using vectorized GPU operations, incurring a one-time overhead of 0.7 seconds and an additional memory cost equal to the input images' footprint.

During training, we perform online accumulation of frequency statistics: in each iteration, for visible Gaussians, we reuse the 2D covariance and intermediate buffers from the forward rendering pass to sample one point within each Gaussian's footprint and compute η. This avoids additional projection or sampling overhead. We use an additional ℓ_2 loss for faster convergence. Our structure-aware densification occurs every 500 iterations. In addition, we apply the densification strategy of AbsGS [Ye et al. 2024] every 100 iterations, primarily to populate empty regions with primitives so that our structure-aware densification can fully leverage the existing primitives.

4 EXPERIMENTS

4.1 Experimental Setup

We evaluate our method on three widely used benchmark datasets. Mip-NeRF 360 [Barron et al. 2022] features unbounded indoor and outdoor scenes with complex central objects and multi-scale backgrounds. Deep Blending [Hedman et al. 2018] contains various indoor environments captured for image-based rendering. Tanks & Temples [Knapitsch et al. 2017] includes large-scale outdoor scenes with challenging geometric complexity and a substantially larger number of training views. We report PSNR, SSIM [Wang et al. 2004], and LPIPS [Zhang et al. 2018] metrics on novel views, along with the number of Gaussians (N_{GS}) to assess model compactness and total training time to measure optimization efficiency.

Following common practice in fast 3DGS methods such as FastGS [Ren et al. 2026] and 3DGS-LM [Höllein et al. 2025] that employ different configurations for various scene types, we apply dataset-dependent training settings tailored to their characteristics. We employ batch training with scene-dependent batch sizes: a batch size of 2 for indoor scenes from Mip-NeRF 360 and Deep Blending, and a batch size of 1 for outdoor scenes. The larger batch size for indoor scenes improves coverage of background regions across viewpoints within our limited iteration budget; while a batch size of 1 typically reconstructs foreground geometry well, it leads to noticeably lower quality in background regions. Although increasing the batch size to 2 doubles the per-iteration training time, we still observe a significant speed-up in convergence compared to baseline methods. We further use different training iterations depending on the dataset characteristics: 3k iterations for Mip-NeRF360 and Deep Blending, and 7k iterations for Tanks & Temples. The extended training for Tanks & Temples is necessary because this dataset contains significantly more training cameras and a wider camera distribution range, requiring additional optimization steps to achieve convergence. Despite these variations, our method remains substantially faster than all baselines. All timing results and quantitative numbers reported in this section, including those for our method and all baselines, were measured on a single Nvidia H100 GPU.

4.2 Comparison with Fast 3DGS Methods

Baselines. We compare against the original 3DGS [Kerbl et al. 2023] as the representative of reactive densification methods. To

Table 1. Quantitative comparisons on the Mip-NeRF360 [Barron et al. 2022], Deep Blending [Hedman et al. 2018], and Tanks & Temples [Knapitsch et al. 2017] datasets. We report optimization time (in seconds), PSNR, SSIM, LPIPS, the number of Gaussians (N_{GS}), and the number of training iterations. Our method achieves competitive or superior quality while requiring significantly fewer training iterations and substantially less training time. Note that for Mip-NeRF360 indoor scenes and all Deep Blending scenes, we use a batch size of 2 to improve background convergence, which nearly doubles the per-iteration training time; nevertheless, our method remains substantially faster than all baselines. The best, second-best, and third-best performers are highlighted.

Method	Mip-NeRF360						Deep Blending						Tanks & Temples					
	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS} ↓	It.↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS} ↓	It.↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS} ↓	It.↓
3DGS	972.9	27.54	0.813	0.221	2.63M	30k	969.0	29.75	0.903	0.241	2.46M	30k	575.0	23.68	0.849	0.171	1.57M	30k
Mini-Splat	926.7	27.37	0.821	0.217	0.53M	30k	704.0	29.97	0.907	0.243	0.56M	30k	495.5	23.41	0.843	0.182	0.30M	30k
Speedy-Splat	704.8	26.89	0.781	0.295	0.30M	30k	581.0	29.46	0.899	0.271	0.25M	30k	343.5	23.36	0.816	0.242	0.18M	30k
Taming-Bgt	277.1	27.37	0.793	0.263	0.67M	30k	174.0	29.72	0.899	0.274	0.29M	30k	172.5	23.90	0.836	0.209	0.32M	30k
Taming-Big	589.0	27.98	0.820	0.211	3.21M	30k	459.5	29.64	0.900	0.239	2.80M	30k	355.5	24.35	0.855	0.166	1.83M	30k
DashG-Base	323.9	27.70	0.814	0.217	2.16M	30k	201.0	29.68	0.900	0.250	1.99M	30k	243.5	23.95	0.840	0.178	1.33M	30k
DashG-Big	339.7	27.69	0.817	0.218	2.41M	30k	249.0	29.66	0.902	0.248	1.98M	30k	286.0	23.95	0.841	0.177	1.34M	30k
FastGS-Base	143.7	27.55	0.797	0.261	0.40M	30k	116.5	29.81	0.900	0.270	0.22M	30k	126.0	24.12	0.838	0.211	0.24M	30k
FastGS-Big	208.9	27.96	0.820	0.216	1.16M	30k	141.5	30.17	0.907	0.243	0.65M	30k	156.0	24.41	0.855	0.175	0.54M	30k
Ours	52.96	27.25	0.821	0.197	4.05M	3k	41.22	29.89	0.910	0.238	1.81M	3k	84.44	23.60	0.857	0.147	1.86M	7k

evaluate our method's efficiency against other acceleration strategies, we also compare against state-of-the-art fast 3DGS variants, including Mini-Splatting [Fang and Wang 2024], Speedy-Splat [Hanson et al. 2025a], Taming-3DGS [Mallick et al. 2024], DashGaussian [Chen et al. 2025], and FastGS [Ren et al. 2026]. For the fast variants, we include both their base and large/high-quality configurations where applicable to provide a comprehensive comparison.

Quantitative Results. As shown in Table 1, our method achieves the fastest training time while simultaneously obtaining the best SSIM and LPIPS scores across all three datasets, with competitive PSNR. On Mip-NeRF360, we complete training in 53 seconds (18× faster than 3DGS, 2.7× faster than FastGS-Base). On Deep Blending, we finish in 41 seconds (23× faster than 3DGS, 2.8× faster than FastGS-Base). On Tanks & Temples, we complete in 84 seconds (6.8× faster than 3DGS, 1.5× faster than FastGS-Base). Compared to FastGS-Big, which achieves the second-best LPIPS, our method improves LPIPS by 9% on Mip-NeRF360 while being 4× faster, and by 16% on Tanks & Temples with nearly 2× speedup. While FastGS-Base is the previously fastest baseline, we outperform its convergence speed by 2.7× with 25% better LPIPS on Mip-NeRF360, and by 1.5× with 30% better LPIPS on Tanks & Temples. These results demonstrate that our structure-aware densification not only accelerates training but also leads to superior perceptual quality.

Convergence Analysis. Figure 2 illustrates the image quality metrics as a function of training time. We include FastGS-Base-30k and FastGS-Big-30k as baselines using the official FastGS implementation. FastGS is the state of the art on fast convergence of 3DGS; other baselines are omitted from this experiment due to the significant speed gap between their methods and FastGS (as demonstrated by FastGS). To validate that our speedup is not simply caused by position learning rate tuning (as shown in our ablation study), but actually stems from our structure-aware densification strategy, we additionally evaluate FastGS-Base-7k and FastGS-Big-7k variants. We run these variants for 7k iterations using their official configurations, with the only modification being the use of the same accelerated position learning rate schedule as ours (`position_lr_max_steps` set to 3k for Mip-NeRF360 and Deep Blending, 7k for Tanks & Temples). Our method converges significantly faster than all baselines, faithfully capturing high-frequency details after only 1–2 densification splits. On Mip-NeRF360 and Deep Blending, we reach reasonable quality as early as 30 seconds, which is reflected in the LPIPS curves. This rapid convergence stems from our structure-aware densification strategy, which enables the representation to reach the necessary resolution early in the optimization process, bypassing the lengthy gradual densification stages characteristic of reactive gradient-based methods. While the FastGS-Base-7k variant shows faster early convergence than its 30k counterpart, it still suffers from the limitations of heuristic densification, as reflected in its blurry texture reconstructions in Figure 2. Notably, even with 30k iterations and significantly longer training time, FastGS-Big still fails to reach our LPIPS performance, demonstrating that our strategy captures high-frequency details both faster and more effectively.

Qualitative Comparison. Figure 4 presents visual comparisons across multiple scenes. Our method reconstructs high-frequency regions substantially better than competing approaches, particularly for fine-grained structures such as grass, leaves, and intricate textures. This improvement is directly attributable to our structure-aware densification strategy, which explicitly aligns Gaussian primitives with local image structures based on frequency analysis. While other methods struggle to resolve these detailed regions even after 30k iterations, our approach captures them accurately within 1.5 minutes. For more qualitative results and an analysis of rendering speed and peak training VRAM, please refer to the supplemental.

4.3 Ablation Study

We analyze the contribution of several components through controlled experiments, all listed in Table 2.

η vs. $\eta^{(proj)}$. The projection-based metric $\eta^{(\text{proj})}$ (Eq. (7)) applies a stricter three-channel anisotropic criterion where each axis is evaluated against its directional frequency content. In contrast, our metric η (Eq. (6)) compares all axes against the same scalar wavelength,

Table 2. Ablations. We compare our full method against three variants: (1) $\eta^{(\text{proj})}$, using a projection-based frequency violation metric; (2) *30k LR Schedule*, using the original 30k position learning rate schedule instead of our accelerated schedule; (3) *w/o MV Consistency*, disabling the multiview consistency mechanism.

Method	Mip-NeRF360					Deep Blending					Tanks & Temples				
	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS} ↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS} ↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS} ↓
$\eta^{(\text{proj})}$	46.97	27.20	0.819	0.203	3.18M	38.17	29.89	0.910	0.242	1.35M	70.59	23.56	0.855	0.159	1.86M
30k LR Schedule	52.75	26.01	0.766	0.261	3.92M	40.38	28.39	0.878	0.308	1.59M	79.64	22.94	0.815	0.203	2.28M
w/o MV Consistency	56.48	27.30	0.822	0.196	4.56M	41.77	29.80	0.910	0.237	2.30M	89.94	23.46	0.857	0.146	2.84M
Ours	52.96	27.25	0.821	0.197	4.05M	41.22	29.89	0.910	0.238	1.81M	84.44	23.60	0.857	0.147	1.86M

making it more likely to trigger densification on any axis that exceeds this global limit. While $\eta^{(\text{proj})}$ is theoretically more precise, we observe that η achieves better LPIPS scores in practice. We hypothesize that during the computation of η, the Gaussian rotations are not yet well-optimized, leading to inaccurate axis-direction alignment. The scalar wavelength approach compensates for this by applying a more uniform densification criterion that does not depend on correct axis orientation, giving the gradient-based optimization more capacity to subsequently fine-tune both rotation and position.

Position Learning Rate Schedule. Tuning the maximum steps for the position learning rate scheduler to match our accelerated training regime is essential. Using the original 30k schedule with our reduced training budget results in significant quality degradation across all metrics, as the position learning rate remains too high throughout most of training, preventing the Gaussians from settling into stable configurations.

Multiview Consistency. Disabling the multiview consistency mechanism leads to marginally better reconstruction quality in some cases, as the method can respond more aggressively to single-view frequency violations. However, this comes at the cost of a 12–53% increase in Gaussian count across datasets,

with correspondingly longer training times. We therefore use multiview consistency by default, as it provides a favorable trade-off between quality and model compactness.

Split Factor. Our split factor, defined in Eq. (8), employs a concave mapping to compress the raw frequency violation η and prevent excessive splitting due to outliers. To evaluate this design choice, we consider a family of mappings of the form $n = \lceil \eta^p \rceil$. Table 3 reports results for different choices of p across all scenes in the Mip-NeRF360 dataset. Training time and Gaussian count increase with p, while image quality largely saturates at $p = 0.5$ (i.e., the square root), which we adopt as our default.

Table 3. Effect of concave mappings of the split factor on quality.

p	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS} ↓
0.25	45.3	27.12	0.817	0.209	3.1M
0.5 (Ours)	53.0	27.25	0.821	0.197	4.1M
0.75	72.4	27.15	0.819	0.194	5.7M

Additional ablations—including a comparison under matched Gaussian count, the effect of extended training, and the effect of the AbsGS densification component—are provided in the supplementary material.

4.4 Limitations

PSNR Performance. While our method achieves the best SSIM and LPIPS scores across all three benchmark datasets, our PSNR is slightly lower than some competing methods. This is primarily due to our significantly reduced training iterations (3k vs. 30k in standard 3DGS). With fewer optimization steps, the Gaussian positions do not fully converge to their optimal locations, resulting in small positional offsets that manifest as high-frequency rendering errors. These sub-pixel misalignments penalize PSNR more heavily than perceptual metrics, which are more robust to minor spatial shifts. Our strong performance on SSIM and LPIPS indicates that the reconstructions are perceptually superior despite the lower PSNR.

Gaussian Count. Our frequency-aware densification strategy tends to produce a higher number of Gaussians compared to budget-constrained methods. This over-densification arises because our metric η is designed to match the *highest* local frequency within each Gaussian's footprint, ensuring that every child primitive can faithfully represent this peak frequency content. However, not all regions within a Gaussian require such fine resolution—only the edges or texture boundaries do. In practice, our stochastic jitter sampling and the conservative split factor n partially compensate for this tendency, reducing unnecessary splits.

A further limitation is that η evaluates each Gaussian in isolation, ignoring the collective contribution of overlapping primitives; as a result, densification may be triggered in regions where existing primitives already provide sufficient frequency support, contributing to over-densification.

Ideally, one would implement a progressive and edge-count-based splitting strategy that subdivides Gaussians iteratively only along detected discontinuities, yielding a more compact representation. We explored this approach but found that it leads to longer convergence times. The underlying trade-off is that gradual densification introduces new Gaussians later in training, leaving insufficient iterations for these primitives to be fully optimized. Consequently, training time is effectively wasted on oversized Gaussians if they are not split early enough. We leave the development of a more compact densification strategy that still enables early splitting—without sacrificing training efficiency—as promising future work.

Stochastic Jitter Sampling. Our frequency violation estimation relies on stochastic jitter sampling within each Gaussian's footprint to robustly estimate local frequency requirements. While this approach improves coverage and reduces sensitivity to edge alignment, it introduces minor variance across training runs. In practice,

this variance is negligible and does not significantly affect final reconstruction quality.

5 CONCLUSION

We have presented a structure-aware 3D Gaussian Splatting framework that addresses the densification bottleneck from a signal-processing perspective. Unlike previous heuristics that only identify *where* to split—requiring multiple successive cycles before reaching the necessary resolution—our frequency violation metric η reveals not just the location but also *how* to split (anisotropically, matching local feature orientation) and *how many* splits are required per axis. By leveraging multi-scale frequency analysis combining structure tensors with Laplacian scale space, our method enables early analytic densification that bypasses the lengthy iterative "split–train–split" cycles. Combined with a multiview consistency mechanism, our method achieves state-of-the-art perceptual quality across three benchmark datasets while reducing training time by up to 23× compared to the original 3DGS approach.

Several promising directions remain for future exploration. Replacing the Adam [Kingma and Ba 2015] optimizer with second-order optimization methods could further accelerate convergence by providing more accurate curvature information for position and shape parameters. Integrating adaptive budget control strategies that dynamically balance primitive count against reconstruction quality would enable deployment on memory-constrained devices while preserving our fast convergence properties. Extending our frequency-aware framework to handle dynamic scenes and temporal consistency also presents an exciting avenue for future research.

REFERENCES

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Youyu Chen, Junjun Jiang, Kui Jiang, Xiao Tang, Zhihao Li, Xianming Liu, and Yinyu Nie. 2025. DashGaussian: Optimizing 3D Gaussian Splatting in 200 Seconds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 11146–11155.

Silvano Di Zenzo. 1986. A note on the gradient of a multi-image. *Computer vision, graphics, and image processing* 33, 1 (1986), 116–125.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. 2024. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Guangchi Fang and Bing Wang. 2024. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision*. Springer, 165–181.

Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Boni Hu, Linning Xu, Zhilin Pei, Hengjie Li, Xiuhong Li, Ninghui Sun, Xingcheng Zhang, and Bo Dai. 2025. FlashGS: Efficient 3D Gaussian Splatting for Large-scale and High-resolution Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 26652–26662.

Wolfgang Förstner and Eberhard Gülch. 1987. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*.

Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. 2025a. Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 21537–21546.

Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. 2025b. Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 5949–5958.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep Blending for Free-Viewpoint Image-Based Rendering. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–15.

Lukas Höllein, Aljaž Božič, Michael Zollhöfer, and Matthias Nießner. 2025. 3dgs-lm: Faster gaussian-splatting optimization with levenberg-marquardt. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 2024. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems* 37 (2024), 80965–80986.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Lei Lan, Tianjia Shao, Zixuan Lu, Yu Zhang, Chenfanfu Jiang, and Yin Yang. 2025. 3dgs2: Near second-order converging 3d gaussian splatting. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*. 1–10.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. 2024. Compact 3D Gaussian Representation for Radiance Field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tony Lindeberg. 1994. *Scale-Space Theory in Computer Vision*. The Springer International Series in Engineering and Computer Science, Vol. 256. Springer.

David G Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60, 2 (2004), 91–110.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. 2024. Taming 3DGS: High-Quality Radiance Fields with Limited Resources. In *ACM SIGGRAPH Asia 2024 Conference Papers*.

Andreas Meuleman, Ishaan Shah, Alexandre Lanvin, Bernhard Kerbl, and George Drettakis. 2025. On-the-fly reconstruction for large-scale novel view synthesis from unposed images. *ACM Transactions on Graphics (TOG)* 44, 4 (2025).

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. 2024. CompGS: Smaller and Faster Gaussian Splatting with Vector Quantization. In *European Conference on Computer Vision (ECCV)*.

Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. 2024. Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–17.

Shiwei Ren, Tianci Wen, Yongchun Fang, and Biao Lu. 2026. FastGS: Training 3D Gaussian Splatting in 100 Seconds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Paul Scheunders. 2002. Wavelet-based enhancement and denoising using multiscale structure tensor. In *Proceedings. International Conference on Image Processing*, Vol. 3. IEEE, 569–572.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. 2024. AbsGS: Recovering Fine Details in 3D Gaussian Splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia (MM '24)*. 1053–1061. https://doi.org/10.1145/3664647.3681361

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2024. Mip-Splatting: Alias-free 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

Yunxiang Zhang, Bingxuan Li, Alexandr Kuznetsov, Akshay Jindal, Stavros Diolatzis, Kenneth Chen, Anton Sochenov, Anton Kaplanyan, and Qi Sun. 2025. Image-GS: Content-adaptive Image Representation via 2D Gaussians. In *ACM SIGGRAPH 2025 Conference Papers*.

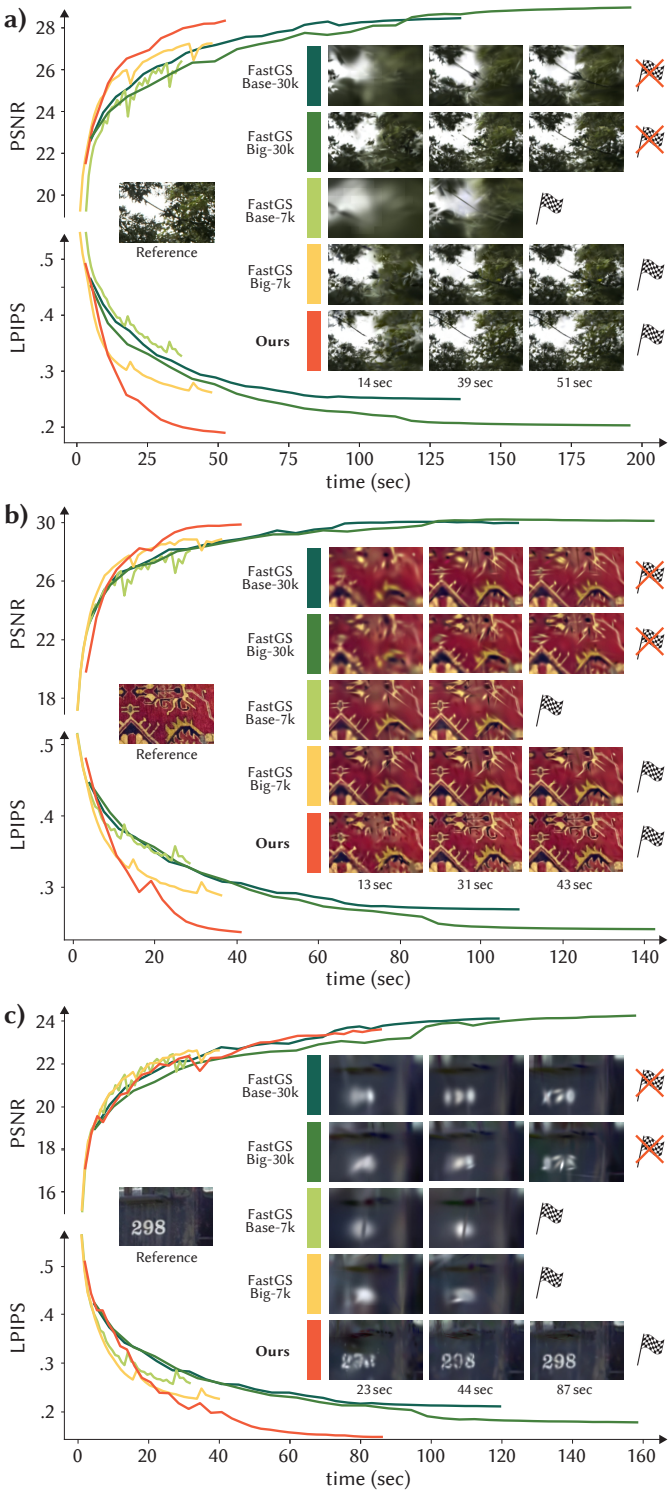

Fig. 2. Image quality versus training time, averaged over two scenes from Mip-NeRF360 (*a*), Deep Blending (*b*), and Tanks & Temples (*c*). Our approach reaches high quality significantly faster than all baselines. The flags denote results corresponding to the final state of a method, whereas crossed-out flags indicate that additional iterations are required beyond those shown.

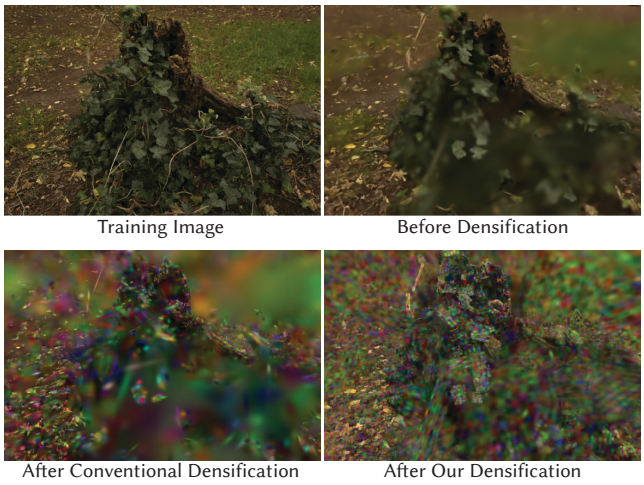

Fig. 3. Comparison of conventional densification and our approach. The top row shows a training image (left) and a rendering of a 3DGS model at an early training stage (right). The bottom row visualizes the effect of a single densification step applied to this model. To highlight individual Gaussian primitives, we overlay the renderings with random colors per Gaussian. Conventional densification (bottom left) fails to resolve high-frequency structures, requiring multiple densification stages interleaved with gradient-based optimization to achieve high fidelity. In contrast, our method (bottom right) leverages frequency information from the input image to guide anisotropic splitting, enabling the representation to capture fine structural details early in training.

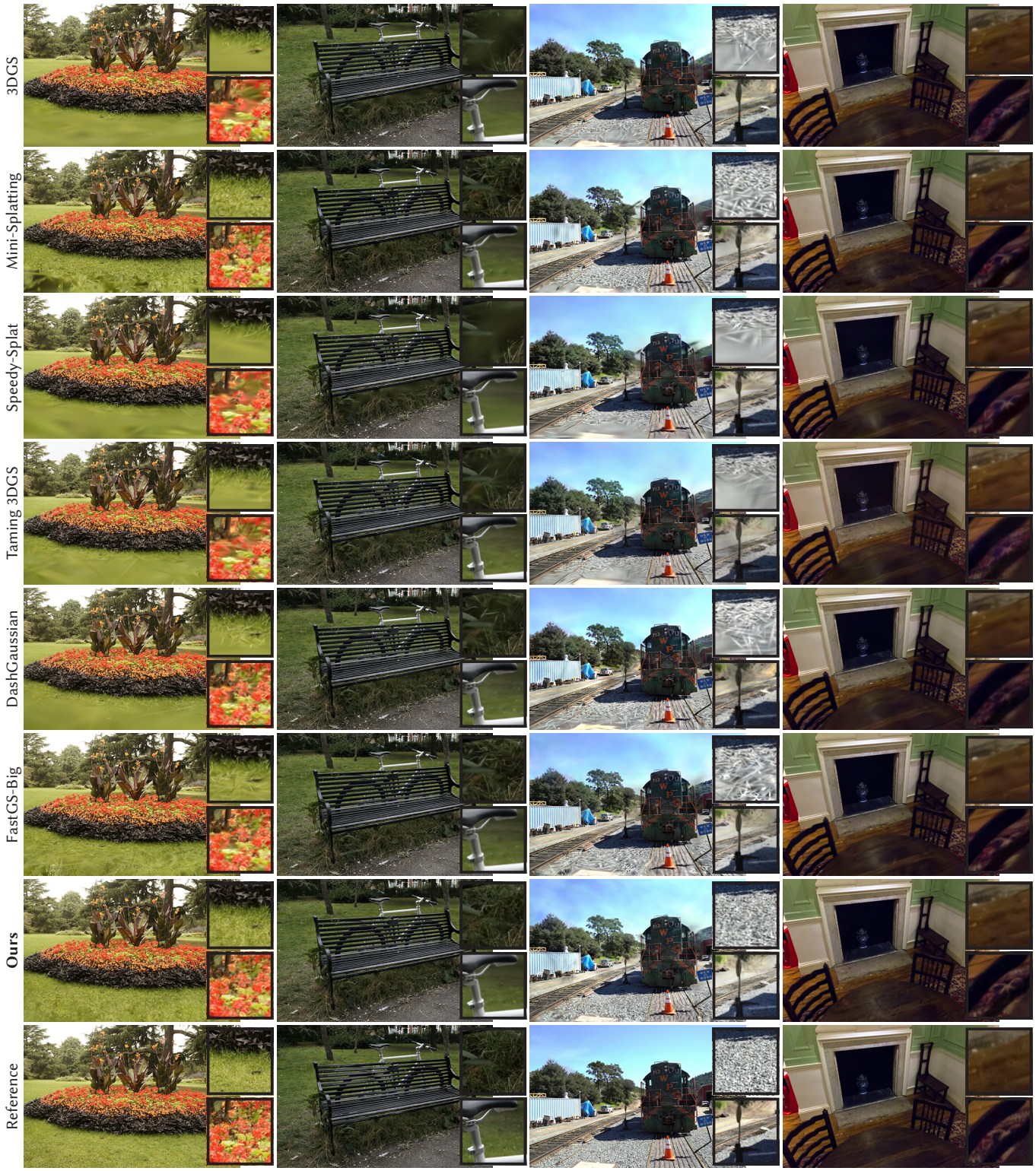

Fig. 4. Qualitative results. We compare our approach with state-of-the-art fast 3DGS methods (rows) across multiple scenes (columns). Our method achieves a 3–20× speedup while simultaneously reconstructing significantly richer high-frequency details.