

SharpGS: Sharpness-Preserving 3D Gaussian Splatting with Differentiable Blur-Driven Density Control

Moonsoo Jeong¹, Dongbeen Kim², Minseong Kim³, and

Sungkil Lee^{1,2,3,*}

¹ Department of Electrical and Computer Engineering, Sungkyunkwan University

² Department of Computer Science and Engineering, Sungkyunkwan University

³ Department of Immersive Media Engineering, Sungkyunkwan University
South Korea

{moonsoo101,rlaehdqls021,leon0106,sungkil}@skku.edu

Abstract. We present SharpGS, a differentiable blur-driven density control, which enhances the quality of 3D Gaussian Splatting (3DGS). Standard 3DGS often struggles to capture intricate details, particularly in textured patterns and object boundaries, due to the limited sensitivity of its density control. Simple finer densification can result in excessive primitive counts with marginal quality gain. To address this, we introduce blur as an effective perceptual cue, leveraging CUDA-based differentiable blur. Our density control estimates per-primitive blur levels by comparing reconstructions to blurred ground-truth images, and identifies high-frequency regions where additional primitives are required. While this naturally produces more primitives, we counterbalance this by suppressing redundant primitives. Specifically, we penalize the blur levels of inherently smooth regions such as sky, and regularize the opacities of potential split/clone candidates. We experimentally demonstrate that SharpGS greatly improves the state-of-the-art 3DGS methods in terms of quality, while keeping learned primitives compact.

Keywords: 3D Gaussian Splatting · Density Control · Differentiable Blur

1 Introduction

Novel view synthesis is one of the fundamental tasks in computer vision, virtual reality, gaming, and photorealistic rendering [30]. Neural Radiance Fields (NeRFs) [24] have set a new standard in this domain by enabling high-quality rendering from multi-view images, but its dense sampling and heavy computation limit real-time use [6]. For higher efficiency, its successors have introduced optimized data structures and sampling strategies to capture high-frequency details with less aliasing [2,3,10,15,25,38], or caching and tensor optimizations [7,11,27]. However, their efficient high-quality rendering remains challenging.

* Corresponding author.

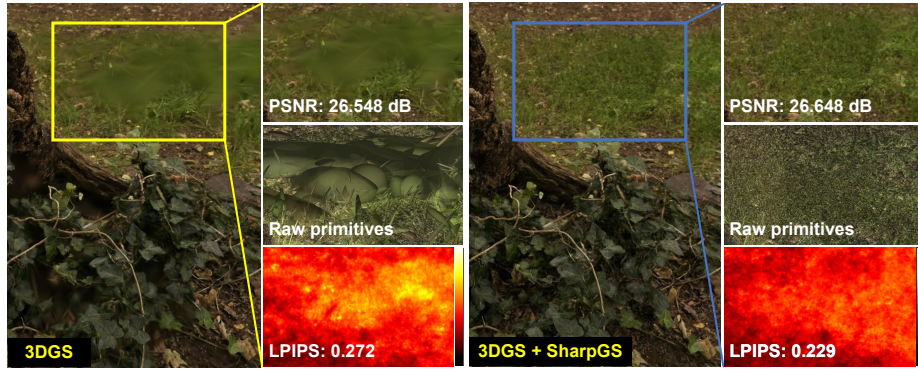


Fig. 1: While the large primitives (the yellow insets) of standard 3DGS are likely to miss details, our blur-driven density control effectively captures the intricate details, resulting in much higher quality with 4845K primitives compared to 4895K in the 3DGS.

3D Gaussian Splatting (3DGS) has recently emerged as an efficient alternative to NeRFs. On the one hand, using explicit 3D Gaussian primitives for scene representation enables bypassing computationally intensive ray marching [16]. On the other hand, 3DGS faces challenges in preserving high-frequency details, where large Gaussians tend to cover extensive areas, leading to over-reconstruction such as image blur and artifacts [9, 16, 43]. 3DGS uses an adaptive density control based on positional gradients and covariance scales to mitigate over-reconstruction, but remains insufficient in capturing intricate details. Such quality degradation has been addressed in recent successors, including anti-aliasing sampling [22, 40], deblurring [26], frequency-based regularization [42, 43], improved densification criteria [33, 37, 45], and modified primitive models [13]. Despite their significant advances, the distribution of positional gradients used in their densification often fails to reflect true details, which inspires us to explore a better alternative.

This paper presents *SharpGS*, a novel *differentiable blur-driven density control* to improve the sensitivity of the 3DGS methods to fine-grained details (see Fig. 1). Unlike the previous work, our approach does not rely solely on the distributions of positional gradients. Instead, we suggest using *blur* as an effective perceptual cue to directly assess the quality of densification from Ground-Truth (GT) images. This is inspired by our observation that under-densification locally resembles blurred GT images. To this end, we employ differentiable blur, which compares the rendering output and a *blurred* GT image, and renders their discrepancies into a *blur map*. The blur levels are defined per primitive and also *learnable* by the differentiable pipeline. Since a coarser rendering from under-densification requires blurring the GT image, we attempt to minimize the blur levels. When optimized, the rendering can represent the GT well *without blur*, implying that a high-quality densification is achieved.

Our densification naturally produces more primitives, but we counterbalance this by suppressing redundant primitives. We incorporate two regularization schemes. First, we penalize per-primitive blur levels in inherently structureless areas (e.g., sky). Such areas do not need many primitives, but often produce non-trivial false positives. This is because different levels of their blur do not make a difference in learning loss. To avoid this, we explicitly regularize them. Second, we apply *soft* opacity modulation to the primitives of high blur levels. Primitives in heterogeneous regions may have a higher chance for densification, but their densification might be redundant to higher quality. We enforce them not to be directly densified using opacity attenuation, unless they truly contribute to rendering through iteration.

Consequently, our approach significantly reduces ambiguity in density control, and does not spawn more primitives than the standard ones. Further, ours can be *synergistically* integrated to *common 3DGS methods* to improve them.

Our contributions are summarized in what follows:

- introduction of *blur* as an effective perceptual cue for enhancing 3DGS for the quality of *high-frequency* details;
- a *blur-driven density control* scheme for 3DGS, which leverages per-primitive blur for selective refinement;
- blur-adaptive regularization schemes to effectively suppress redundant false positive primitives.

2 Related work

2.1 Neural radiance fields

Neural Radiance Fields (NeRFs) [24] have pioneered high-quality novel view synthesis by representing scenes as continuous volumetric radiance fields. Due to their computational cost, subsequent works have focused on improving efficiency. Key strategies include cone-based sampling [2, 3], optimized data structures like multi-resolution grids and tensor decompositions [7, 11, 15, 25, 38], and polygon rasterization [8, 27]. Others have aimed to balance quality and speed through regularization and hybrid models [4, 31] or to improve visual fidelity with techniques such as explicit reflection modeling [12, 19, 32, 34].

2.2 3D Gaussian splatting

3DGS [16] more efficiently renders scenes via the projection of 3D Gaussians onto a 2D plane, avoiding intensive ray marching. While efficient in performance, it may suffer from lower quality including limited details, aliasing, and excessive Gaussian expansion.

Aliasing and blur have been addressed by several methods: Mip-Splatting [40] combines 3D low-pass and 2D box filtering; Analytic-Splatting [22] approximates Gaussian integrals per pixel; Mipmap-GS [21] embeds multi-scale Gaussian adaptation; SA-GS [29] applies adaptive filtering at test-time; and BAGS [26] employs multi-scale blur kernels via a blur proposal network.

Scalability towards high-resolution and performance has been also addressed. High-resolution rendering has been enhanced by capturing fine details and photorealism [5, 36, 39]. FSGS [47] leverages limited inputs to achieve real-time synthesis in sparse-view scenarios. FreGS [43] progressively incorporates frequency components in Fourier space, while Wavelet-GS [46] adopts wavelet-based frequency decomposition, both aiming to mitigate over-reconstruction artifacts. In this context, AbsGS [37] and GOF [41] address gradient collisions by using homodirectional positional gradients. FDS-GS [42] further extends this by linking Gaussian density to scale via a dynamic threshold, while Pixel-GS [45] refines gradients by considering each primitive’s pixel footprint and view-space depth. Revising-3DGS [28] replaces gradient-based density control with a pixel-error-driven alternative, and ConeGS [1] inserts new Gaussians at high-error pixels using INGP-predicted depths and pixel-cone-based scales.

Reducing memory consumption without compromising quality has been addressed by Scaffold-GS [23], GES [13], and Compact-GS [20]. Scaffold-GS organizes Gaussians around stable anchor points from sparse voxel grids, improving multi-view consistency through dynamic adjustments. GES generalizes Gaussians with flexible exponents to prune high-frequency primitives, but is limited in improving high-frequency details. Compact-GS [20] further enhances efficiency by learning to mask redundant Gaussians and compress attributes using a grid-based neural field and residual vector quantization. More recently, SteepGS [33] also achieves a more compact representation by theoretically analyzing densification and splitting primitives stalled at saddle points using second-order information.

Many existing 3DGS methods improve gradient-based density control, but the gradients are still indirect indicators to faithful densification. Unlike them, our blur-driven approach uses *blur* as a more direct indicator, since the blurred GT resembles under-densification. Further, the blur-driven regularization is efficient even in avoiding false positives. This is proved when our integration with the previous models improves quality further (see Sec. 5).

3 Preliminary: 3D Gaussian splatting

We first briefly review the core of the standard 3DGS [16]. 3DGS represents a scene using 3D Gaussians as primitives. Each Gaussian is parameterized by its mean $\mu \in \mathbb{R}^3$, covariance-like matrix $\Sigma \in \mathbb{R}^{3 \times 3}$, decomposed as $\Sigma = RSS^\top R^\top$ for rotation R and scaling S . The Gaussian function is then defined as:

$$G(x) = \exp^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)}. \quad (1)$$

For rendering, each Gaussian i is projected using the view-projection matrix, and its covariance Σ_i is approximated as a 2D covariance Σ'_i [48], yielding a 2D Gaussian $G'_i(p)$ at pixel p . The alpha is evaluated as: $\alpha_i(p) = o_i G'_i(p)$, where o_i is the opacity. The final pixel color $I_{gs}(p)$, influenced by N primitives, is computed by front-to-back alpha blending:

$$I_{gs}(p) = \sum_{i=1}^N T_i(p) \alpha_i(p) c_i, \quad (2)$$

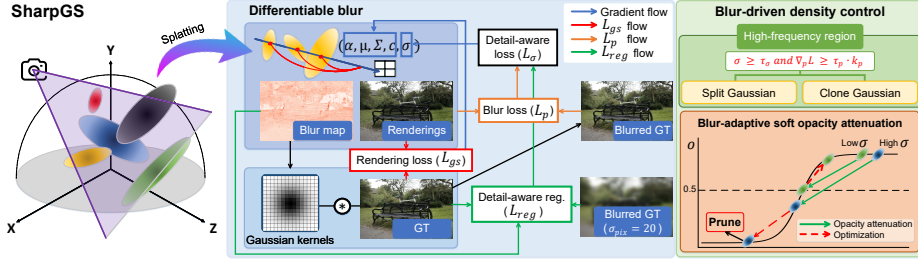


Fig. 2: Overview of our SharpGS (blur-driven density control) for 3DGS. During training, our differentiable blur pipeline (the blue block) compares renderings with the blurred GT images, and estimates per-primitive blur levels. The per-primitive blur levels are used to adaptively densify (split or clone) primitives in the blur-driven density control of the densification pass (the green block). The loss encompasses detail-aware regularization term, penalizing inherently smooth regions (e.g., sky) in addition to the standard loss terms. We also periodically apply blur-adaptive soft opacity attenuation (the orange block) to prune high-blur primitives out, when they contribute little to the rendering quality.

where c_i is the color of primitive i and the transmittance defined as: $T_i(p) = \prod_{j=1}^{i-1} (1 - \alpha_j(p))$.

The rendered image I_{gs} is trained to match the ground truth I_{gt} using a combined L_1 and SSIM loss:

$$L_{gs} = (1 - \lambda_{dssim}) \|I_{gs} - I_{gt}\|_1 + \lambda_{dssim} (1 - \text{SSIM}(I_{gs}, I_{gt})), \quad (3)$$

where λ_{dssim} is typically set to 0.2. This loss updates parameters including position, color, opacity, and covariance.

4 SharpGS: Blur-driven density control

The key idea of our density control in SharpGS is we consider the *blur* of GS renderings as a perceptual cue that identifies the degradation in reconstructed images. When Gaussians fail to capture the true details of the scene, their reconstructions are degraded and appear as blurred (see Fig. 1 for example), because Gaussians larger than needed tend to still cover the detailed regions. Our insight is that such degradation patterns can be approximately identified by blurring the GT images. To this end, our density control estimates per-primitive blur, σ , using a differentiable rendering pipeline. The training process tries to minimize the blur levels by adaptive densification. When optimized, the renderings can represent GTs without blur, indicating no missing details.

The sole application of our densification may produce more primitives than the standard gradient-based densification, requiring higher cost in storage and rendering performance. In order for ours to be also cost-effective, we introduce two regularization schemes. Firstly, we suppress redundant densifications in inherent smooth regions; their σ values do not need to be high. We derive a spatial

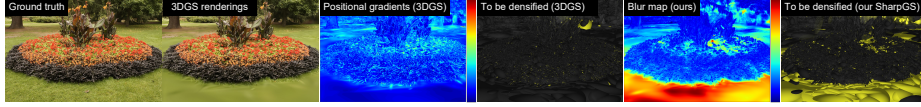


Fig. 3: Comparison of the candidates selected for densification (at 12k iterations in 3DGS). Our differentiable blur well identifies over-reconstructed regions (yellow primitives) using the blur map, whereas the previous densification misses many regions (e.g., grassy area).

prior from GT images, reflecting the degrees of per-pixel details. The prior is incorporated as a regularization term in a loss function of the differentiable blur. Secondly, we prevent unnecessary densification unless the primitives greatly contribute to reconstructions. To realize this, we propose to softly attenuate the opacities of primitives, the amounts of the attenuation of which are scaled by their blur levels to more penalize unreliable high-blur primitives. The overview of our solution is shown in Fig. 2. The following sections describe the details of our model and density control.

4.1 Blur model and rationale

Standard densification often neglects primitives in detailed regions [16] (Fig. 3), as primitives with insufficient details may exhibit near-saturated positional gradients and thus avoid densification. We instead focus on the *granularity mismatch* between GT images and renderings, prioritizing primitives whose learned details are distinct from the GT.

Frequency-domain approaches such as FreGS [43] and Wavelet-GS [46] address this issue, but often introduce excessive primitives (e.g., $1.8\times$ more than 3DGS [43]) or memory overhead. Our approach is simpler, respects perceptible cues (i.e., *blurry* appearance of under-densified primitives), and directly handles granularity mismatch in image space without Fourier transform.

To derive a blur-driven mismatch measure, we approximate GS rendering as a spatially varying convolution of GT images, because the optimization indirectly infers colors from the GT. Given a projected primitive i at its center μ'_i and screen-space support $\Omega(\Sigma'_i)$, its color c_i can be viewed as a locally weighted aggregation of GT pixels:

$$c_i = \int_{u \in \Omega(\Sigma'_i)} h_i(u) I_{\text{gt}}(\mu'_i - u) du, \quad (4)$$

where $h_i(u)$ is the contribution of pixel u to μ'_i . This can be interpreted as a gather-form view of the original scatter-based 3DGS rendering. While the scatter form is parameterized by Gaussian weights, the gather weights $h_i(u)$ require identifying contributions from all source pixels within $\Omega(\Sigma'_i)$.

Although Eq. (4) considers a single primitive at its projected center μ'_i , the same intuition extends to the final image formed by blending multiple primitives. Assuming similar projected scales among neighboring primitives, we approximate their aggregate gathering effect at arbitrary pixels p by an isotropic

2D Gaussian parameterized by $\sigma_{\text{pix}}(p)$ and kernel $H_{\sigma_{\text{pix}}(p)}$ over $\Omega(\sigma_{\text{pix}}(p))$. The GS rendering can then be approximated as

$$I_{\text{gs}}(p) \approx \hat{I}_{\text{gt}}(p) = \int_{u \in \Omega(\sigma_{\text{pix}}(p))} H_{\sigma_{\text{pix}}(p)}(u) I_{\text{gt}}(p-u) du. \quad (5)$$

This provides our key intuition: σ_{pix} reflects the degree of granularity mismatch. As $\sigma_{\text{pix}} \rightarrow 0$, the convolution model reduces to the original GT image, indicating that local details are already well reconstructed. On the other hand, larger values indicate unresolved details that may benefit from further densification, even though they do not directly measure rendering error (see Fig. 3).

4.2 Differentiable blur

To learn the per-primitive blur levels σ_i , we start from the image-space reconstruction splatted from the 3D Gaussians. The rendered reconstructions I_{gs} are compared against the *blurred* GT images $\hat{I}_{\text{gt}}$, and their granularity mismatch defines a *blur map* of the per-pixel blur levels (see Fig. 3). In our density control, the blur levels need to be learnable. To realize this, we employ a *differentiable blur*, which optimizes them via image-space optimization [18].

After 3DGS rasterization, we blur the GT images using a 2D Gaussian blur; note that the 2D Gaussian blur here is *different from the 3D Gaussian*, and is used only for blurring GT images. Given a pixel p , its blur level $\sigma_{\text{pix}}(p)$ defines the standard deviation and radius of the kernel $H(\cdot)$ in Eq. (5) as $\sigma_{\text{pix}}(p)$ and $3\sigma_{\text{pix}}(p)$, respectively. The 2D Gaussian kernel is chosen for its smooth differentiability and efficiency, which, in our experience, yields more stable gradients and lower computational overhead.

Given the per-pixel blur kernel $H_{\sigma_{\text{pix}}(p)}$, the differentiable blur is applied to a GT image I_{gt} as in Eq. (5). In our implementation, σ_i is parameterized via the sigmoid activation scaled by $k_\sigma = 20$, which is bounded to avoid excessive growths in the differentiable blur; k_σ represents the maximum level of allowed blur. To start learning with virtually no blur, σ_i is initialized with a tiny non-zero value (e.g., 0.3), resulting in a kernel radius of 0.9.

4.3 Detail-aware loss for differentiable blur

The loss of the differentiable blur, L_σ , is composed of two terms. The primary term, L_p , measures the discrepancy between the rendering I_{gs} and the blurred GT $\hat{I}_{\text{gt}}$ as:

$$L_p = (1 - \lambda_{\text{dssim}})|I_{\text{gs}} - \hat{I}_{\text{gt}}| + \lambda_{\text{dssim}}(1 - \text{ssim}(I_{\text{gs}}, \hat{I}_{\text{gt}})), \quad (6)$$

where λ_{dssim} is 0.2.

To additionally regularize σ_i in low-frequency regions (e.g., sky), we apply a spatial prior derived from the GT image, which is based on a detail-aware weight map w_d . Specifically, w_d is computed using the formulation in Eq. (6) to quantify the difference between the GT image and its heavily blurred counterpart

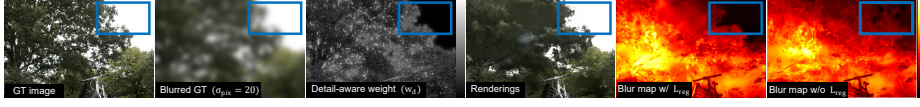


Fig. 4: Detail-aware regularization term L_{reg} suppresses large σ_{pix} values in inherently smooth regions (here, sky) using the differences of GT and its heavily blurred version.

generated with a Gaussian kernel of radius 60 (i.e., $\sigma_{\text{pix}} = 20$). Since the low-frequency areas make little difference from the heavy blur, their w_d exhibits low values. Then, we define the detail-aware regularization term as:

$$L_{\text{reg}} = (1 - w_d)\sigma_{\text{pix}}/k_{\sigma}. \quad (7)$$

As σ_{pix} is rendered from σ_i through Eq. (2), minimizing L_{reg} naturally penalizes low-detail regions. Fig. 4 visually demonstrates this effect; inherently smooth regions (e.g., sky) exhibit lower σ_{pix} values when L_{reg} is applied, indicating that our detail-aware loss effectively suppresses unnecessary blur levels where fine details are absent.

By integrating L_{reg} , the overall loss, then, is defined as:

$$L_{\sigma} = (1 - \lambda_{\sigma})L_p + \lambda_{\sigma}L_{\text{reg}}, \quad (8)$$

where λ_{σ} is set to 0.1. Gradients from L_{σ} are backpropagated exclusively to optimize the per-primitive blur σ_i via the chain rule: $\frac{\partial L_{\sigma}}{\partial \sigma_i} = \sum_{p \in \mathcal{P}_i} \frac{\partial L_{\sigma}}{\partial \sigma_{\text{pix}}(p)} \frac{\partial \sigma_{\text{pix}}(p)}{\partial \sigma_i}$, where $\mathcal{P}_i$ is the support of pixels influenced by primitive i .

By optimizing σ_i , the GT image is adaptively blurred to align with the rendered output I_{gs} . To be more precise, a primitive i of a higher σ_i value requires better densification, because it covers a region lacking details present in GT. This is demonstrated in Fig. 3. Our blur map aligns well with excessively large over-reconstructed Gaussians where standard 3DGS densification overlooks, and our differentiable blur identifies them well.

We note that, after each densification step, the σ_i values are reset to minimize bias in subsequent densification. Importantly, L_{σ} does not affect other 3DGS parameters, leaving the 3DGS loss function (Eq. (3)) unchanged; Eqs. (5) and (8) are used solely for the differentiable blur. Hence, this process is seamlessly integrated into the backward pass of the 3DGS pipeline, enabling end-to-end optimization.

4.4 Blur-adaptive soft opacity attenuation

Our detail-aware loss suppresses the densification of low-frequency primitives well, but its effect is limited only for homogeneous areas. In heterogeneous regions (where fine details and larger objects are entangled; e.g., grasses over the ground), large objects still receive high blur values σ_i , since their blur levels are blended with surrounding details. Hence, even if the details of large objects are already saturated, they are reported to be under-densified and likely to be split.

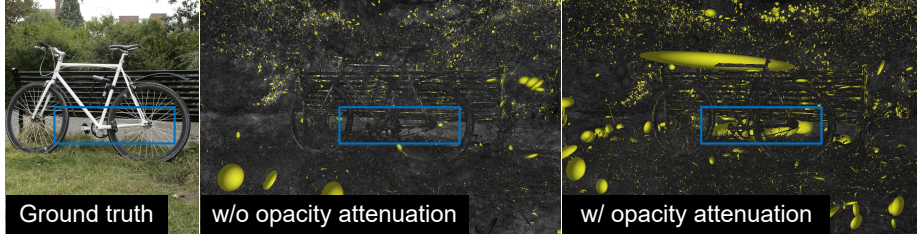


Fig. 5: Comparison of the candidates selected for pruning (yellow primitives) at 14k iterations. Both results share the same primitives, but the opacity of the selected ones is set to 1.0 for visualization. Our opacity attenuation effectively prunes low-frequency primitives overlapped with high-frequency details in heterogeneous regions (e.g., the ground behind the bicycle and grass).

To cope with this, we propose to softly attenuate opacities of the overly-blurred primitives. The standard 3DGS already applies hard attenuation (to near-zero constant) every 3000 iterations [16], but ours is applied at a higher frequency (e.g., every 500 iterations).

For our attenuation to be effectively selective for large-blur objects, the degrees of attenuation scale with the blur levels. In other words, unreliable high-blur primitives are penalized aggressively, and low-blur (higher-quality) primitives are respected relatively. We realize this by formulating a new opacity toward $\tau_m = 0.5$ as:

$$o_{\text{new}} = (1 - (\frac{\sigma_i}{k_\sigma})^\gamma) o_i + (\frac{\sigma_i}{k_\sigma})^\gamma \tau_m, \quad (9)$$

where k_σ is the limit of the allowed blur levels. γ controls the strength of the attenuation, and is empirically set to 0.01.

In order for the attenuated high-blur objects to be qualified for densification, they have to prove their strong contributions through iteration. Otherwise, their opacities are weakened, eventually leading to prune-out. As shown in Fig. 5, our method effectively prunes large-surface primitives (e.g., ground) that overlap with high-frequency details such as grass and bicycle spokes. Additionally, some primitives appear only in the “with attenuation” result, even though both results (w/ and w/o attenuation) share the same primitive set, because the opacities of the selected ones were manually set to 1.0. This indicates that our attenuation also effectively prunes primitives with little visual impact. In this way, we can further avoid redundant densification.

4.5 Blur-driven density control

Now that we have a better densification metric (the per-primitive blur levels σ) and regularization metrics, we improve the 3DGS density control that is based solely on positional gradients $\nabla_p L$ and maximum scale $\|S\|$ of the Gaussian primitive. Algorithm 1 summarizes our densification process with color-coded lines indicating our modifications.

Algorithm 1 Optimization and densification

```

 $\Theta, \bar{\sigma} \leftarrow \text{InitAttributes}(), \Theta = (M, S, R, C, O), \triangleright$  Positions, Scales, Rotations, Colors,
Opacities, Set of  $\sigma$ 
 $i \leftarrow 0 \quad \triangleright$  Iteration Count
while not converged do
   $V, I_{\text{gt}} \leftarrow \text{SampleTrainingView}(), I, \sigma_{\text{pix}} \leftarrow \text{Rasterize}(\Theta, \bar{\sigma}, V)$ 
   $\hat{I}_{\text{gt}} \leftarrow \text{DifferentiableBlur}(I_{\text{gt}}, \sigma_{\text{pix}}) \quad \triangleright$  Sec. 4.2
   $L_{\text{gs}} \leftarrow \text{Loss}_{\text{gs}}(I, I_{\text{gt}}), L_{\sigma} \leftarrow \text{Loss}_{\sigma}(I, \hat{I}_{\text{gt}}, w_d) \quad \triangleright$  Sec. 4.3
   $\Theta \leftarrow \text{Adam}(\nabla L_{\text{gs}}), \bar{\sigma} \leftarrow \text{Adam}(\nabla L_{\sigma}) \quad \triangleright$  Step
  if IsRefinementIteration( $i$ ) then
    for all  $G = (\mu, \Sigma, c, o, \sigma) \in (\Theta, \bar{\sigma})$  do
      if  $\nabla_p L_{\text{gs}} \geq \tau_p$  or ( $\sigma \geq \tau_{\sigma}$  and  $\nabla_p L_{\text{gs}} \geq \tau_p \cdot k_p$ ) then
        if  $\|S\| > \tau_S$  then SplitGaussian( $G$ ) else CloneGaussian( $G$ )
        if  $o < \tau_o$  then PruneGaussian( $G$ )
      if IsAttenuateOpacityIteration( $i$ ) then  $O \leftarrow \text{AttenuateOpacity}(O, \bar{\sigma}) \quad \triangleright$  Sec. 4.4
       $\bar{\sigma} \leftarrow \text{ResetSigma}(\bar{\sigma}) \quad \triangleright$  Avoid bias
   $i \leftarrow i + 1$ 

```

Precisely, we notify that a high σ ($\geq$ a threshold τ_{σ}) of Gaussian primitives requires densification. $\tau_{\sigma}=2$ is empirically set to be balanced in terms of primitive counts and reconstruction quality. In order for our condition to work, the original condition ($\nabla_p L \geq \tau_p$) requires to be relaxed. We reduce τ_p to $\tau_p k_p$ ($k_p < 1$; default: $k_p = 0.05$). This allows many primitives passing the minimal-gradient test to undergo our new test condition.

Our blur-adaptive soft opacity attenuation follows pruning every 500 iterations. Lastly, the blur levels (σ_i) are reset to the initial states to avoid bias in subsequent densification.

5 Experiment

5.1 Experimental setup

Datasets and metrics We follow the 3DGS protocol [16] on Mip-NeRF360 [3], Tanks&Temples [17], and Deep Blending [14], using every 8th image for testing. We report PSNR, SSIM [35], LPIPS [44], and primitive count.

Implementation We build on each 3DGS baseline and keep its training settings and losses. All models are trained for 30k iterations on an NVIDIA A6000 GPU (48GB). The main modification is the integration of the differentiable blur into the 3DGS pipelines. We extend the CUDA kernels by incorporating the forward and backward passes for the per-primitive blur σ , achieving 2.52 ms per step at 1256×828 resolution (see the supplementary for the training-time overhead).

Table 1: Comparison of our method with the previous novel-view synthesis approaches. The results marked with a ‘*’ were taken from the 3DGS paper. The rest of the results were obtained through our own experiments. The **top score**, **second-highest score**, and **third-highest score** are color-coded in red, orange, and yellow, respectively.

Method	Mip-NeRF360					Tanks&Temples					Deep Blending				
	PSNR ↑	SSIM ↑	LPIPS ↓	Prim. ↓	Mem. ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Prim. ↓	Mem. ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Prim. ↓	Mem. ↓
Plenoxels * [10]	23.08	0.626	0.463	-	2.10GB	21.08	0.719	0.379	-	2.30GB	23.06	0.795	0.510	-	2.70GB
iNGP-big * [25]	25.59	0.699	0.331	-	48MB	21.92	0.745	0.305	-	48MB	24.96	0.817	0.390	-	48MB
Mip-NeRF360 * [3]	27.69	0.792	0.237	-	8.6MB	22.22	0.759	0.257	-	8.6MB	29.40	0.901	0.245	-	8.6MB
3DGS [16]	27.414	0.812	0.218	3350K	792MB	23.655	0.844	0.179	1893K	447MB	29.394	0.898	0.248	2833K	670MB
3DGS ($\tau_p = 1.6 \times 10^{-4}$)	27.541	0.818	0.205	4763K	1.12GB	23.772	0.848	0.169	2551K	603MB	29.176	0.896	0.247	3653K	864MB
Mip-Splatting [40]	27.557	0.816	0.217	3271K	786MB	23.963	0.852	0.176	1864K	448MB	29.598	0.904	0.243	2961K	711MB
Pixel-GS [45]	27.537	0.822	0.190	5622K	1.32GB	23.759	0.853	0.151	4598K	1.08GB	28.812	0.891	0.252	4623K	1.09GB
FDS-GS [42]	27.556	0.822	0.208	1542K	370MB	23.990	0.859	0.145	1530K	367MB	29.794	0.905	0.232	1711K	411MB
SteepGS [33]	26.999	0.792	0.249	2182K	516MB	23.393	0.838	0.193	1365K	322MB	29.418	0.903	0.252	1604K	378MB
Compact-GS [20]	26.989	0.796	0.245	1469K	50MB	23.360	0.831	0.199	928K	41MB	29.701	0.900	0.256	1142K	44MB
Ours+3DGS	27.576	0.820	0.190	3083K	729MB	23.821	0.850	0.152	1658K	392MB	29.679	0.900	0.237	1836K	434MB
Ours+Mip-Splatting	27.679	0.824	0.199	2905K	698MB	23.047	0.856	0.162	1629K	391MB	29.802	0.906	0.237	1809K	434MB
Ours+Pixel-GS	27.652	0.822	0.182	4218K	997MB	23.900	0.855	0.138	2999K	709MB	29.620	0.897	0.235	2687K	635MB
Ours+FDS-GS	27.696	0.826	0.203	1501K	360MB	24.230	0.866	0.139	1499K	360MB	29.841	0.907	0.231	1491K	358MB
Ours+SteepGS	27.333	0.811	0.213	2112K	499MB	23.563	0.843	0.173	1206K	285MB	29.592	0.903	0.245	1315K	311MB
Ours+Compact-GS	27.070	0.805	0.225	1530K	51MB	23.560	0.836	0.189	906K	40MB	29.886	0.904	0.249	1028K	42MB

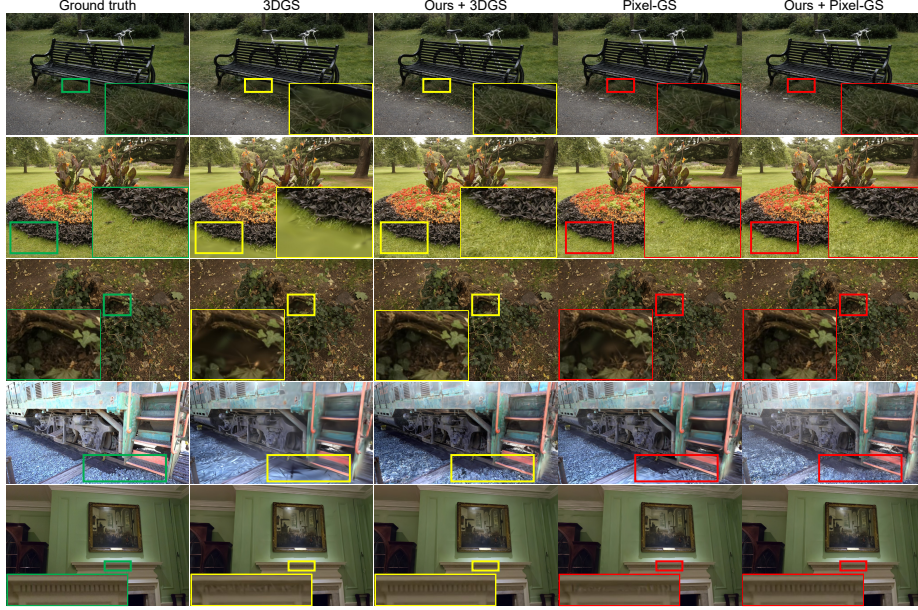


Fig. 6: Qualitative visual comparisons of 3DGS and Pixel-GS with and without our densification. The scenes (from top to bottom) are: Bicycle, Flowers, Stump (Mip-NeRF360), Train (Tanks&Temples), and Dr. Johnson (Deep Blending).

5.2 Comparison with 3DGS methods

We evaluated several state-of-the-art 3DGS methods integrated with our density control: 3DGS [16], Mip-Splatting [40], Pixel-GS [45], FDS-GS [42], SteepGS [33], and Compact-GS [20]. FreGS [43] (no public code) and Wavelet-GS [46] (memory limitations) were excluded. For comparable primitive counts, we adjusted τ_σ and

Table 2: Quantitative comparison with alternative densification methods.

Method / Signal	Mip-NeRF360						Tanks&Temples						Deep Blending					
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	Mem. $\downarrow$	Train $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	Mem. $\downarrow$	Train $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	Mem. $\downarrow$	Train $\downarrow$
Pixel-error-based 3DGS densification methods																		
Revising-3DGS [28]	27.199	0.806	0.201	3079K	728MB	67m	23.640	0.845	0.161	1654K	391MB	44m	29.554	0.901	0.242	1832K	433MB	33m
ConeGS [1]	27.579	0.816	0.184	3083K	729MB	49m	23.924	0.856	0.139	1658K	392MB	30m	30.091	0.902	0.228	1837K	434MB	35m
Ours+3DGS (w/ iNGP init.)	27.797	0.829	0.183	2611K	617MB	38m	23.838	0.851	0.152	1368K	323MB	25m	30.070	0.903	0.231	1893K	448MB	33m
LPIPS-guided densification cue																		
LPIPS-guided	26.971	0.801	0.204	3073K	727MB	107m	23.523	0.841	0.165	1650K	390MB	43m	29.515	0.900	0.246	1825K	432MB	65m
Ours+3DGS	27.576	0.820	0.190	3083K	729MB	39m	23.821	0.850	0.152	1658K	392MB	19m	29.679	0.900	0.237	1836K	434MB	27m

**Fig. 7:** Qualitative comparison with pixel-error-based 3DGS densification methods.

k_p per baseline: $k_p=0.2$ for Mip-Splatting, $(\tau_\sigma, k_p)=(4, 0.4)$ for FDS-GS, $k_p=0.1$ for SteepGS, and $\tau_\sigma=3$ for Compact-GS. Unspecified parameters use the default $(\tau_\sigma, k_p)=(2, 0.05)$. We also tested 3DGS with reduced τ_p and included non-3DGS methods [3, 10, 25].

Quantitative comparison Table 1 reports quantitative results. Our method consistently improves reconstruction quality over the 3DGS baselines with fewer primitives for all datasets. In particular, the integration with Pixel-GS [45] yields the highest quality improvements. Additionally, the great improvements in LPIPS support our differentiable blur as an effective perceptual cue, while the 3DGS with reduced τ_p further validates our method, confirming that simply increasing the primitive count leads to only marginal gains. Per-scene results and timing comparisons are provided in the supplementary.

Qualitative comparison Fig. 6 compares visual quality. Our integrations reduce over- and under-reconstruction artifacts in 3DGS and Pixel-GS, and recover fine details such as grass and fireplace moulding. The gains are most visible in high-frequency outdoor regions. Nevertheless, the indoor scenes having more artificial objects are also improved well. Additional results are provided in the supplementary.

5.3 Comparison with alternative densification cues

Pixel-error cues Pixel-error cues are direct alternatives to our learned blur cue. We thus compare ours with Revising-3DGS [28] and ConeGS [1], representative pixel-error-based densification methods. Since Revising-3DGS has no public code, we reproduce it and verify that its visual tendency is consistent with the paper. ConeGS uses iNGP initialization; we use the same setting to isolate the density-control effect. Table 2 shows that ours consistently outperforms Revising-3DGS. Compared with ConeGS, ours achieves better quality with

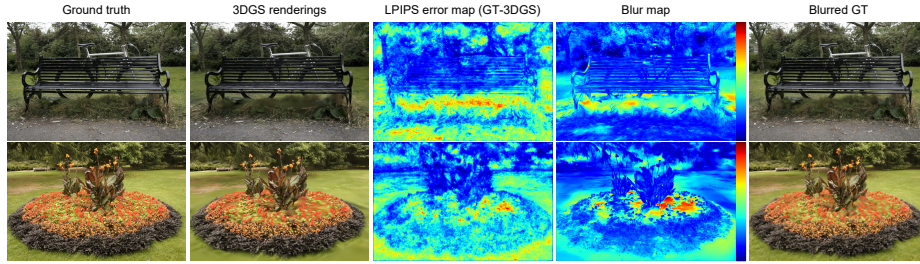


Fig. 8: Visual demonstration of our differentiable blur. From left to right: GT image of a test view, corresponding 3DGS output at 5,000 iterations, LPIPS error map between GT and 3DGS, learned blur map, and blurred GT image using the blur map.

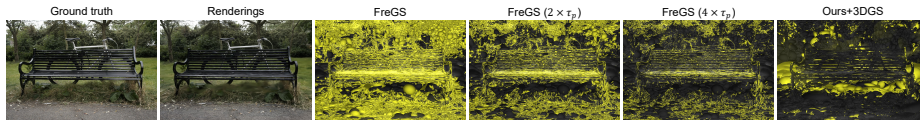


Fig. 9: Comparison of densification candidate selection between a frequency-based densification cue and ours.

fewer primitives on Mip-NeRF360, and comparable quality on Tanks&Temples and Deep Blending. Fig. 7 shows that pixel-error cues tend to smooth fine textures, whereas ours preserves local details, suggesting that our learnable blur better captures insufficient spatial details.

LPIPS-guided cue The learned blur is related to perceptual reconstruction error, but remains more localized. In Fig. 8, high blur values appear on under-detailed structures, while already reconstructed regions receive low values. The blur map follows LPIPS in difficult regions but remains more compact. Table 2 compares ours with an LPIPS-guided cue under a matched primitive budget. As direct LPIPS backpropagation is costly (e.g., over 5 hours on *Bicycle*), the LPIPS-guided baseline follows Revising-3DGS [28] and accumulates LPIPS errors onto primitives. Ours improves quality and training time with similar primitive counts. The blurred GT images also resemble 3DGS outputs, indicating that per-primitive blur reflects reconstruction degradation.

Frequency-based cue We further compare our blur cue with a frequency-based densification cue. Since FreGS [43] code and exact thresholds are unavailable, we analyze the qualitative behavior of densification candidates. Fig. 9 shows that, as the threshold increases ($2\tau_p, 4\tau_p$), candidates around the blurred grass under the bench are discarded earlier than those in less problematic regions. In contrast, our learned blur remains concentrated on the under-resolved grass region, indicating that it better captures missing details. This distinction also relates to efficiency: FreGS [43] reports primitive count only on *Stump*, where it

Table 3: Ablation study evaluating the effects of differentiable blur (Diff. blur; Sec. 4.2), detail-aware regularization (L_{reg} ; Sec. 4.3), and opacity attenuation (opacity att.; Sec. 4.4) on three real-world datasets. The base method is 3DGS [16].

	Mip-NeRF360					Tanks&Temples					Deep Blending				
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	Mem. $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	Mem. $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	Mem. $\downarrow$
3DGS	27.414	0.812	0.218	3350K	792MB	23.655	0.844	0.179	1893K	447MB	29.394	0.898	0.248	2833K	670MB
+ Diff. blur	27.594	0.818	0.185	4213K	996MB	23.685	0.846	0.143	2586K	611MB	29.626	0.897	0.231	2565K	606MB
+ Diff. blur + L_{reg}	27.570	0.819	0.189	3621K	856MB	23.709	0.848	0.152	1977K	467MB	29.617	0.899	0.236	2140K	506MB
+ Diff. blur + Opacity att.	27.605	0.820	0.185	3520K	832MB	23.839	0.851	0.141	2061K	487MB	29.693	0.898	0.232	2197K	519MB
FULL	27.576	0.820	0.190	3083K	729MB	23.821	0.850	0.152	1658K	392MB	29.679	0.900	0.237	1836K	434MB



Fig. 10: Qualitative results for each component of our method on Garden scene from the Mip-NeRF360 dataset.

uses 8.12M primitives compared with 4.90M for 3DGS, while Wavelet-GS [46] exceeded GPU memory in our experiments.

5.4 Ablation study

We evaluate our design choices from two aspects: the contribution of each component and the behavior of the blur-related parameters under different settings.

Component-wise analysis To assess the effectiveness of each component, we conduct an ablation study using 3DGS as the baseline, as summarized in Table 3. The differentiable blur (Sec. 4.2) improves all metrics but increases primitive count, motivating regularization of Gaussian growth.

The detail-aware regularization L_{reg} (Sec. 4.3) and soft opacity attenuation (Sec. 4.4) reduce primitives by up to 23.6% and 20.3%, respectively, over differentiable blur alone while preserving comparable quality.

With all components, ours reduces primitives by up to 35.2% (Deep Blending) while improving quality. Fig. 10 shows that the full model recovers subtle leaf details with the fewest primitives, underscoring the complementary effects of each component. Additional hyperparameter ablations are provided in the supplementary.

Kernel radius for detail-aware weight map We verify whether the detail-aware weight map w_d is sensitive to the Gaussian kernel radius used for its construction. The default setting uses $r=60$ to compare the GT image with its heavily blurred counterpart (Sec. 4.3). Table 4 shows stable results for $r \in \{30, 60, 90\}$ on all datasets, suggesting that w_d acts as a coarse prior for suppressing low-detail regions rather than depending on a precise radius. We therefore use $r=60$ by default as a balanced trade-off between quality and compactness.

Table 4: Ablation on the Gaussian kernel radius r for the detail-aware weight map w_d . The default is $r=60$.

r	Mip-NeRF360				Tanks&Temples				Deep Blending			
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$
30	27.585	0.820	0.190	3076K	23.812	0.850	0.155	1585K	29.717	0.900	0.237	1846K
60 (default)	27.576	0.820	0.190	3083K	23.821	0.850	0.152	1658K	29.679	0.900	0.237	1836K
90	27.564	0.820	0.189	3116K	23.873	0.850	0.152	1663K	29.637	0.900	0.237	1842K

Table 5: Ablation on the resolution scaling of per-primitive blur σ_i . “Norm.” denotes $\tilde{\sigma}_i=\sigma_i/s$, where s is the resolution scale factor.

Method	Mip-NeRF360				Tanks&Temples				Deep Blending			
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Prim. $\downarrow$
3DGS ($\times 0.5$)	28.352	0.859	0.136	2909K	24.805	0.892	0.103	1196K	29.488	0.910	0.149	2515K
Ours+3DGS ($\times 0.5$)	28.514	0.863	0.123	1855K	24.888	0.893	0.095	798K	30.029	0.915	0.139	1458K
3DGS ($\times 1.25$)	27.452	0.810	0.235	3176K	24.027	0.851	0.187	1712K	29.516	0.907	0.244	2667K
Ours+3DGS ($\times 1.25$)	27.625	0.818	0.200	3738K	24.002	0.856	0.161	1897K	29.683	0.908	0.234	1863K
Ours+3DGS (Norm., $\times 1.25$)	27.593	0.819	0.207	2759K	24.007	0.854	0.170	1428K	29.759	0.910	0.237	1602K

Resolution dependency of σ_i The learned per-primitive blur σ_i can depend on rendering resolution, as it is optimized through the pixel-space blur map σ_{pix} . We therefore test ours at $\times 0.5$ and $\times 1.25$ rendering resolutions (see Table 5). At $\times 0.5$, ours still improves quality with fewer primitives. At $\times 1.25$, however, our blur criterion allocates more primitives, since the same image structure spans more pixels. We mitigate this issue with the normalized per-primitive blur $\tilde{\sigma}_i=\sigma_i/s$, where s is the resolution factor, while maintaining better quality with fewer primitives. The result indicates that our blur criterion can be adapted to non-default resolutions.

6 Conclusion

We introduced SharpGS, a differentiable blur-driven density control for 3DGS that preserves high-frequency details. Our density control leverages per-primitive blur levels as a perceptual cue, effectively densifies primitives to match varying spatial complexities, and facilitates CUDA for efficiency. We demonstrated that our solution greatly improves the quality of the novel view synthesis, still keeping primitive counts compact. Ours can be virtually integrated with most of the existing 3DGS methods. When integrated, our models significantly outperform the original models in terms of both quality and primitive count.

Acknowledgements

This work was supported in part by the Mid-career Research Program and CRC Program through the NRF Grants (Nos. RS-2024-00339681 and RS-2023-00221186), and IITP grants (No. RS-2024-00454666), funded by the Korea government (MSIT). Correspondence concerning this article can be addressed to Sungkil Lee.

References

1. Baranowski, B., Esposito, S., Gschöfmann, P., Chen, A., Geiger, A.: Conegs: Error-guided densification using pixel cones for improved reconstruction with fewer primitives. In: International Conference on 3D Vision (3DV). pp. 989–999. IEEE (2026)
2. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: ICCV. pp. 5855–5864 (2021)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: CVPR. pp. 5470–5479 (2022)
4. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. In: ICCV. pp. 19697–19705 (2023)
5. Blanc, H., Deschaud, J.E., Paljic, A.: Raygauss: Volumetric gaussian-based ray casting for photorealistic novel view synthesis. In: WACV. pp. 1808–1817. IEEE (2025)
6. Cai, J., Lu, H.: Nerf-based multi-view synthesis techniques: A survey. In: Int. Wireless Communications and Mobile Computing. pp. 208–213 (2024)
7. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: ECCV. pp. 333–350 (2022)
8. Chen, Z., Funkhouser, T., Hedman, P., Tagliasacchi, A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In: CVPR. pp. 16569–16578 (2023)
9. Fei, B., Xu, J., Zhang, R., Zhou, Q., Yang, W., He, Y.: 3d gaussian splatting as a new era: A survey. IEEE TVCG **31**(8), 4429–4449 (2025)
10. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR. pp. 5501–5510 (2022)
11. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. In: ICCV. pp. 14346–14355 (2021)
12. Guo, Y.C., Kang, D., Bao, L., He, Y., Zhang, S.H.: Nerfren: Neural radiance fields with reflections. In: CVPR. pp. 18409–18418 (2022)
13. Hamdi, A., Melas-Kyriazi, L., Mai, J., Qian, G., Liu, R., Vondrick, C., Ghanem, B., Vedaldi, A.: Ges: Generalized exponential splatting for efficient radiance field rendering. In: CVPR. pp. 19812–19822 (2024)
14. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM TOG **37**(6), 1–15 (2018)
15. Hu, T., Liu, S., Chen, Y., Shen, T., Jia, J.: Efficientnerf efficient neural radiance fields. In: CVPR. pp. 12902–12911 (2022)
16. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM TOG **42**(4), 1–14 (2023)
17. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM TOG **36**(4), 1–13 (2017)
18. Kopanas, G., Philip, J., Leimkühler, T., Drettakis, G.: Point-based neural rendering with per-view optimization. Comput. Graph. Forum **40**(4), 29–43 (2021)
19. Lee, B., Lee, H., Ali, U., Park, E.: Sharp-nerf: Grid-based fast deblurring neural radiance fields using sharpness prior. In: WACV. pp. 3709–3718 (2024)
20. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. In: CVPR. pp. 21719–21728 (2024)
21. Li, J., Shi, Y., Cao, J., Ni, B., Zhang, W., Zhang, K., Van Gool, L.: Mipmap-gs: Let gaussians deform with scale-specific mipmap for anti-aliasing rendering. In: International Conference on 3D Vision (3DV). pp. 935–945. IEEE (2025)

22. Liang, Z., Zhang, Q., Hu, W., Feng, Y., Zhu, L., Jia, K.: Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. In: ECCV. pp. 281–297 (2024)
23. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: CVPR. pp. 20654–20664 (2024)
24. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV. pp. 405–421 (2020)
25. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM TOG **41**(4), 1–15 (2022)
26. Peng, C., Tang, Y., Zhou, Y., Wang, N., Liu, X., Li, D., Chellappa, R.: Bags: Blur agnostic gaussian splatting through multi-scale kernel modeling. In: ECCV. pp. 293–310 (2024)
27. Rivas-Manzanique, F., Sierra-Acosta, J., Penate-Sanchez, A., Moreno-Noguer, F., Ribeiro, A.: Nerflight: Fast and light neural radiance fields using a shared feature grid. In: CVPR. pp. 12417–12427 (2023)
28. Rota Bulò, S., Porzi, L., Kotschieder, P.: Revising densification in gaussian splatting. In: ECCV. pp. 347–362. Springer (2024)
29. Song, X., Zheng, J., Yuan, S., Gao, H.a., Zhao, J., He, X., Gu, W., Zhao, H.: Sags: Scale-adaptive gaussian splatting for training-free anti-aliasing. arXiv preprint arXiv:2403.19615 (2024)
30. Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Yifan, W., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., et al.: Advances in neural rendering. Comput. Graph. Forum **41**(2), 703–735 (2022)
31. Turki, H., Agrawal, V., Bulò, S.R., Porzi, L., Kotschieder, P., Ramanan, D., Zöllhöfer, M., Richardt, C.: Hybridnerf: Efficient neural rendering via adaptive volumetric surfaces. In: CVPR. pp. 19647–19656 (2024)
32. Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-nerf: Structured view-dependent appearance for neural radiance fields. In: CVPR. pp. 5481–5490 (2022)
33. Wang, P., Wang, Y., Wang, D., Mohan, S., Fan, Z., Wu, L., Cai, R., Yeh, Y.Y., Wang, Z., Liu, Q., et al.: Steepest descent density control for compact 3d gaussian splatting. In: CVPR. pp. 26663–26672 (2025)
34. Wang, Y., Wang, C., Gong, B., Xue, T.: Bilateral guided radiance field processing. ACM TOG **43**(4), 1–13 (2024)
35. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE TIP **13**(4), 600–612 (2004)
36. Xie, S., Wang, Z., Wang, X., Pan, C., Wang, Q., Dong, X.: Super-resolution 3d gaussian splatting via latent feature field and gradient-guided splitting. In: 2025 IEEE International Conference on Unmanned Systems (ICUS). pp. 2019–2025. IEEE (2025)
37. Ye, Z., Li, W., Liu, S., Qiao, P., Dou, Y.: Absgs: Recovering fine details in 3d gaussian splatting. In: ACM MM. pp. 1053–1061 (2024)
38. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenotrees for real-time rendering of neural radiance fields. In: ICCV. pp. 5752–5761 (2021)
39. Yu, X., Zhu, H., He, T., Chen, Z.: Gaussiansr: 3d gaussian super-resolution with 2d diffusion priors. arXiv preprint arXiv:2406.10111 (2024)
40. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: Mip-splatting: Alias-free 3d gaussian splatting. In: CVPR. pp. 19447–19456 (2024)
41. Yu, Z., Sattler, T., Geiger, A.: Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. ACM TOG **43**(6), 1–13 (2024)

42. Zeng, Z., Wang, Y., Ju, L., Guan, T.: Frequency-aware density control via reparameterization for high-quality rendering of 3d gaussian splatting. In: AAAI. pp. 9833–9841 (2025)
43. Zhang, J., Zhan, F., Xu, M., Lu, S., Xing, E.: Fregs: 3d gaussian splatting with progressive frequency regularization. In: CVPR. pp. 21424–21433 (2024)
44. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
45. Zhang, Z., Hu, W., Lao, Y., He, T., Zhao, H.: Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting. In: ECCV. pp. 326–342. Springer (2024)
46. Zhao, B., Zhou, Y., Yu, S., Wang, Z., Wang, H.: Wavelet-gs: 3d gaussian splatting with wavelet decomposition. In: ACM MM. pp. 8616–8625 (2025)
47. Zhu, Z., Fan, Z., Jiang, Y., Wang, Z.: Fsgs: Real-time few-shot view synthesis using gaussian splatting. In: ECCV. pp. 145–163. Springer (2024)
48. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Ewa volume splatting. In: Proceedings Visualization, 2001. VIS'01. pp. 29–538. IEEE (2001)