Gaussian Blending: Rethinking Alpha Blending in 3D Gaussian Splatting

Junseo Koo, Jinseo Jeong, Gunhee Kim

Seoul National University
junseo.koo@vision.snu.ac.kr, jinseo.jeong@vision.snu.ac.kr, gunhee@snu.ac.kr

Abstract

The recent introduction of 3D Gaussian Splatting (3DGS) has significantly advanced novel view synthesis. Several studies have further improved the rendering quality of 3DGS, yet they still exhibit noticeable visual discrepancies when synthesizing views at sampling rates unseen during training. Specifically, they suffer from (i) erosion-induced blurring artifacts when zooming in and (ii) dilation-induced staircase artifacts when zooming out. We speculate that these artifacts arise from the fundamental limitation of the alpha blending adopted in 3DGS methods. Instead of the conventional alpha blending that computes alpha and transmittance as scalar quantities over a pixel, we propose to replace it with our novel *Gaussian Blending* that treats alpha and transmittance as spatially varying distributions. Thus, transmittances can be updated considering the spatial distribution of alpha values across the pixel area, allowing nearby background splats to contribute to the final rendering. Our Gaussian Blending maintains real-time rendering speed and requires no additional memory cost, while being easily integrated as a drop-in replacement into existing 3DGS-based or other NVS frameworks. Extensive experiments demonstrate that Gaussian Blending effectively captures fine details at various sampling rates unseen during training, consistently outperforming existing novel view synthesis models across both unseen and seen sampling rates.

Project Page —
https://1207koo.github.io/html/gaussianblending/

1 Introduction

Novel view synthesis (NVS) has advanced rapidly, playing a pivotal role across diverse content generation tasks. A key milestone was Neural Radiance Field (NeRF) (Mildenhall et al. 2021), an implicit neural representation employing a neural network to estimate volumetric density and view-dependent radiance of 3D points. However, NeRF suffers from slow rendering speeds due to intensive ray marching. Recently, 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) explicitly represents 3D scenes using Gaussian splats, enabling faster and finer view synthesis results.

In the 3DGS framework, Gaussian splats are projected onto the 2D image plane, followed by the alpha blending to render the final pixel color. However, 3DGS and its variants still suffer from noticeable artifacts when synthesizing views at unseen sampling rates (e.g., zoom-in or zoom-out). This persistent challenge arises since existing methods treat pixels as points rather than planes, which leads to aliasing effects such as boundary erosion and dilation (see Figure 1).

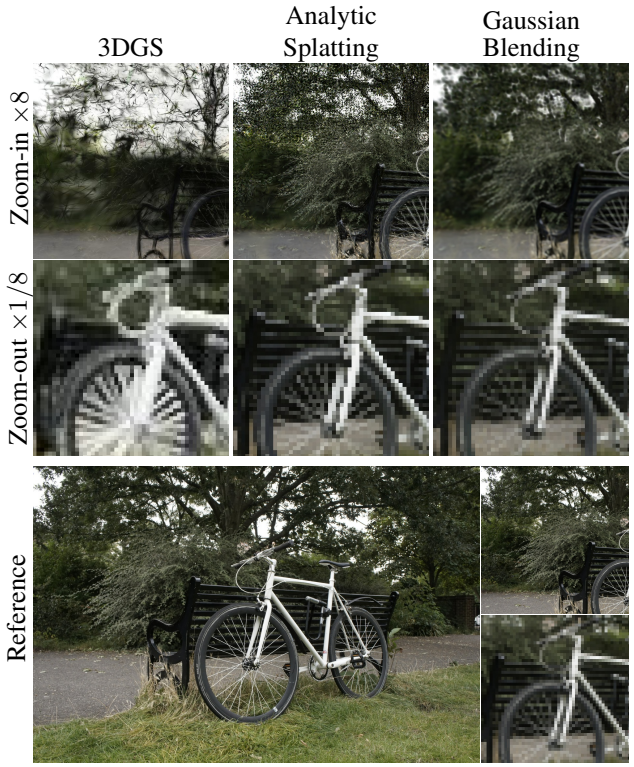

Figure 1: Scalar alpha blending used in previous novel view synthesis methods (e.g., 3DGS and Analytic Splatting) exhibits aliasing artifacts when rendering at different sampling rates. Specifically, erosion with noisy edges occurs during zoom-in (top), while dilation with staircase artifacts occurs during zoom-out (bottom). In contrast, our Gaussian Blending produces consistent synthesis results at unseen sampling rates (e.g., leaves and bicycle frame) without any priors and additional training.

In real-world camera systems, a sensor integrates radiances within a pixel area, whereas existing NVS methods use single ray for rendering a pixel color. According to the Nyquist–Shannon sampling theorem (Shannon 1949), suppressing high-frequency components at lower sampling rates can alleviate this issue, as supported by several prefiltering methods (Barron et al. 2021, 2022; Yu et al. 2024b). More recently, Analytic-Splatting (Liang et al. 2024b) explicitly integrates Gaussian splat responses over pixel regions, better emulating physical camera sensors.

Despite recent advances, existing NVS approaches still exhibit noticeable aliasing, especially at object boundaries. As illustrated in Figure 1, rendering at higher sampling rates (zoom-in) produces boundary erosion with blurry edges, while lower sampling rates (zoom-out) produces boundary dilation as staircase artifacts. Even Analytic-Splatting suffers from these issues due to reliance on scalar alpha blending. All NVS methods, including various anti-aliasing methods, use scalar alpha values computed along single rays. This inherently leads foreground splats (or density points in NeRF-based methods) to fully occlude background splats, disregarding their spatial overlap in the 2D screen space. Consequently, the spatial contribution of background splats is inaccurately computed, causing persistent aliasing. To mitigate this issue, existing NVS methods typically require retraining models at specific sampling rates, which is inefficient and impractical when training data are unavailable.

To address this limitation, we introduce *Gaussian Blending* as a novel rendering methodology. Rather than modeling alpha and transmittance as single scalar values, our approach models them as spatially varying functions across the 2D pixel area. Our key insight is that Gaussian splats, being smooth and continuous, naturally cluster to represent distinct scene structures. Leveraging this spatial coherence, we approximate their combined transmittance using simplified 2D uniform distributions within each pixel region. This approximation enables efficient, dynamic updating of spatial transmittance distributions during the alpha blending process via distributional moments.

In experiments on the multi-scale Blender (Mildenhall et al. 2021; Barron et al. 2021) and Mip-NeRF 360 datasets (Barron et al. 2022), Gaussian Blending significantly reduces intra-pixel aliasing, mitigating erosion and dilation at previously unseen sampling rates. Moreover, Gaussian Blending achieves real-time rendering speeds comparable to the original 3DGS method via optimized CUDA implementation, improving visual fidelity without extra memory overhead. In addition, it can be seamlessly integrated as a drop-in replacement for the rendering kernels of existing NVS frameworks.

Finally, our key contributions are as follows.

- We revisit existing scalar alpha blending methods and identify that aliasing arises from inadequate handling of the spatial variations within pixel regions.
- To the best of our knowledge, our work is the first to integrate intra-pixel anti-aliasing into the alpha blending process explicitly. Our proposed *Gaussian Blending* efficiently models and dynamically tracks spatial variations within pixels to effectively suppress aliasing.
- Extensive experiments demonstrate that Gaussian Blending significantly reduces aliasing, yielding higher-quality synthesized views at both unseen and seen sampling rates during training, without additional priors or retraining, all while preserving real-time performance. Our Gaussian Blending can be easily integrated as a drop-in replacement into existing NVS frameworks.

2 Related Work

2.1 Novel View Synthesis (NVS)

NVS aims to reconstruct the geometry and appearance of 3D scenes from multi-view images, enabling rendering from unseen viewpoints (Pumarola et al. 2021; Park et al. 2021; Yang et al. 2024; Jeong et al. 2024; Liang et al. 2024a; Moenne-Loccoz et al. 2024). While Neural Radiance Fields (NeRF) (Mildenhall et al. 2021) have laid the groundwork, they suffer from slow rendering speeds due to intensive ray marching. To overcome these limitations, subsequent methods adopt explicit grid-based scene representations (Sun, Sun, and Chen 2022; Chen et al. 2022), significantly enhancing rendering efficiency through strategies like free-space skipping. Hybrid methods further combine NeRF with Signed Distance Functions (SDF) (Wang et al. 2021; Wu et al. 2023), extending implicit representations to support accurate surface reconstruction.

3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) represents scenes as collections of Gaussian splats, enabling real-time, high-quality rendering. Unlike implicit neural methods involving computationally heavy ray marching, 3DGS rasterizes Gaussian splats directly into screen space, allowing efficient rendering via GPU-optimized rasterization with explicit alpha blending. Building upon this framework, several variants have been proposed to further enhance rendering quality and efficiency. 2D Gaussian Splatting (2DGS) (Huang et al. 2024) and 3D Half-Gaussian Splatting (3D-HGS) (Li et al. 2025) respectively introduce 2D Gaussian splats and half-Gaussian kernels, to better capture geometric details. Scaffold-GS and Octree-GS (Lu et al. 2024; Ren et al. 2024) employ anchor-based or Level-of-Detail-structured 3D Gaussians to reduce redundancy and enhance efficiency in large-scale view-varying scenarios. These advances have broadened the applicability of NVS, including dynamic scene reconstruction (Pumarola et al. 2021; Park et al. 2021; Yang et al. 2024), generative modeling (Poole et al. 2023; Haque et al. 2023; Lin et al. 2025), and inverse rendering tasks (Jeong et al. 2024; Liang et al. 2024a; Moenne-Loccoz et al. 2024), inspiring numerous downstream developments.

2.2 NVS at Different Sampling Rates

Despite rapid progress in NVS, most existing methods assume fixed resolutions, camera distances, and camera intrinsics. However, real-world applications often involve varying sampling rates, causing noticeable aliasing artifacts due to single ray rendering that neglects the visible frustum.

Several methods have addressed this issue through prefiltering-based anti-aliasing approaches following the

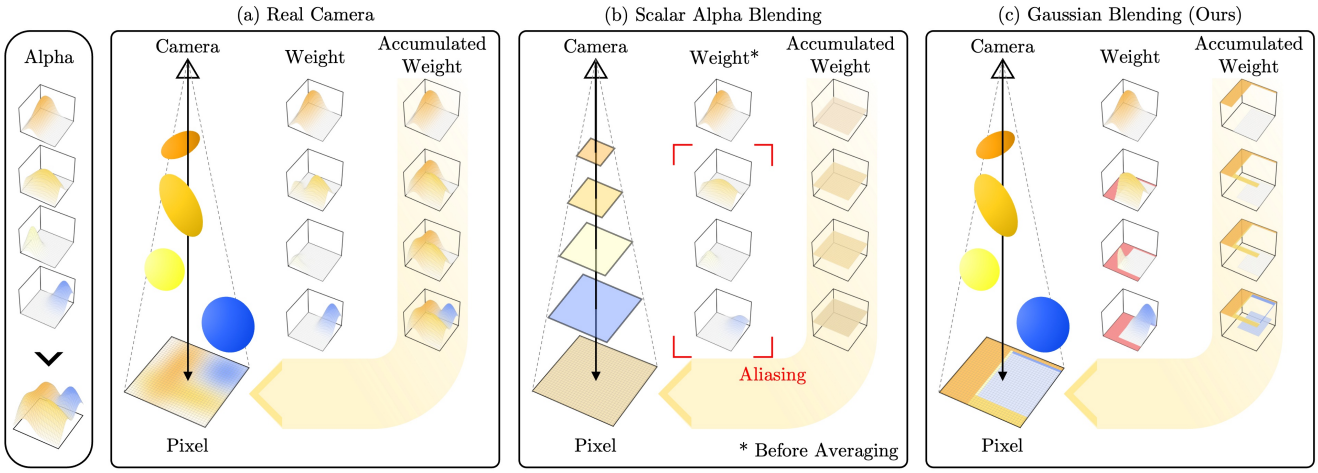

Figure 2: Comparison of blending behaviors when rendering overlapping splats. (a) In a real camera system, spatial occlusion is properly handled—the yellow splat is attenuated by closer splats, while the blue splat, which does not overlap in screen space, retains high transmittance and remains visible. (b) Scalar alpha blending ignores spatial occlusion by averaging alpha values across the pixel, causing dilation of the overlapping yellow splat and undesired suppression of the non-overlapping blue splat. (c) Our Gaussian Blending dynamically adjusts the transmittance window within each pixel, preserving high transmittance in regions without splats and effectively maintaining the visibility of the blue splat while reducing the influence of the occluded yellow splat.

Nyquist–Shannon Sampling Theorem (Shannon 1949). For example, Mip-NeRF (Barron et al. 2021) reduces aliasing by integrating positional encoding over spatial regions, effectively applying implicit low-pass filtering. Mip-NeRF 360 (Barron et al. 2022) further extends this approach to handle unbounded scenes, addressing aliasing more effectively in realistic scenarios. With the emergence of 3DGS, Mip-Splatting (Yu et al. 2024b) combines 2D and 3D filtering strategies, mitigating dilation artifacts caused by varying sampling rates. Going beyond these filtering-based approaches, Analytic-Splatting (Liang et al. 2024b) analytically computes Gaussian splat responses over pixel areas, explicitly integrating splat contributions to achieve effective and accurate anti-aliasing.

Mipmap-based approach is another popular strategy to handle anti-aliasing. They often train separate scene representations for each sampling rate. Instant-NGP (Müller et al. 2022) pioneers resolution-adaptive rendering through multi-resolution hash-grid features. Tri-MipRF (Hu et al. 2023) introduces learnable mipmap representations, dynamically retrieving resolution-specific features during rendering. Recent adaptations (Li et al. 2024; Yan et al. 2024) group Gaussian splats based on sampling rates. However, these mipmap-based approaches can only handle specific resolutions seen during training, limiting their generalization.

Additionally, several studies focus on enhancing high-frequency details or resolving artifacts when rendering at higher sampling rates beyond training conditions. While straightforward techniques such as supersampling show limited effectiveness (Wang et al. 2022), recent approaches leverage generative diffusion models (Yu et al. 2024a; Xia and Liu 2025) or single-image super-resolution (SISR) frameworks (Feng et al. 2024) to generate pseudo-ground-truth details of test-time resolutions. Unlike these methods that improve image quality by generating high-frequency details, our approach directly addresses aliasing artifacts caused by scalar alpha blending.

3 Approach

3.1 Preliminaries

3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) explicitly represents a 3D scene using a set of Gaussian splats. Given a total of N splats, the i-th closest splat to the camera is parameterized by its center position $\mu_i \in \mathbb{R}^3$, scale vector $s_i \in \mathbb{R}^3$, rotation quaternion $r_i \in \mathbb{R}^4$, and opacity value $o_i \in \mathbb{R}$. The scale vector s_i forms a diagonal scale matrix $S_i \in \mathbb{R}^{3\times3}$, and the quaternion r_i defines a rotation matrix $R_i \in \mathbb{R}^{3\times3}$. Consequently, the 3D covariance matrix of the Gaussian splat is computed as

$$\Sigma_i = R_i S_i S_i^\top R_i^\top. \tag{1}$$

The influence of the i-th Gaussian splat at a 3D position x is defined as

$$G_i(x) = o_i \exp\left(-\frac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\right). \tag{2}$$

Each 3D Gaussian splat is projected onto the 2D screen space, resulting in the corresponding 2D Gaussian splat. Given a camera extrinsic matrix W and an intrinsic projection matrix K, the projected mean vector and covariance matrix are

$$\mu_i' = KW[\mu_i, 1]^\top, \quad \Sigma_i' = J_i W \Sigma_i W^\top J_i^\top, \tag{3}$$

where J_i is the Jacobian of the local affine approximation of the perspective projection at μ_i. By removing its third row

and column, Σ_i' becomes a 2D covariance matrix in $\mathbb{R}^{2\times2}$. Thus, the screen space influence of the projected Gaussian splat at a 2D position x is

$$G_i'(x) = o_i \exp\left(-\frac{1}{2}(x-\mu_i')^\top \Sigma_i'^{-1}(x-\mu_i')\right). \quad (4)$$

Given a pixel center p, each splat's alpha value is computed as the Gaussian influence evaluated at the pixel center $G_i'(p)$. Rendering is performed by blending these splats in the front-to-back depth order using alpha blending. As c_i is the view-dependent color of the i-th splat, the final pixel color C_p is computed as

$$C_p = \sum_{i=1}^{N} c_i G_i'(p) T_i, \quad T_i = \prod_{j=1}^{i-1}(1 - G_j'(p)), \quad (5)$$

where the contribution of each splat can be interpreted as the weight $w_i = G_i'(p)T_i$.

Analytic-Splatting (Liang et al. 2024b) extends the 3DGS approach by incorporating anti-aliasing through analytic integration of each Gaussian splat's influence over the entire pixel area, rather than evaluating it only at the pixel center. This integral of 2D Gaussian cannot be simplified with a closed form; thus, Analytic-Splatting employs an efficient approximation, which performs an eigen-decomposition on each 2D Gaussian splat to identify principal axes. Then, the pixel coordinate frame is rotated to align with these axes, eliminating the correlation term in the Gaussian function. Consequently, the integral can be factorized into two separable 1D Gaussian integrals along these axes, each computed analytically. Further details can be found in the original papers (Kerbl et al. 2023; Liang et al. 2024b).

3.2 Scalar Alpha Blending

Although Analytic-Splatting theoretically eliminates aliasing artifacts, as illustrated in Figure 1, it still exhibits persistent aliasing issues like erosion and dilation. We identify that these problems fundamentally originate from scalar alpha blending, and subsequently describe how our proposed approach overcomes this limitation.

Alpha blending is a fundamental rendering technique used by most NVS methods, including anti-aliasing approaches. However, conventional alpha blending represents alpha and transmittance as scalar values, ignoring critical spatial variations within pixels.

Physically, rendering should involve integrating spatially varying alpha and color distributions over each pixel region. The physically correct pixel color C_p^p is computed by integrating the contributions of all splats and resulting transmittance T_i^p within the pixel area p:

$$C_p^p = \int_p \sum_{i=1}^{N} T_i^p(x)\alpha_i(x)c_i dx,$$
$$T_i^p(x) = \prod_{j=1}^{i-1}(1-\alpha_j(x)). \quad (6)$$

Existing methods approximate this integration by computing scalar alpha integrals independently for each splat:

$$C_p^s = \sum_{i=1}^{N} T_i^s c_i \int_p \alpha_i(x)dx,$$
$$T_i^s = \prod_{j=1}^{i-1}\left(1 - \int_p \alpha_j(x)dx\right), \quad (7)$$

where C_p^s is the approximated pixel color, and T_i^s is the scalar transmittance after rendering the i-th splat.

Figure 2 illustrates that such scalar approximations inevitably lead to aliasing artifacts. In a real camera system, overlapping splats that lie behind other splats should attenuate their alpha values due to spatial occlusion. However, the scalar approximation in alpha and transmittance ignores the spatial context such as occlusions, computing alpha values regardless of the spatial occlusion. This results in the object edges, which partially overlap within the pixel area, being rendered with higher alpha values than they should be, even though object splats are highly occluding each other. This rapidly saturates transmittance, leading to pixel-level dilation artifacts, where foreground objects appear artificially enlarged, and background objects (e.g., the blue splat in Figure 2) become overly occluded, as shown in Figure 1. At lower sampling rates (zoom-out), saturated transmittance causes dilation artifacts, despite the broader visible frustum per pixel. Conversely, at higher sampling rates (zoom-in), scalar alpha blending induces erosion artifacts with blurry boundaries due to dilation bias learned during training.

This limitation persists across all existing NVS approaches, including standard rendering, and prefiltering or mipmap-based anti-aliasing, since all rely on scalar alpha blending. Only supersampling approaches, which explicitly compute spatial variations in alpha and transmittance, accurately capture these effects. However, supersampling is computationally expensive, requiring multiple rendering passes, and is thus challenging to employ in real-time applications.

3.3 Efficient Spatial Alpha Blending

A straightforward and physically accurate approach to spatial variations in transmittance is to integrate the transmittance T_i and the corresponding weight w_i across each pixel's 2D screen space coverage. However, computing these integrals requires exponential computational complexity, as the transmittance $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$ and the weight $w_i = \alpha_i T_i$ consist of $2^{i-1}-1$ and 2^{i-1} terms, respectively.

Our key insight to mitigate this computational challenge is that Gaussian splats collectively represent continuous object surfaces, typically forming uniform alpha distributions across 2D screen space, such as $\alpha = 1$ for opaque surfaces. Thus, despite the transmittance involving an exponentially large number of Gaussian terms, these terms spatially cluster into compact regions, effectively approximated by a uniform distribution within each pixel area. Leveraging this observation, we approximate the transmittance distribution using a simpler, spatially uniform 2D representation.

By adaptively controlling the spatial extent of this uniform distribution, we dynamically modulate the effective

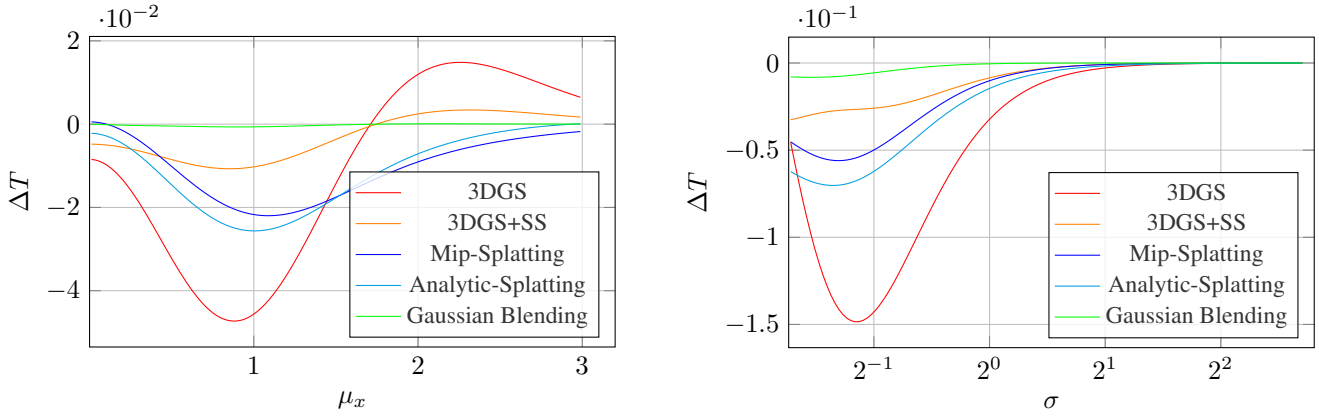

Figure 3: Comparison of the remaining transmittance error ΔT after rendering two splats symmetrically placed at $\mu_1 = [\mu_x, -0.1]^\top$ and $\mu_2 = [\mu_x, 0.1]^\top$, with $o_1 = o_2 = 1$ and $\sigma_1 = \sigma_2 = \sigma$. (Left) Varying μ_x while fixing $\sigma = 1$. (Right) Varying σ while fixing $\mu_x = 0.5$. A negative ΔT indicates dilation, where a lower transmittance remains after rendering.

transmittance, reducing the influence of splats located in regions already occluded by previously rendered surfaces, where transmittance values are low. This approach enables real-time rendering without compromising visual accuracy.

Moreover, representing transmittance as a 2D uniform distribution offers two physical advantages. First, it inherently satisfies $0 \leq w_i \leq T_i$ throughout the pixel, ensuring physically consistent blending. Second, the initial pixel state naturally conforms to a uniform distribution, simplifying the computation of subsequent rendering windows.

Formally, we represent transmittance T_i by a window centered at $x_i \in \mathbb{R}^2$ with size $l_i = [l_{i,1}, l_{i,2}]^\top \in \mathbb{R}^2$ and uniform transmittance value $t_i \in \mathbb{R}$. Initially, before any splats are rendered, T_1 is defined as $x_1 = p$, $l_1 = [1, 1]^\top$, and $t_1 = 1$. Assuming splats composing an object form uniform distributions in the pixel region, dynamically adjusting the window (x_i, l_i) enables accurate representation of lower-transmittance subregions.

When rendering splat α_i onto distribution T_i, we compute two quantities: (i) the integrated splat response within the current window region, $\int w_i(x)dx$, and (ii) the optimal distribution T_{i+1} approximating the remaining transmittance $T_i(x)(1 - \alpha_i(x))$.

Weight Computation. To compute a splat's response $\int w_i(x)dx$, we first perform eigen-decomposition on the 2D covariance matrix Σ_i', yielding eigenvalues λ_1, λ_2 and unit eigenvectors e_1, e_2, defining principal axes. The standard deviations along these axes are $\sigma_1 = \sqrt{\lambda_1}$ and $\sigma_2 = \sqrt{\lambda_2}$. Next, we rotate the current window T_i slightly (within 45°) to align it with the Gaussian axes defined by e_1, e_2 to efficiently approximate the integral. Without loss of generality, let us assume λ_1, e_1 align with the first axis, and λ_2, e_2 with the second axis. In the coordinate system defined by the splat α_i, centered at μ_i', the window center is located at $[u, v]^\top = [(x_i - \mu_i') \cdot e_1, (x_i - \mu_i') \cdot e_2]^\top$, and the window region spans $[u_1, u_2] \times [v_1, v_2]$ with boundaries:

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} - \frac{1}{2} l_i, \quad \begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} + \frac{1}{2} l_i. \tag{8}$$

To simplify the integrals, we introduce the notation for the k-th order moments of the 1D Gaussian as $I_\sigma^k(a, b) = \int_a^b x^k \exp(-x^2/2\sigma^2)dx$. Then, we express the integrated weight as

$$\int w_i(x)dx = t_i o_i I_{\sigma_1}^0(u_1, u_2) I_{\sigma_2}^0(v_1, v_2). \tag{9}$$

Transmittance Computation. Next, we compute the optimal distribution T_{i+1} to approximate the remaining transmittance, $T_i(x)(1 - \alpha_i(x))$, after blending the current splat. Specifically, we calculate the zeroth-, first-, and second-order moments (M_i^0, M_i^1, M_i^2) of the resulting distribution within the current window to preserve the total transmittance mass, mean, and variance after each blending step. They are computed in the rotated coordinate system aligned with the splat's principal axes by

$$\begin{aligned} M_i^0 &= t_i l_{i,1} l_{i,2} - \int w_i(x)dx, \\ M_i^1 &= t_i l_{i,1} l_{i,2} \begin{bmatrix} u \\ v \end{bmatrix} - t_i o_i \begin{bmatrix} I_{\sigma_1}^1(u_1, u_2) \cdot I_{\sigma_2}^0(v_1, v_2) \\ I_{\sigma_1}^0(u_1, u_2) \cdot I_{\sigma_2}^1(v_1, v_2) \end{bmatrix}, \\ M_i^2 &= t_i l_{i,1} l_{i,2} \left(\begin{bmatrix} u^2 \\ v^2 \end{bmatrix} + \frac{1}{12} l_i^2 \right) \\ &\quad - t_i o_i \begin{bmatrix} I_{\sigma_1}^2(u_1, u_2) \cdot I_{\sigma_2}^0(v_1, v_2) \\ I_{\sigma_1}^0(u_1, u_2) \cdot I_{\sigma_2}^2(v_1, v_2) \end{bmatrix}. \end{aligned} \tag{10}$$

Using these moments, we obtain the parameters of the new 2D uniform transmittance distribution T_{i+1} as

$$\begin{aligned} x_{i+1} &= \mu_i' + [e_1, e_2] M_i^1 / M_i^0, \\ l_{i+1} &= \sqrt{12 \left(M_i^2 / M_i^0 - (M_i^1 / M_i^0)^2 \right)}, \\ t_{i+1} &= M_i^0 / l_{i+1,1} l_{i+1,2}. \end{aligned} \tag{11}$$

Through this process, starting from an initially fully visible pixel area, we iteratively integrate the response of each splat and dynamically update the rendering window as in

	PSNR ↑					SSIM ↑					LPIPS ↓				
	×1	×1/2	×1/4	×1/8	Avg.	×1	×1/2	×1/4	×1/8	Avg.	×1	×1/2	×1/4	×1/8	Avg.
TensoRF	33.22	33.27	30.23	26.77	30.87	0.964	0.972	0.968	0.953	0.964	0.047	0.034	0.049	0.079	0.052
3DGS	33.57	27.04	21.43	17.74	24.95	0.970	0.950	0.876	0.767	0.891	0.037	0.036	0.071	0.130	0.068
Scaffold-GS	31.76	27.46	22.14	18.34	24.92	0.961	0.949	0.886	0.784	0.895	0.050	0.041	0.067	0.121	0.070
2DGS	31.81	26.73	20.21	16.41	23.79	0.965	0.946	0.846	0.711	0.867	0.048	0.052	0.102	0.177	0.095
3D-HGS	33.70	27.34	21.71	17.99	25.19	0.970	0.953	0.883	0.777	0.896	0.037	0.035	0.067	0.124	0.066
Tri-MipRF	32.86	32.80	28.44	24.22	29.58	0.959	0.968	0.955	0.924	0.951	0.056	0.039	0.051	0.075	0.055
3DGS+EWA	33.60	32.05	28.28	25.11	29.76	0.970	0.972	0.963	0.946	0.962	0.039	0.027	0.033	0.045	0.036
3DGS+SS	33.86	31.47	26.53	22.41	28.57	**0.971**	0.972	0.952	0.906	0.950	**0.036**	0.024	0.035	0.064	0.040
Mip-Splatting	33.54	34.09	31.50	27.80	31.73	0.969	0.976	0.977	0.968	0.973	0.038	0.022	0.022	0.032	0.028
Analytic-Splatting	33.78	34.20	31.16	27.22	31.59	0.970	0.977	0.977	0.965	0.972	**0.036**	0.022	0.025	0.037	0.030
Scaffold-GS+GB	30.22	32.02	33.84	33.80	32.47	0.949	0.964	0.976	0.983	0.968	0.070	0.041	0.025	0.019	0.039
2DGS+GB	31.96	33.53	34.93	34.74	33.79	0.960	0.970	0.979	0.985	0.974	0.055	0.033	0.022	0.015	0.031
Mip-Splatting+GB_{test}	33.45	35.37	36.98	36.36	35.54	0.969	0.978	0.984	0.988	0.980	0.038	0.021	0.014	0.012	0.021
Analytic-Splatting+GB_{test}	33.62	35.72	**37.36**	**36.51**	**35.80**	0.970	**0.979**	**0.985**	**0.989**	**0.981**	0.037	**0.020**	**0.013**	**0.011**	**0.020**
Gaussian Blending	**33.92**	**35.80**	36.82	35.79	35.58	0.970	**0.979**	**0.985**	0.988	**0.981**	**0.036**	**0.020**	**0.013**	0.012	**0.020**

Table 1: Single-scale training (×1 resolution) and multi-scale testing results on the multi-scale Blender dataset. The best results are highlighted in bold.

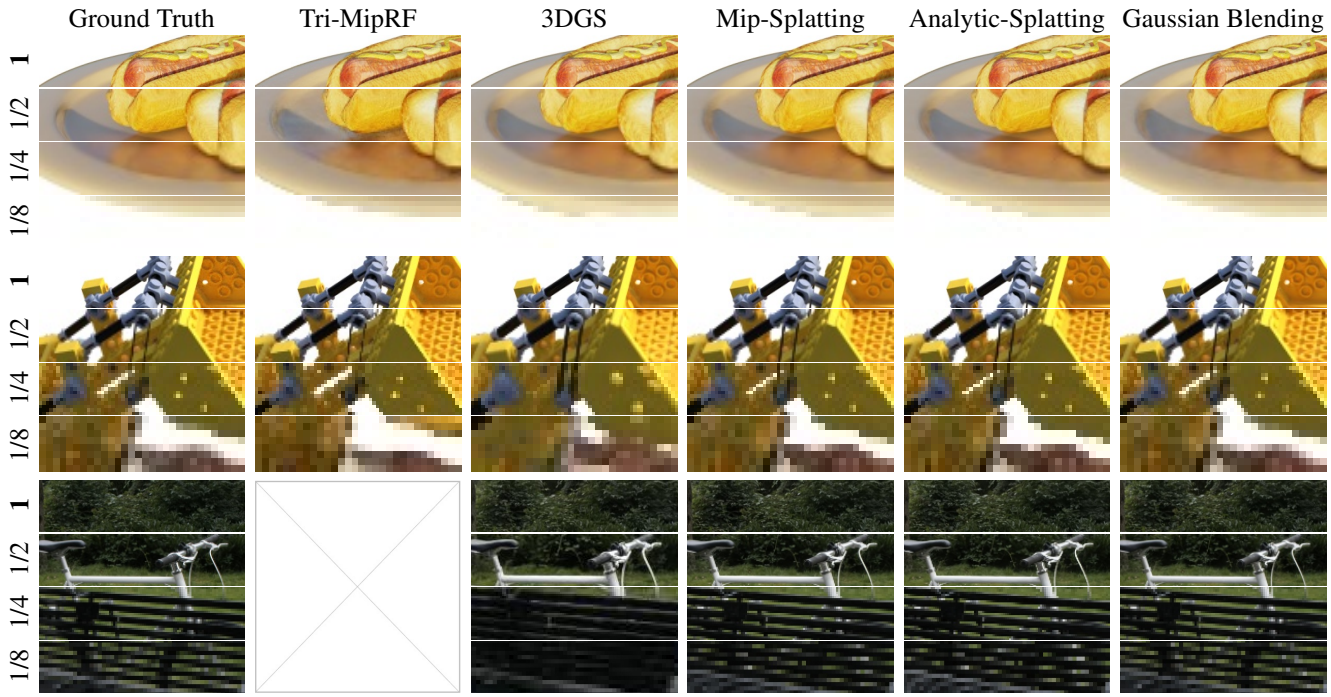

Figure 4: Qualitative results of zoom-out setting (Single-scale training (×1 resolution) and multi-scale testing). The training resolution is highlighted in bold. Our Gaussian Blending effectively suppresses pixel-level dilation in zoom-out scenarios (e.g., the disk edge in the top scene at $\times 1/8$ resolution).

Figure 2(c). This dynamic update allows us to handle spatial occlusions by adaptively shrinking the rendering window in regions of lower transmittance, reducing the influence of overlapping splats. Consequently, each splat is blended according to its spatial distribution, effectively and efficiently reducing intra-pixel aliasing within a single rendering pass.

Finally, we compute the final pixel color as

$$C_p = \sum_{i=1}^{N} c_i \int w_i(x)dx. \tag{12}$$

4 Experiments

4.1 Experiment Setup

Datasets. We evaluate novel view synthesis performance on two standard datasets: multi-scale Blender (Mildenhall et al. 2021; Barron et al. 2021) and multi-scale Mip-NeRF 360 dataset (Barron et al. 2022). The Blender dataset comprises eight synthetic scenes, each containing 100 training and 200 test images, with a resolution of 800×800 pixels. The Mip-NeRF 360 dataset includes nine real-world scenes (five out-

	PSNR ↑					SSIM ↑					LPIPS ↓				
	×1	×1/2	×1/4	×1/8	Avg.	×1	×1/2	×1/4	×1/8	Avg.	×1	×1/2	×1/4	×1/8	Avg.
TensoRF	25.42	26.57	29.02	32.87	28.47	0.874	0.902	0.946	0.977	0.925	0.159	0.132	0.078	0.028	0.099
3DGS	18.51	19.90	23.33	34.72	24.11	0.824	0.832	0.905	0.984	0.886	0.163	0.136	0.072	0.016	0.097
Scaffold-GS	16.37	17.71	21.35	34.06	22.37	0.802	0.797	0.867	0.980	0.862	0.187	0.172	0.106	0.020	0.121
2DGS	22.87	23.59	25.46	31.85	25.94	0.864	0.879	0.920	0.973	0.909	0.159	0.139	0.087	0.031	0.104
3D-HGS	18.87	20.16	23.39	35.08	24.38	0.826	0.834	0.904	0.984	0.887	0.169	0.142	0.073	0.015	0.100
Tri-MipRF	21.99	22.91	25.84	33.93	26.17	0.809	0.821	0.888	0.979	0.874	0.240	0.232	0.170	0.023	0.166
3DGS+EWA	21.76	23.15	26.64	34.32	26.47	0.834	0.871	0.938	0.983	0.907	0.231	0.184	0.095	0.019	0.132
3DGS+SS	20.98	22.47	26.33	**36.96**	26.68	0.851	0.877	0.942	**0.987**	0.914	0.136	0.100	0.048	**0.012**	0.074
Mip-Splatting	25.97	27.24	30.06	35.29	29.64	**0.892**	0.920	0.959	0.985	0.939	0.150	0.117	0.056	0.015	0.084
Analytic-Splatting	25.45	26.98	30.04	35.63	29.53	0.878	0.914	0.956	0.984	0.933	0.138	0.102	0.051	0.015	0.077
Scaffold-GS+GB	26.57	28.02	31.11	35.92	30.41	0.886	0.918	0.958	0.985	0.937	0.149	0.116	0.061	0.016	0.085
2DGS+GB	25.49	26.94	29.47	33.05	28.74	0.880	0.908	0.947	0.976	0.928	0.151	0.120	0.067	0.025	0.091
Mip-Splatting+GB_{test}	25.95	27.17	29.67	33.69	29.12	0.891	0.919	0.958	0.984	0.938	0.151	0.120	0.059	0.016	0.087
Analytic-Splatting+GB_{test}	25.37	26.87	29.51	33.34	28.78	0.876	0.912	0.954	0.982	0.931	0.139	0.104	0.053	0.017	0.078
Gaussian Blending	**26.74**	**28.53**	**31.97**	36.92	**31.04**	0.891	**0.927**	**0.965**	**0.987**	**0.943**	**0.132**	**0.092**	**0.042**	**0.012**	**0.069**

Table 2: Single-scale training ($\times 1/8$ resolution) and multi-scale testing results on the multi-scale Blender dataset. The best results are highlighted in bold.

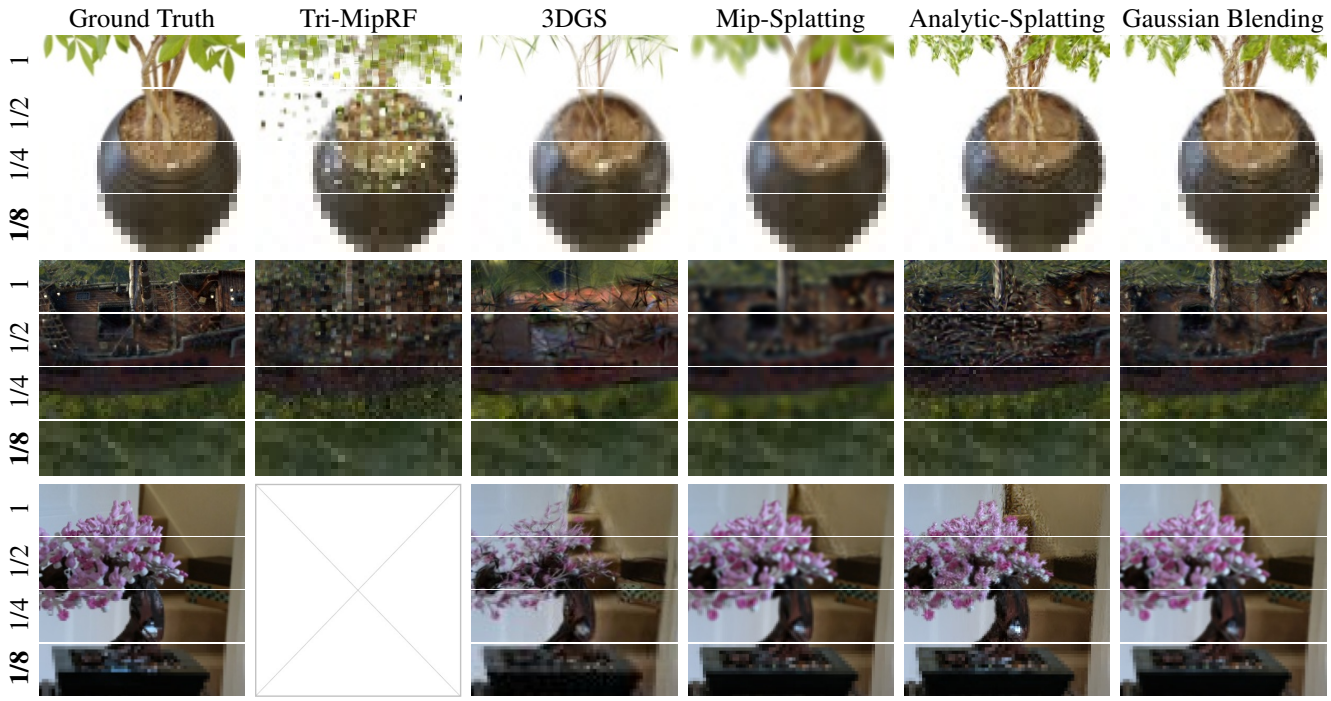

Figure 5: Qualitative results of zoom-in setting (Single-scale training ($\times 1/8$ resolution) and multi-scale testing). The training resolution is highlighted in bold. Our Gaussian Blending effectively prevents erosion in zoom-in scenarios (e.g., the leaves in the top scene at $\times 1$ resolution).

door and four indoor scenes). Following standard practice, we use every eighth image for testing, and the other images for training. For both datasets, to assess rendering quality at varying sampling rates, we also include downsampled versions of each image by factors of 2, 4, and 8.

Performance Measures. We report results using three standard metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS).

Implementation. Our Gaussian Blending algorithm is implemented in CUDA. Through careful optimization, it achieves identical time complexity and comparable runtime performance to 3DGS models without additional memory overhead. Following prior works, we train all models for 30k iterations using identical hyperparameters across all scenes.

Baselines. We compare the performance against state-of-the-art real-time NVS and anti-aliased NVS methods, including (i) five standard NVS approaches: TensoRF (Chen et al. 2022), 3DGS (Kerbl et al. 2023), Scaffold-GS (Lu et al. 2024), 2DGS (Huang et al. 2024), and 3D-HGS (Li

	(a) Single-scale Training ($\times 1$ res)					(b) Single-scale Training ($\times 1/8$ res)					(c) Multi-scale Training				
	$\times 1$	$\times 1/2$	$\times 1/4$	$\times 1/8$	Avg.	$\times 1$	$\times 1/2$	$\times 1/4$	$\times 1/8$	Avg.	$\times 1$	$\times 1/2$	$\times 1/4$	$\times 1/8$	Avg.
3DGS	27.63	25.76	21.98	19.34	23.68	17.25	18.73	22.41	30.60	22.25	26.85	27.89	28.28	27.17	27.55
Scaffold-GS	27.27	26.05	22.48	19.70	23.87	17.09	18.43	21.98	31.06	22.14	26.61	27.75	28.42	27.23	27.50
2DGS	26.71	25.99	21.84	18.41	23.24	20.92	21.66	23.85	29.49	23.98	25.70	26.73	27.72	26.52	26.67
3D-HGS	**28.07**	26.12	22.13	19.36	23.92	18.16	19.59	23.10	**31.26**	23.03	27.47	28.62	29.40	28.75	28.56
3DGS+EWA	27.67	28.40	28.23	27.19	27.87	20.27	21.76	24.90	29.38	24.08	26.50	27.75	28.93	29.25	28.11
3DGS+SS	27.67	27.71	25.47	22.59	25.86	20.14	21.50	24.84	31.22	24.42	27.37	28.54	29.69	29.44	28.76
Mip-Splatting	27.50	28.27	29.22	28.89	28.47	24.35	25.51	27.79	30.95	27.15	27.41	28.46	30.01	31.13	29.25
Analytic-Splatting	27.36	28.12	28.92	28.51	28.23	23.00	24.37	27.12	30.86	26.34	27.31	28.40	29.98	31.17	29.21
Gaussian Blending	27.55	**28.62**	**29.92**	**30.58**	**29.17**	**24.86**	**25.97**	**28.13**	30.90	**27.47**	**27.58**	**28.77**	**30.24**	**31.54**	**29.53**

Table 3: PSNR scores for multi-scale testing on the multi-scale Mip-NeRF 360 dataset. The best results are highlighted in bold.

et al. 2025), (ii) one mipmap-based anti-aliasing method: Tri-MipRF (Hu et al. 2023), and (iii) three prefiltering-based anti-aliasing methods: 3DGS+EWA (Zwicker et al. 2001), Mip-Splatting (Yu et al. 2024b), and Analytic-Splatting (Liang et al. 2024b). We also include a supersampling baseline, 3DGS+SS, which renders images at the $2\times$ target resolution and then downsamples it by half. Note that TensoRF and Tri-MipRF are tested only in the Blender dataset, since they are not applicable to unbounded scenes.

Gaussian Blending is a general rendering formulation that can function both as a standalone model and as a drop-in renderer for existing NVS frameworks. We demonstrate its flexibility across three representative integration settings: (i) for anti-aliased NVS methods such as Mip-Splatting and Analytic-Splatting, we retain each model's original training pipeline and simply substitute their rendering module with our kernel at test time, denoted as *Mip-Splatting+GB$_{test}$* and *Analytic-Splatting+GB$_{test}$*; (ii) for general 3DGS-based frameworks such as Scaffold-GS, we replace the original CUDA kernel with our Gaussian Blending kernel during both training and inference, denoted as *Scaffold-GS+GB*; and (iii) for models using non-3DGS rendering backbones, such as 2DGS, we reimplement our spatial alpha blending concept within their native kernel, yielding *2DGS+GB*.

4.2 Approximation Error Analysis

As shown in Figure 3, scalar alpha blending methods—such as 3DGS, 3DGS+SS, Mip-Splatting, and Analytic-Splatting—exhibit noticeable transmittance errors. In particular, their transmittance error ΔT become negative, indicating dilation, where the occluded splat contributes excessively to the pixel color. Although Analytic-Splatting analytically integrates each splat's response over the pixel area, it still relies on scalar alpha blending when compositing multiple splats. As a result, the first splat's contribution is computed accurately, but the second splat, which spatially overlaps with the first splat, is blended using an averaged transmittance value. This averaging neglects spatial occlusion, causing the second splat to receive an erroneously higher weight and leading to over-attenuated (dilated) transmittance after rendering. In contrast, our Gaussian Blending dynamically adjusts the transmittance window to explicitly account for spatial occlusion. As a result, Gaussian Blending achieves on average more than $5\times$ lower transmittance error compared to previous methods.

	PSNR ↑				
	$\times 1$	$\times 1/2$	$\times 1/4$	$\times 1/8$	Avg.
TensoRF	33.03	33.92	30.99	27.26	31.30
3DGS	30.19	31.38	30.59	27.05	29.90
Scaffold-GS	28.56	29.85	29.41	26.08	28.48
2DGS	27.95	29.39	30.31	26.23	28.47
3D-HGS	30.96	32.40	31.90	28.70	30.99
Tri-MipRF	32.60	34.18	34.97	35.32	34.27
3DGS+EWA	30.34	31.78	32.45	31.50	31.52
3DGS+SS	32.12	33.81	33.98	31.13	32.76
Mip-Splatting	32.93	34.68	35.79	35.48	34.72
Analytic-Splatting	33.28	35.02	36.07	35.90	35.07
Scaffold-GS+GB	29.68	31.56	33.81	34.87	32.48
2DGS+GB	31.12	32.89	34.99	36.26	33.81
Mip-Splatting+GB$_{\text{test}}$	32.85	34.69	36.32	37.00	35.22
Analytic-Splatting+GB$_{\text{test}}$	33.15	35.09	36.52	36.44	35.30
Gaussian Blending	**33.50**	**35.48**	**37.38**	**38.39**	**36.19**

Table 4: Multi-scale training and multi-scale testing results on the multi-scale Blender dataset.

4.3 Single-Scale Training and Multi-Scale Testing

To evaluate anti-aliasing performance at sampling rates not encountered during training, we experiment with a single-scale training and multi-scale testing setup. Specifically, models are trained at each of the resolutions ($\times 1$, $\times 1/2$, $\times 1/4$, or $\times 1/8$), and their rendering quality is tested across all resolutions. Quantitative results demonstrate that Gaussian Blending consistently outperforms competing methods in both zoom-out (Table 1 and 3(a)) and zoom-in (Table 2 and 3(b)) scenarios, achieving particularly impressive gains in zoom-out settings. Moreover, our qualitative results in Figure 4 and 5 further illustrate that Gaussian Blending successfully mitigates aliasing artifacts at sampling rates unseen during training.

Models employing scalar alpha blending suffer from pixel-level dilation and erosion artifacts. Mipmap-based models struggle with rendering at unseen sampling rates due to extrapolation issues, thus creating floaters from the unseen resolution mipmap. Mip-Splatting avoids sparsity issues during zoom-in due to its 3D smoothing filter, but it produces blurry renderings and fails to accurately reconstruct high-frequency details. In contrast, our model prevents saturation of the transmittance by avoiding rendering

Method	PSNR ↑	FPS ↑	Training Time ↓	Memory (MB) ↓
TensoRF	30.87	0.83	18m 18s	72.59
3DGS	24.95	131.80	16m 2s	61.86
Scaffold-GS	24.92	134.68	12m 2s	8.14
2DGS	23.79	95.35	19m 12s	25.97
3D-HGS	25.19	184.33	10m 43s	61.52
Tri-MipRF	29.58	3.18	6m 33s	58.31
3DGS+EWA	29.76	104.71	13m 18s	56.19
3DGS+SS	28.57	79.21	21m 56s	60.84
Mip-Splatting	31.73	131.58	10m 10s	71.28
Analytic-Splatting	31.59	72.07	11m 37s	69.62
Gaussian Blending-7K	34.02	128.49	2m 3s	55.17
Gaussian Blending-15K	35.24	112.45	6m 6s	72.82
Gaussian Blending-30K	**35.58**	123.08	12m 34s	72.82

Table 5: Efficiency comparison across different NVS methods. We report average zoom-out quality (PSNR), $\times 1$ resolution rendering speed (FPS), training time, and storage memory usage in the multi-scale Blender dataset.

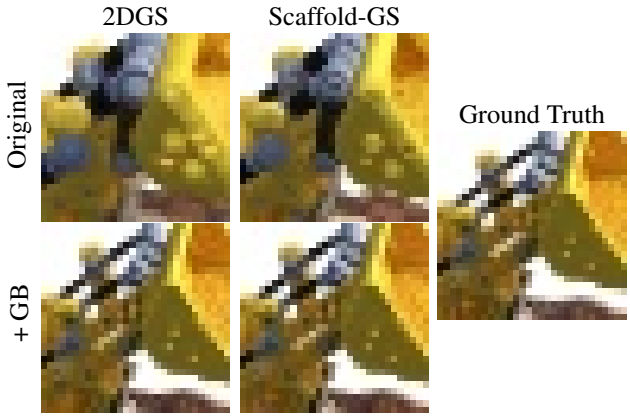

Figure 6: Qualitative comparison of Gaussian Blending applied to 2DGS and Scaffold-GS.

in a low transmittance area. As shown in Table 5, Gaussian Blending achieves real-time rendering without any additional memory overhead and converges within only 7K iterations—over $3\times$ faster than the baselines—while already surpassing them in anti-aliasing performance.

4.4 Multi-Scale Training and Multi-Scale Testing

To further demonstrate the robustness of Gaussian Blending in handling varying sampling rates, we also experiment with multi-scale training and multi-scale testing. In this scenario, models are simultaneously trained on images across all four resolutions: $\times 1$, $\times 1/2$, $\times 1/4$, and $\times 1/8$, and then evaluated on the same scales to assess whether models can effectively learn from multi-scale data.

Table 4 and Table 3(c) show that our model outperforms all other real-time and unbounded NVS methods. Our model consistently captures spatial variations without introducing dilation artifacts across diverse sampling rates, even when trained on complex scenes spanning a wide range of sampling rates.

4.5 Drop-in Gaussian Blending

To further demonstrate the generality of our formulation, we apply Gaussian Blending to various NVS frameworks with different rendering backbones, following the three integration types described in Section 4.1. Across all integration settings, Gaussian Blending consistently enhances rendering quality and anti-aliasing performance. Remarkably, even when the splat size and spatial distribution are learned according to each model's native renderer—such as in Mip-Splatting and Analytic-Splatting—simply replacing their rendering kernel with ours *at test time only* prevents pixel-level dilation. This demonstrates that Gaussian Blending offers a more physically consistent integration of alpha and transmittance, effectively reducing pixel-level dilation and preserving high-frequency details, all without retraining or modifying the underlying representation.

Moreover, Gaussian Blending exhibits strong robustness to unseen sampling rates, including both zoom-in and zoom-out scenarios. While conventional scalar alpha blending tends to blur boundaries under high sampling rates and introduce staircase-like dilation under low rates, our method maintains stable transmittance accumulation and consistent image quality across scales. When the model is trained directly using our kernel, the improvements become even more pronounced across all resolutions, confirming that Gaussian Blending functions as a unified rendering formulation suitable for both training and inference. Qualitative comparisons in Figure 6 further support these findings: both *Scaffold-GS+GB* and *2DGS+GB* deliver sharper object boundaries compared to their original rendering kernels. Our spatial alpha blending concept generalizes beyond 3DGS and can be applied to any neural rendering framework that employs an integrable alpha or density function.

5 Conclusion

We introduced *Gaussian Blending*, a novel spatial alpha blending method to address intra-pixel aliasing artifacts arising from traditional scalar alpha blending used in existing novel view synthesis (NVS) methods. By representing alpha and transmittance as spatially varying distributions within pixel regions, Gaussian Blending effectively mitigates erosion and dilation artifacts when rendering views at sampling rates unseen during training. Leveraging the spatial coherence of Gaussian splats, our approach maintains identical computational complexity and memory usage to standard 3DGS. Extensive experiments demonstrate that Gaussian Blending consistently synthesizes higher-quality views across both single-scale and multi-scale scenarios without additional priors or retraining. Furthermore, it can be seamlessly integrated as a drop-in replacement for existing NVS frameworks, providing an efficient and practical anti-aliasing solution.

While Gaussian Blending performs well even on complex real scenes, it cannot resolve boundary blurring caused by real camera effects such as diffraction, chromatic aberration, and defocus blur. Additionally, since the zoom-in setting is an ill-posed problem, high-frequency noise can occur when highly specular objects are present (e.g., the *drums* scene). Addressing these issues to further enhance performance in complex real-world scenarios remains future work.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-II220156, Fundamental research on continual meta-learning for quality enhancement of casual videos and their 3D metaverse transformation), Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2025-25442338, AI star Fellowship Support Program(Seoul National Univ.)), Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(RS-2023-00274280), IITP(Institute of Information & Communications Technology Planning & Evaluation)-ITRC(Information Technology Research Center) grant funded by the Korea government(Ministry of Science and ICT)(IITP-2025-RS-2024-00437633), and Center for Applied Research in Artificial Intelligence(CARAI) grant funded by Defense Acquisition Program Administration(DAPA) and Agency for Defense Development(ADD) (UD230017TD). Gunhee Kim is the corresponding author.

References

Barron, J. T.; Mildenhall, B.; Tancik, M.; Hedman, P.; Martin-Brualla, R.; and Srinivasan, P. P. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, 5855–5864.

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5470–5479.

Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, 333–350. Springer.

Feng, X.; He, Y.; Wang, Y.; Yang, Y.; Li, W.; Chen, Y.; Kuang, Z.; Fan, J.; Jun, Y.; et al. 2024. SRGS: Super-Resolution 3D Gaussian Splatting. *arXiv preprint arXiv:2404.10318*.

Haque, A.; Tancik, M.; Efros, A. A.; Holynski, A.; and Kanazawa, A. 2023. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19740–19750.

Hu, W.; Wang, Y.; Ma, L.; Yang, B.; Gao, L.; Liu, X.; and Ma, Y. 2023. Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19774–19783.

Huang, B.; Yu, Z.; Chen, A.; Geiger, A.; and Gao, S. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery.

Jeong, J.; Koo, J.; Zhang, Q.; and Kim, G. 2024. ESR-NeRF: Emissive Source Reconstruction Using LDR Multi-view Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4598–4609.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4): 139–1.

Li, H.; Liu, J.; Sznaier, M.; and Camps, O. 2025. 3D-HGS: 3D Half-Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10996–11005.

Li, J.; Shi, Y.; Cao, J.; Ni, B.; Zhang, W.; Zhang, K.; and Van Gool, L. 2024. Mipmap-GS: Let Gaussians Deform with Scale-specific Mipmap for Anti-aliasing Rendering. *CoRR*.

Liang, Z.; Zhang, Q.; Feng, Y.; Shan, Y.; and Jia, K. 2024a. Gs-ir: 3d gaussian splatting for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 21644–21653.

Liang, Z.; Zhang, Q.; Hu, W.; Zhu, L.; Feng, Y.; and Jia, K. 2024b. Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. In *European conference on computer vision*, 281–297. Springer.

Lin, C.; Pan, P.; Yang, B.; Li, Z.; and Mu, Y. 2025. DiffSplat: Repurposing Image Diffusion Models for Scalable Gaussian Splat Generation. *arXiv preprint arXiv:2501.16764*.

Lu, T.; Yu, M.; Xu, L.; Xiangli, Y.; Wang, L.; Lin, D.; and Dai, B. 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20654–20664.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

Moenne-Loccoz, N.; Mirzaei, A.; Perel, O.; de Lutio, R.; Martinez Esturo, J.; State, G.; Fidler, S.; Sharp, N.; and Gojcic, Z. 2024. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM Transactions on Graphics (TOG)*, 43(6): 1–19.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4): 1–15.

Park, K.; Sinha, U.; Hedman, P.; Barron, J. T.; Bouaziz, S.; Goldman, D. B.; Martin-Brualla, R.; and Seitz, S. M. 2021. HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields. *ACM Transactions on Graphics (TOG)*, 40(6): 1–12.

Poole, B.; Jain, A.; Barron, J. T.; and Mildenhall, B. 2023. DreamFusion: Text-to-3D using 2D Diffusion. In *The Eleventh International Conference on Learning Representations*.

Pumarola, A.; Corona, E.; Pons-Moll, G.; and Moreno-Noguer, F. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10318–10327.

Ren, K.; Jiang, L.; Lu, T.; Yu, M.; Xu, L.; Ni, Z.; and Dai, B. 2024. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898*.

Shannon, C. E. 1949. Communication in the presence of noise. *Proceedings of the IRE*, 37(1): 10–21.

Sun, C.; Sun, M.; and Chen, H.-T. 2022. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5459–5469.

Wang, C.; Wu, X.; Guo, Y.-C.; Zhang, S.-H.; Tai, Y.-W.; and Hu, S.-M. 2022. Nerf-sr: High quality neural radiance fields using supersampling. In *Proceedings of the 30th ACM International Conference on Multimedia*, 6445–6454.

Wang, P.; Liu, L.; Liu, Y.; Theobalt, C.; Komura, T.; and Wang, W. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *Advances in Neural Information Processing Systems*, 34: 27171–27183.

Wu, T.; Wang, J.; Pan, X.; Xudong, X.; Theobalt, C.; Liu, Z.; and Lin, D. 2023. Voxurf: Voxel-based Efficient and Accurate Neural Surface Reconstruction. In *The Eleventh International Conference on Learning Representations*.

Xia, J.; and Liu, L. 2025. Close-up-GS: Enhancing Close-Up View Synthesis in 3D Gaussian Splatting with Progressive Self-Training. *arXiv preprint arXiv:2503.09396*.

Yan, Z.; Low, W. F.; Chen, Y.; and Lee, G. H. 2024. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20923–20931.

Yang, Z.; Yang, H.; Pan, Z.; and Zhang, L. 2024. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *The Twelfth International Conference on Learning Representations*.

Yu, X.; Zhu, H.; He, T.; and Chen, Z. 2024a. GaussianSR: 3D Gaussian Super-Resolution with 2D Diffusion Priors. *CoRR*.

Yu, Z.; Chen, A.; Huang, B.; Sattler, T.; and Geiger, A. 2024b. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 19447–19456.

Zwicker, M.; Pfister, H.; Van Baar, J.; and Gross, M. 2001. EWA volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, 29–538. IEEE.