Deformable Beta Splatting

RONG LIU*, University of Southern California, Institute for Creative Technologies, USA

DYLAN SUN*, University of Southern California, USA

MEIDA CHEN, University of Southern California, Institute for Creative Technologies, USA

YUE WANG[†], University of Southern California, USA

ANDREW FENG[†], University of Southern California, Institute for Creative Technologies, USA



(a): GT vs Our Reconstruction

(b): Deformable Beta Kernels for modelling geometry

(c): Deformable Spherical Betas for modelling specular light

Fig. 1. Superior scene reconstruction by Beta Splatting. Our deformable Beta Kernels achieve high-fidelity geometry and appearance representation (a), while naturally enabling the decoupling of surface geometry from texture details (b) and the separation of specular and diffuse lighting components (c).

3D Gaussian Splatting (3DGS) has advanced radiance field reconstruction by enabling real-time rendering. However, its reliance on Gaussian kernels for geometry and low-order Spherical Harmonics (SH) for color encoding limits its ability to capture complex geometries and diverse colors. We introduce Deformable Beta Splatting (DBS), a deformable and compact approach that enhances both geometry and color representation. DBS replaces Gaussian kernels with deformable Beta Kernels, which offer bounded support and adaptive frequency control to capture fine geometric details with higher fidelity while achieving better memory efficiency. In addition, we extended the Beta Kernel to color encoding, which facilitates improved representation of diffuse and specular components, yielding superior results compared to SH-based methods. Furthermore, Unlike prior densification techniques that depend on Gaussian properties, we mathematically prove that adjusting regularized opacity alone ensures distribution-preserved Markov chain Monte Carlo (MCMC), independent of the splatting kernel type. Experimental results demonstrate that DBS achieves state-of-the-art visual quality while utilizing only 45% of the parameters and rendering 1.5x faster than 3DGS-MCMC, highlighting the superior performance of DBS for real-time radiance field rendering. Interactive demonstrations and source code are available on $our\ project\ website: https://rongliu-leo.github.io/beta-splatting/.$

Authors' addresses: Rong Liu, roliu@ict.usc.edu, University of Southern California, Institute for Creative Technologies, Los Angeles, California, USA; Dylan Sun, dylansun@usc.edu, University of Southern California, Los Angeles, California, USA; Meida Chen, mechen@ict.usc.edu, University of Southern California, Institute for Creative Technologies, Los Angeles, California, USA; Yue Wang, yue.w@usc.edu, University of Southern California, Los Angeles, California, USA; Andrew Feng, feng@ict.usc.edu, University of Southern California, Institute for Creative Technologies, Los Angeles, California, USA.

CCS Concepts: • Computing methodologies \rightarrow Rasterization; Shape representations; Reflectance modeling; Machine learning approaches.

Additional Key Words and Phrases: Novel View Synthesis, Radiance Fields, 3D Ellipsoidal Betas, Spherical Betas

1 INTRODUCTION

Neural Radiance Fields (NeRF) [Barron et al. 2021, 2023; Mildenhall et al. 2021; Müller et al. 2022] have demonstrated exceptional capabilities in 3D scene representation and novel view synthesis but are hindered by high computational costs and limited applicability in real-time scenarios due to their reliance on neural networkbased encoding and dense volumetric ray-marching. More recently, 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] introduced an explicit representation paradigm that shifts from volumetric raymarching to point-based splatting. By representing scenes as a set of 3D Gaussian primitives, 3DGS significantly reduces computational overhead, achieving real-time radiance field rendering while maintaining visual quality comparable to earlier NeRF methods. This explicit representation eliminates the need for heavy neural networks and enables direct manipulation of scene geometry and real-time rendering, making 3DGS ideal for applications such as immersive VR exploration [Franke et al. 2024; Jiang et al. 2024; Yu et al. 2024], dynamic scene representation [Guo et al. 2024; Luiten et al. 2023; Wu et al. 2024; Yang et al. 2023, 2024b], 3D content creation [Chung et al. 2023; Tang et al. 2024; Yi et al. 2024a,b], mesh extraction [Dai et al. 2024; Guédon and Lepetit 2023; Huang et al. 2024; Liu et al. 2024], semantic field distillation [Qin et al. 2024; Qiu

 $^{^{\}ast}\text{Co-first}$ authors, equal technical contribution.

[†]Co-advisors, equal leading contribution.

et al. 2024; Shi et al. 2023; Ye et al. 2024a; Zhou et al. 2024], and cinematic production [Wang et al. 2024]. However, while 3DGS excels in efficiency, its rendering quality still falls short of the latest NeRF advancements [Barron et al. 2023], particularly in capturing fine details and modeling view-dependent colors. To address this gap, a range of extensions has emerged, focusing on improving its visual quality through anti-alising [Liang et al. 2024; Song et al. 2024; Yan et al. 2024; Yu et al. 2023], deblurring [Lee et al. 2024; Oh et al. 2024; Peng et al. 2024; Seiskari et al. 2024; Zhao et al. 2024], appearance modeling [Svitov et al. 2024; Weiss and Bradley 2024; Xu et al. 2024; Yang et al. 2024a], and revisting densification [Bulò et al. 2024; Kheradmand et al. 2024; Liu et al. 2024; Lu et al. 2023; Niemeyer et al. 2024; Zhang et al. 2024]. Despite these efforts, 3DGS-based methods have yet to surpass the state-of-the-art NeRF methods [Kulhanek and Sattler 2024].

The limitations of 3DGS primarily arise from the constraints of its foundational components. The Gaussian kernel, while enabling real-time rendering through its explicit representation, inherently produces smooth and blurred results. This smoothing effect hinders its ability to accurately depict flat surface geometries and sharp edges, resulting in a loss of fine detail and diminished realism in rendered scenes. Additionally, the Gaussian kernel's long-tailed and unbounded nature necessitates a hard-coded cut-off for efficient rendering. This workaround introduces cut-off artifacts and further compromises the accurate representation of fine geometric details. Furthermore, 3DGS employs low-order Spherical Harmonics (SH) to encode view-dependent color information. Although SH can model lighting variations, its parameter count increases polynomially with the SH level, limiting practical use in real-time applications. As a result, 3DGS relies on lower SH levels, which inherently produce smooth and overly simplified color transitions. This prevents the accurate modeling of specular highlights and fails to decouple diffuse and specular components, resulting in suboptimal representations of complex lighting interactions. Notably, both Gaussian kernel and SH are fixed-function mappings, restricting the adaptability required to handle varying scene complexities and lighting conditions, further compromising visual fidelity.

To address these shortcomings, we propose Deformable Beta Splatting (DBS), a novel framework designed to enhance both geometric and color representation in radiance field rendering. At the core of DBS is the Beta Kernel, inspired by the Beta distribution, which provides bounded support and flexible shape control. This kernel adapts naturally to diverse scene complexities, enabling accurate representation of flat surfaces, sharp edges, and smooth regions with high fidelity. For color representation, we introduce the Spherical Beta (SB) function for view-dependent color encoding. By decoupling diffuse and specular components, SB effectively models sharp specular reflections and captures high-frequency lighting effects. Compared to spherical harmonics (SH) of degree 3, SB achieves superior performance using only 31% of the parameters, significantly improving model efficiency. Finally, we revisit and adapt Markov Chain Monte Carlo (MCMC) principles from 3DGS-MCMC [Kheradmand et al. 2024] into the Deformable Beta Splatting framework to address unique challenges posed by deformable kernels. Specifically, the adaptive density control in 3DGS and the probability-preserved densification strategies in 3DGS-MCMC were designed for Gaussian

kernels, making their direct application to deformable Beta kernels nontrivial. To overcome this, we mathematically prove that by regularizing opacity, adjusting opacity alone is valid for distribution-preserved densification, regardless of the number of densifications or the choice of splatting kernel. This results in a kernel-agnostic MCMC strategy, eliminating the need for complex heuristics and scale recomputation, thereby streamlining both the optimization and densification processes. Additionally, we redefine the noise function used in MCMC with the Beta kernel to maintain mathematical consistency and compatibility with opacity constraints.

The comprehensive experimental results demonstrate that our approach consistently outperforms both 3DGS-based and NeRF-based methods in visual quality across all benchmarks. Furthermore, our method utilizes only 45% of the parameters required by 3DGS and renders 1.5x faster than 3DGS-MCMC, demonstrating DBS's superior performance in real-time radiance field rendering.

To summarize, the main contributions of our paper are as follows:

- Beta Kernel for Geometry Representation: A bounded, flexible kernel that improves geometric fidelity by accurately capturing flat surfaces, sharp edges, and smooth regions.
- (2) Spherical Beta for Color Encoding: An efficient view-dependent color model that decouples diffuse and specular components, enabling sharp specular highlights with fewer parameters.
- (3) Kernel-Agnostic MCMC for optimization: A distributionpreserved densification leveraging regularized opacity and a redefined Beta noise function, together enhancing optimization stability and efficiency independent of kernel shape.

2 RELATED WORK

This section reviews key approaches in novel view synthesis, including implicit NeRFs and explicit 3DGS, highlighting how our work builds on these advancements and contributes to high-fidelity, real-time 3D rendering.

2.1 Neural Radiance Fields

The introduction of Neural Radiance Fields (NeRFs) [Mildenhall et al. 2021] revolutionized 3D scene representation and novel view synthesis. By encoding a continuous volumetric scene within a multilayer perceptron (MLP), NeRFs jointly learn scene density and view-dependent color, enabling photorealistic novel view synthesis through differentiable volumetric rendering.

Since their inception, NeRFs have spurred extensive research to address challenges in convergence speed, computational efficiency, and scalability. Early advancements, such as FastNeRF [Garbin et al. 2021] and PlenOctrees [Yu et al. 2021], improved efficiency through caching strategies and spatial acceleration structures. Methods like NeRF++[Zhang et al. 2020] and DeRF[Rebain et al. 2021] introduced optimized scene decompositions and novel optimization techniques, further accelerating training and inference. Additionally, techniques such as TensoRF [Chen et al. 2022] and DVGO [Sun et al. 2022] leveraged tensor factorization and voxel grids to reduce memory requirements and computation time. To address scalability, methods like Mip-NeRF 360 [Barron et al. 2021], Block-NeRF [Tancik et al.

2022], and UrbanNeRF [Rematas et al. 2022] introduced scene partitioning and multiscale representations for handling large and unbounded scenes. Meanwhile, approaches such as NeRF-W [Martin-Brualla et al. 2021] disentangled appearance variations, and Ref-NeRF [Verbin et al. 2022] extended NeRFs to handle reflective and refractive effects, advancing realism and robustness. Additional advancements, such as Instant-NGP [Müller et al. 2022] and Zip-NeRF [Barron et al. 2023], accelerated NeRF training by introducing a multi-resolution hash encoding and addressed aliasing issues inherent in grid-based models by incorporating anti-aliasing techniques, continuously pushing the boundaries of efficiency and quality in neural scene representation.

High-fidelity NeRF rendering requires dense ray sampling and extensive network evaluations, hindering real-time deployment. Its implicit scene representation also complicates integration with game engines and industrial pipelines, which prefer explicit geometry. These challenges drive the exploration of compact representations, explicit models, and novel optimization techniques to enable interactive, practical applications.

2.2 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] offers a compelling alternative to NeRF by representing scenes as collections of anisotropic Gaussian ellipsoids that capture local properties such as position, density, and view-dependent appearance. Combining point-based rendering[Zwicker et al. 2001] with modern machine learning strategy, it enables flexible and efficient scene representation. However, despite its computational efficiency, 3DGS lags behind NeRF in photorealism, struggling with fine details, aliasing artifacts, and complex view-dependent effects.

Research has addressed these shortcomings across several fronts. Anti-aliasing methods such as Mip-Splatting [Yu et al. 2023], multiscale approaches [Yan et al. 2024], analytic approximations [Liang et al. 2024], and scale-adaptive filtering [Song et al. 2024] have been proposed to reduce jagged edges and shimmering, thereby smoothing the rendered output. Deblurring techniques have also been explored to enhance clarity. Methods leveraging small MLPs [Lee et al. 2024], event camera data [Peng et al. 2024], joint optimization of Gaussian parameters with camera trajectory recovery [Oh et al. 2024; Seiskari et al. 2024; Zhao et al. 2024] have improved the sharpness of reconstructions from blurred inputs. These approaches integrate motion models and depth-aware filters into the 3DGS pipeline to refine both geometry and appearance.

Beyond addressing aliasing and blurring, splatting kernel modeling has emerged as an attractive direction for improving rendering quality and efficiency. 2DGS introduced a perspective-accurate 2D splatting pipeline that leverages ray-splat intersections and rasterization [Huang et al. 2024]. Li et al. proposed 3D Half Gaussian Splatting, which optimizes the Gaussian representation by restricting it to half-space for better modeling of occlusions and interactions in complex 3D scenes [Li et al. 2024]. Hamdi et al. replaced Gaussians with a Generalized Exponential Function, capturing sharp edges with fewer primitives [Hamdi et al. 2024]. Chen et al. developed Linear Kernel Splatting, swapping Gaussian kernels for linear ones to accelerate computation while preserving high-fidelity output [Chen

et al. 2024]. Recent advances in splatting kernels include 3D Convex Splatting [Held et al. 2024], which swaps 3D Gaussians for smooth, convex 3D primitives to better resolve edges and planes, and Deformable Radial Kernel Splatting [Huang et al. 2025], which enriches 2D Gaussians with learnable radial basis functions plus optimized ray-primitive intersection and culling for flexible, boundary-sharp shape modeling. Despite advancements in splatting kernel designs, the absence of an aligned view-dependent color modeling and dedicated densification strategies for deformable kernels constrains further improvements in final rendering quality.

In the absence of a coherent view-dependent color framework, sophisticated appearance modeling becomes essential to elevate 3DGS rendering fidelity. Xu et al. introduced Texture-GS, disentangling appearance from geometry by mapping 2D textures onto 3D surfaces via an MLP for high-fidelity editing [Xu et al. 2024]. Svitov et al. proposed BBSplat, using textured planar primitives with learnable RGB textures and alpha maps, achieving over 1200 FPS [Svitov et al. 2024]. Weiss and Bradley developed Gaussian Billboards, unifying 2D Gaussian Splatting with UV-parameterized textures for expressive scene representations [Weiss and Bradley 2024]. Yang et al. introduced Spec-Gaussian, leveraging an MLP to model anisotropic spherical Gaussians for complex lighting [Yang et al. 2024a]. While existing methods focus on either speed (via billboard textures) or fidelity (via MLPs, with slower rendering), we propose Spherical Beta, which explicitly models appearance by decoupling diffuse and specular components, inspired by the Phong Reflection Model [Phong 1975].

Likewise, the lack of bespoke kernel densification strategies makes revisiting optimization and densification methods vital for unlocking further performance gains. Techniques such as Scaffold-GS [Lu et al. 2023], RadSplat [Niemeyer et al. 2024], Pixel-GS [Zhang et al. 2024], and AtomGS [Liu et al. 2024] focus on efficient Gaussian placement and density control. Notably, 3DGS-MCMC [Kheradmand et al. 2024] introduces stochastic processes to refine the sampling and placement of Gaussians, enhancing both rendering quality and speed. While these strategies effectively capture scene details and balance rendering quality with computational cost, they are tailored specifically to Gaussian kernels. We therefore extend the principles of 3DGS-MCMC to propose a kernel-agnostic densification strategy that both stabilizes the optimization process and enforces consistent spatial distribution across arbitrary kernel types.

3 DEFORMABLE BETA SPLATTING

The primary bottleneck in Gaussian Splatting-based approaches lies in their reliance on a fixed, smooth splatting kernel and low-order SH encoding, which inherently hampers their ability to capture flat surfaces and specular highlights. To overcome this limitation, we first derive a deformable Beta Kernel function. We then illustrate how this flexible kernel can be applied to both geometric splatting and color encoding, enabling the modeling of varying geometric complexities and lighting frequencies, which enhances the overall scene representation. Finally, we address the unique challenges introduced by employing a deformable kernel and detail how our optimization strategy effectively manages these issues.

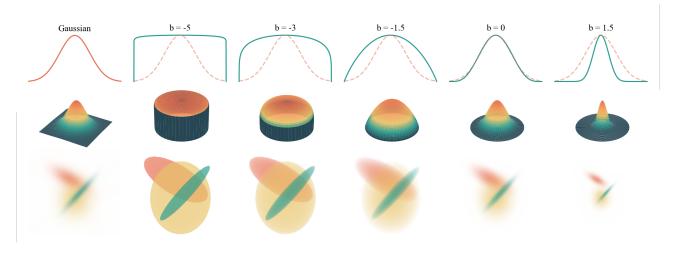


Fig. 2. Unlike the fixed Gaussian Kernel, Deformable Beta Kernel adapts its shape to capture fine geometric and texture details. This figure shows how the kernel varies by different b values as in Equation (3). Negative b generates flatter surface but sharper cutoffs for learning solid support and sharp geometry, while positive b generates sharper peaks for learning high frequency details. When b = 0, the beta kernel is almost identical to Gaussian (domain scaled by 3σ)

3.1 The Beta Kernel

The Beta Kernel is derived from the Beta distribution [Johnson et al. 1995], defined as a power of the input x and its reflection (1 - x):

$$f(x; \alpha, \beta) = \frac{1}{\mathbf{B}(\alpha, \beta)} x^{\alpha - 1} (1 - x)^{\beta - 1}, \quad x \in [0, 1], \tag{1}$$

where **B** is the Beta Function for normalization, and $\alpha, \beta > 0$ are shape-controlling parameters.

To obtain a bell-like shape that promotes smooth 3D view consistency, we fix $\alpha=1$, focusing solely on the reflection component. By removing the normalization term, we simplify the Beta distribution to the Beta Kernel:

$$\mathcal{B}(x;\beta) = (1-x)^{\beta}, \quad x \in [0,1], \beta \in (0,\infty).$$
 (2)

The Beta Kernel is a power function over the closed domain $x \in [0,1]$, inherently compact and bounded, with the parameter β introducing adaptive shape control. It is worth noting that not every function maintains multi-view consistency or qualifies as a valid splatting kernel. We have employed the inverse Abel transform to prove that the Beta Kernel meets these criteria (please refer to Appendix A for details).

However, direct optimization of β can bias the kernel toward low-frequency representations because as β approaches zero, the Beta Kernel emphasizes lower frequencies, while high-frequency shapes require $\beta \to \infty$. To allow for unbiased optimization while maintaining the proper range of β , we reparameterize β using an exponential activation: $\beta(b) = e^b$. Moreover, we want the Beta Kernel to initially resemble a Gaussian-like shape and then gradually adapt to learn different geometry forms. To achieve this, we compute a constant c=4 such that $\int_0^1 (1-x)^{ce^b} dx \approx \int_0^1 e^{-\frac{9x}{2}} dx$ when b=0. This ensures that the kernel starts with a Gaussian-like function when b is zero initialized.

As a result, we parameterize the Beta Kernel as:

$$\mathcal{B}(x;b) = (1-x)^{\beta(b)}, \quad \beta(b) = 4e^b, \quad x \in [0,1], \quad b \in \mathbb{R}.$$
 (3)

This parameterization ensures that the Beta Kernel begins with a Gaussian-like shape when b is initialized to zero and can adapt its shape during optimization. It eliminates the need for manual fine-tuning of initial β values, effectively making the Beta Kernel a more flexible superset of the Gaussian Kernel.

3.2 3D Ellipsoidal Beta Primitive

We define a 3D Ellipsoidal Beta Primitive as a set of parameters:

$$B = \{ \boldsymbol{\mu}, o, \boldsymbol{q}, \boldsymbol{s}, b, \boldsymbol{f} \} \tag{4}$$

where the geometry parameters μ , o, q, and s mirror those in 3DGS. Specifically, $\mu \in \mathbb{R}^3$ denotes the center of the primitive in 3D space, $o \in [0,1]$ represents opacity, $q \in [-1,1]^4$ defines the rotation through a quaternion, and $s \in [0,\infty)^3$ determines the scale of the ellipsoidal shape.

The key innovation in our approach is the introduction of the parameter $b \in \mathbb{R}$, which controls the shape of the Beta Kernel. As illustrated in Figure 2, varying b deforms the kernel shape, with b=0 yielding a function nearly identical to a Gaussian. This additional parameter provides adaptive shape control with minimal overhead, allowing the kernel to dynamically adjust from a Gaussian-like function to forms that can capture sharper edges and flat surfaces as optimization progresses.

For appearance modeling, each primitive carries a feature vector $f \in \mathbb{R}^d$ that encodes view-dependent color information. This vector is crucial for representing complex lighting interactions and will be discussed in detail in the next subsection.

The rendering process for a pixel $x \in \mathbb{R}^2$ unfolds as follows: each 3D Ellipsoidal Beta Primitive is projected into the 2D image plane using the viewing transformation W. This results in a 2D projected center $\mu' \in \mathbb{R}^2$ and a corresponding 2D covariance matrix $\Sigma' = JW\Sigma W^\mathsf{T}J^\mathsf{T}$, where $\Sigma = RSS^\mathsf{T}R^\mathsf{T}$ with R and S derived from the quaternion and scale, respectively. We then compute the distance

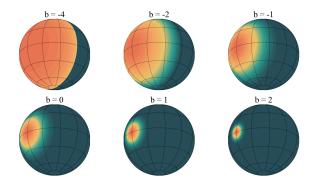


Fig. 3. Learnable Spherical Beta can effectively capture specular highlights with varying sharpness. Smaller b values result in broader reflections, while higher b values correspond to sharper specular highlights.

between the pixel and each primitive center:

$$r_i(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu}_i')^{\top} {\Sigma_i'}^{-1} (\mathbf{x} - \boldsymbol{\mu}_i')}.$$
 (5)

After sorting overlapping primitives by depth relative to the camera, denoted by the ordered set $\mathcal{N} = \{B_1, \dots, B_N\}$, we composite their contributions to the final pixel color C(x) using the Beta Kernel:

$$C(\mathbf{x}) = \sum_{i=1}^{N} c_i o_i \mathcal{B}(r_i(\mathbf{x})^2; b_i) \prod_{j=1}^{i-1} \left(1 - o_j \mathcal{B}(r_j(\mathbf{x})^2; b_j) \right)$$
(6)

Spherical Beta

3DGS employs Spherical Harmonics (SH) for view-dependent color encoding. For a SH of degree N, the feature dimension scales as $3(N+1)^2$, resulting in a quadratic increase in parameters. To achieve real-time performance in practical applications, 3DGS is constrained to using low-order SH (e.g., N = 3), which only provide smooth viewdependent colors and struggle to model sharp specular highlights effectively. To address this, we introduce Spherical Beta (SB) function, which is inspired by the Phong Reflection Model [Phong 1975]:

$$c(\hat{V}) = A_m + \sum_{m \in \mathcal{M}} \left[D_m + (\hat{R}_m \cdot \hat{V})^{\alpha_m} c_m \right], \tag{7}$$

where A_m is the ambient term, D_m and $(\hat{R}_m \cdot \hat{V})^{\alpha_m} c_m$ models the diffuse and specular components for light source m. Here, \hat{R}_m is the normalized reflection direction, \hat{V} is the normalized view direction, α_m is the shininess coefficient controlling specularity, and c_m is the color of light source. Note that the reflection vector \hat{R}_m is derived from the surface normal and corresponding light source directions. In contrast, Spherical Beta merges ambient and diffuse into a single base color, then directly models specular lobes via learnable bounded Beta kernels.

$$c(\hat{V}) = c_0 + \sum_{m \in \mathcal{M}} \mathcal{B}(1 - \hat{R}_m \cdot \hat{V}; b_m) c_m \tag{8}$$

where c_0 represents the diffuse or base color for the Beta primitive. For each 3D Beta primitive, we introduce $M = |\mathcal{M}|$ SB lobes modeling outgoing reflected radiance. For each SB lobe $m \in \mathcal{M}$, it is paramertized by reflection direction \hat{R}_m , color c_m , and shininess b_m .

We can observe that its feature dimension scales as 3 + 6M, growing linearly with the number of reflection lobes and making it more efficient than SH.

Spherical Beta shares similarities with Spherical Gaussian in modeling specularities but offers improvements. Unlike Spherical Gaussian, which have unbounded support and require truncation-leading to radiance discontinuities when view and reflection directions are orthogonal-the Beta Kernel is inherently bounded. Its input range is confined to [0, 1], ensuring that radiance naturally drops to zero when $\hat{R}_m \cdot \hat{V} = 0$. This eliminates artifacts associated with cutoff truncation and maintains continuity in radiance.

Figure 3 illustrates the specular highlights and corresponding reflection lobes for different b_m values, demonstrating the capability of Spherical Beta to model both high and low frequency specularities. By integrating bounded support and adaptive shininess, Spherical Beta enhances the ability to model complex lighting interactions and specular effects with fewer parameters, thereby improving both visual fidelity and computational efficiency.

Kernel-Agnostic Markov Chain Monte Carlo

One of the key innovations of 3DGS is its optimization with densification strategy, which utilizes heuristic gradient thresholds to clone, split, and prune primitives. Additionally, 3DGS-MCMC redefines Gaussian Splatting optimization as a Markov Chain Monte Carlo (MCMC) process by treating each primitive's opacity as a probability. This approach identifies "dead" primitives—those with opacity below a pruning threshold—and respawns them by relocating to live Gaussians through multinomial sampling based on their opacity values. By adjusting both opacity and scale, this method preserves the underlying probability distribution during densification, thereby stabilizing the process. Moreover, the introduction of positional noise encourages exploration and prevents overfitting.

However, these heuristic strategies and scale adjustments are inherently tied to the properties of the Gaussian function, making them difficult to directly apply to arbitrary kernels.

To address this limitation, we present Kernel-Agnostic MCMC, building upon the framework established by 3DGS-MCMC [Kheradmand et al. 2024]. We first detail the optimization process and then revisit distribution-preserved densification.

Optimization. To encourage exploration and prevent overfitting, we adopt a noise term ϵ that perturbs the positions of primitives during optimization. The primitive positions μ are then updated as

$$\mu \leftarrow \mu - \lambda_{lr} \cdot \nabla_{\mu} \mathcal{L} + \lambda_{\epsilon} \cdot \epsilon, \quad \epsilon = \lambda_{lr} \cdot \mathcal{B}(o_i; b') \cdot \Sigma_{\eta}.$$
 (9)

Note that we replace the logit noise function from 3DGS-MCMC with a Beta Kernel function for a more compact and well-defined noise function. Here, λ_{ϵ} controls the impact of the noise term, ensuring that the primitive is not only optimized through backpropagation of the loss function but also explores alternative solutions for better representation. λ_{lr} represents the learning rate, and Σ_n is the sampled noise. $\mathcal{B}(o_i; b')$ is the Beta Kernel component, which scales the noise based on the opacity of each primitive, with $b' = \ln(25)$ to resemble the fall-off behavior of the original logit function.



Fig. 4. Geometry Decomposition: our designed Beta Kernel provides the capability to decompose scene geometry into fundamental structures and intricate details, such as high-frequency textures and fine surface variations.



Fig. 5. Qualitative Comparison: Leveraging our Deformable Beta Kernel, Spherical Beta color encoding, and Kernel-Agnostic MCMC optimization process, the Deformable Beta Splatting framework qualitatively outperforms the implicit SOTA method (Zip-NeRF) and the explicit SOTA method (3DGS-MCMC).

To train the Beta Primitives, we employ a composite loss function:

$$\mathcal{L} = (1 - \lambda_{\text{SSIM}}) \mathcal{L}_1 + \lambda_{\text{SSIM}} \mathcal{L}_{\text{SSIM}} + \lambda_o \sum_i |o_i| + \lambda_{\Sigma} \sum_i \sum_j |\sqrt{\text{eig}_j(\Sigma_i)}|,$$
(10)

where \mathcal{L}_1 is the pixel-wise L1 loss that enforces accurate color reproduction in the rendered image, and $\mathcal{L}_{\text{SSIM}}$ is the Structural Similarity Index Measure (SSIM) loss that promotes structural consistency across the scene. The hyperparameters $\lambda_{\text{SSIM}}, \lambda_o$, and λ_{Σ} balance the contributions of each loss term.

The term $\lambda_o \sum_i |o_i|$ acts as an opacity regularizer to ensure the kernel-agnostic densification and encourage primitive to explore. The eigenvalues $\operatorname{eig}_{i}(\Sigma_{i})$ of the covariance matrix Σ_{i} for the *i*-th primitive are used in the regularization term $\lambda_{\Sigma} \sum_{i} \sum_{j} |\sqrt{\operatorname{eig}_{j}(\Sigma_{i})}|$, which together with opacity regularizer encourage primitives to disappear and respawn.

Densification. Here, we mathematically demonstrate that regularizing opacity and adjusting it alone are both effective and valid, regardless of the number of clones or the chosen splatting kernel.

Assume a primitive defined by a splatting kernel f(x) with opacity o and it will be copied N times, [Bulò et al. 2024; Kheradmand et al. 2024] established that

$$o' = 1 - \sqrt[N]{1 - o},\tag{11}$$

Incorporating f(x) complicates preservation beyond the primitive center, especially at boundaries. Prior approaches like 3DGS-MCMC handled boundary preservation via scale adjustments tailored to Gaussian kernels, which are complex and not directly applicable to deformable kernels.

However, we find that given a regularized small o value, we approximate o' using a Taylor expansion $\sqrt[N]{1-o} \approx 1 - \frac{o}{N}$, such that

$$o' = 1 - \sqrt[N]{1 - o} \approx \frac{o}{N} \tag{12}$$

Now consider the densified distribution:

$$1 - \left(1 - o' \cdot f(x)\right)^N \approx 1 - \left(1 - \frac{o}{N}f(x)\right)^N.$$
 (13)

Since $\frac{o}{N} f(x)$ is also small, applying the binomial approximation:

$$\left(1 - \frac{o}{N}f(x)\right)^{N} \approx 1 - N \cdot \frac{o}{N}f(x) + O(o^{2}) = 1 - of(x) + O(o^{2}), (14)$$

we find:

$$1 - \left[1 - of(x) + O(o^2)\right] = of(x) + O(o^2), \tag{15}$$

which matches the original distribution $o \cdot f(x)$.

The error term $O(o^2)$ becomes negligible as we employ an opacity regularizer to maintain a small o. Therefore, the difference between original distribution and densified distribution diminishes by regularizing opacity regardless of copy times N and splatting kernel f(x). Detailed proof can be found in Appendix D.

EXPERIMENTS

We evaluate Deformable Beta Splatting (DBS) on a diverse set of real-world and synthetic datasets to demonstrate its superior visual

quality and rendering efficiency compared to existing state-of-theart methods. Additionally, DBS effectively decouples scene geometry and lighting information, leveraging the learnable frequency characteristics of the Beta Kernel to enhance scene representation.

4.1 Experimental Setup

Implementations. Our implementation builds upon the 3DGS-MCMC framework [Kheradmand et al. 2024], incorporating modifications to the GSplat library [Ye et al. 2024b]. Specifically, we integrate the Beta Kernel and Spherical Beta (SB) functions, which are fully explicit and differentiable, implemented in CUDA for optimized performance. By default, we utilize two light sources in Spherical Beta component per primitive (i.e., sb = 2), unless otherwise specified. To ensure that the Beta Kernel initially resembles the Gaussian Kernel, we set b to zero at initialization. We observe that different scenes exhibit varying convergence speeds when training the model for the default 30,000 iterations. To accommodate these variations, we have integrated a flexible stopping strategy with patience of 10k iterations into the framework, allowing for two training modes: (Ours, 30K) and (Ours, Full). Detailed training times and iteration counts for each scene are provided in Appendix B. For additional hyperparameters and configuration settings, please refer to Appendix F.

4.2 Results and Comparisons

Radiance Field Reconstruction. We evaluate DBS using the standard 3DGS evaluation pipeline across several widely recognized datasets, including Mip-NeRF 360 [Barron et al. 2021], Tanks and Temples [Knapitsch et al. 2017], Deep Blending [Hedman et al. 2018], and the NeRF Synthetic dataset [Mildenhall et al. 2021]. Performance is measured using standard metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). For the Mip-NeRF 360, Tanks and Temples, and Deep Blending datasets, we utilize the Colmap Structure-from-Motion (SfM) point cloud provided by the 3DGS [Kerbl et al. 2023] repository for initialization. In contrast, for the NeRF Synthetic dataset, primitives are randomly initialized to simulate diverse starting conditions. We benchmark DBS against prior works, including the state-of-the-art implicit method (ZipNeRF [Barron et al. 2023]) and explicit method (3DGS-MCMC [Kheradmand et al. 2024]). The quantitative results of comparisons are presented in Table 1.

Beta Splatting demonstrates superior performance for radiance field reconstruction across all metrics over almost all datasets compared to prior methods. Both the 30K and Full training modes consistently outperform previous implicit and explicit state-of-the-art methods. See Appendix B for detailed per-scene analysis. Qualitative visual comparisons in Figure 5 reveal that while Zip-NeRF tends to exhibit floater artifacts, and 3DGS-MCMC struggles with rendering sharp specularities, DBS maintains high fidelity in both smooth and highly detailed regions, effectively capturing intricate lighting interactions and geometric details.

Efficiency Analysis. To evaluate efficiency, we compare our method with Zip-NeRF [Barron et al. 2023] and 3DGS-MCMC [Kheradmand et al. 2024] in terms of training efficiency, rendering efficiency, and

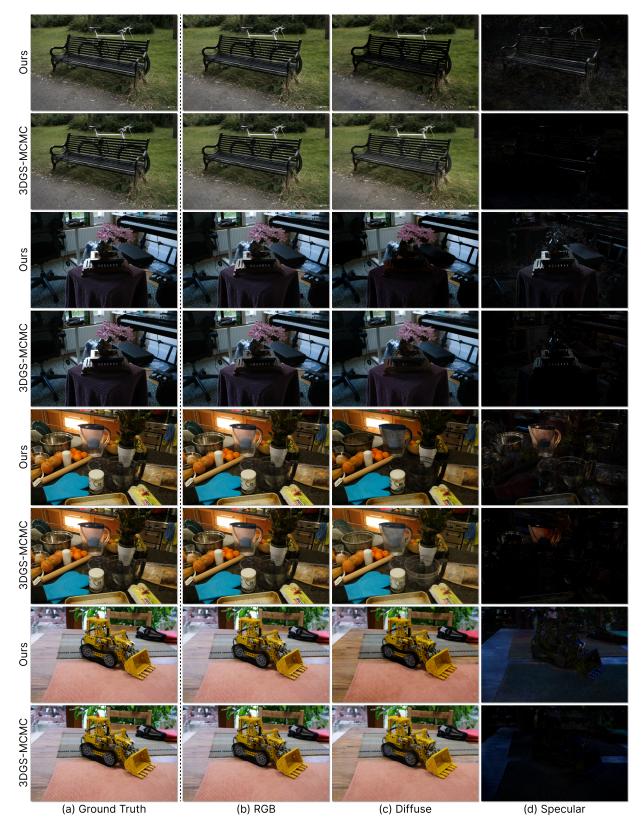


Fig. 6. Light Decomposition: our Spherical Beta color encoding can effectively decompose diffuse and specular components compared to 3DGS-based approaches using low-order Spherical Harmonics, as illustrated in this figure.

	Methods		Mip-NeRF360			Tanks&Temples			Deep Blending			NeRF Synthetic		
			SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	
cit	Instant-NGP [Müller et al. 2022]	25.51	0.684	0.398	21.72	0.722	0.330	23.62	0.797	0.423	33.18	0.959	0.055	
implicit	Mip-NeRF360 [Barron et al. 2021]	27.69	0.792	0.237	22.22	0.758	0.257	29.40	0.900	0.245	33.25	0.962	0.039	
.E	Zip-NeRF [Barron et al. 2023]	28.54	0.828	0.189	-	-	-	-	-	-	33.10	0.971	0.031	
	3DGS [Kerbl et al. 2023]	27.20	0.815	0.214	23.15	0.840	0.183	29.41	0.903	0.243	33.31	0.969	0.037	
	GES [Hamdi et al. 2024]	26.91	0.794	0.250	23.35	0.836	0.198	29.68	0.901	0.252	-	-	-	
Ħ	2DGS [Huang et al. 2024]	27.04	0.805	0.297	-	-	-	-	-	-	33.07	-	-	
explicit	Mip-Splatting [Yu et al. 2023]	27.79	0.827	0.203	-	-	-	-	-	-	33.33	0.969	0.039	
ex	3DGS-MCMC [Kheradmand et al. 2024]	28.29	0.840	0.210	24.29	0.860	0.190	29.67	0.895	0.320	33.80	0.970	0.040	
	Ours (30k)	28.60	0.844	0.182	24.79	0.868	0.148	30.10	0.910	0.240	34.64	0.973	0.028	
	Ours (full)	28.75	0.845	0.179	24.85	0.870	0.140	30.12	0.914	0.236	34.66	0.973	0.028	

Table 1. Quantitative evaluation was conducted on the Mip-NeRF360 [Barron et al. 2021], Tanks&Temples [Knapitsch et al. 2017], and Deep Blending [Hedman et al. 2018] datasets. We adopt the original number from corresponding papers when available. A dash ("-") indicates results that were not reported.

Method	Mip-NeRF 360							
Method	Training Time↓	Storage Mem [↓]	Rendering FPS [↑]					
Zip-NeRF	5h29m	569.7MB	0.184					
3DGS	21m	797.06MB	116.15					
3DGS-MCMC	31m	733.19MB	82.46					
(Ours,30k,sb=2)	16m	356.04MB	123.11					
(Ours,full,sb=2)	22m	356.04MB	123.09					

Table 2. Quantitative evaluation of training and rendering efficiency.

model size on a single NVIDIA RTX 6000 Ada Generation GPU. The summarized efficiency metrics are presented in Table 2, with additional per-scene details available in Appendix B. Our findings indicate that DBS not only achieves better reconstruction quality but also renders 1.5x faster than 3DGS-MCMC [Kheradmand et al. 2024] under the same number of primitives. When sb=2, our method uses less than half storage memory compared to 3DGS-based methods while achieving the state-of-the-art in all metrics. Additional details on applying compression methods to further reduce storage requirements are provided in Appendix E. These results highlight effectiveness of DBS in delivering high-quality renderings with enhanced computational efficiency.

Methods	Tank	s&Temples	Deep Blending		
Wethous	PSNR↑	Mem (MB)↓	PSNR↑	Mem (MB)↓	
3DGS, w/ MCMC, sh=3	24.29	437.63	29.67	750.23	
DBS, w/ MCMC, sh=3	24.45	445.05	29.88	762.94	
DBS, w/ Kernel-Agnostic MCMC, sh=3	24.57	445.05	30.03	762.94	
DBS, w/ Kernel-Agnostic MCMC, sb=1	24.53	155.77	30.04	267.03	
DBS, w/ Kernel-Agnostic MCMC, sb=2	24.79	200.27	30.10	343.32	
DBS, w/ Kernel-Agnostic MCMC, sb=4	24.81	289.28	30.12	495.91	
DBS, w/ Kernel-Agnostic MCMC, sb=6	24.80	378.29	30.08	648.50	
DBS (full)	24.85	200.27	30.12	343.32	

Table 3. Ablation Studies on Tank&Temples and Deep Blending datasets.

Ablation Study. Table 3 compares PSNR and memory usage on the Tanks & Temples and Deep Blending datasets across several DBS configurations: moving from 3DGS-MCMC to a DBS-MCMC raises PSNR from 24.29 dB to 24.45 dB (Tanks & Temples) and from 29.67 dB to 29.88 dB (Deep Blending); switching to Kernel-Agnostic MCMC further boosts PSNR to 24.57 dB and 30.03 dB with no added memory; varying the number of spherical-beta light sources (sb) reveals a trade-off: sb = 1 uses only 155.8 MB but yields 24.53 dB/30.04 dB, sb = 2 hits a sweet spot at 200.3 MB for 24.79 dB/30.10 dB, while

sb = 4 or 6 slightly increase PSNR (up to 24.81 dB/30.12 dB) at much higher memory (289-378 MB); finally, adding our flexible stopping strategy to the sb = 2 setup ("DBS (full)") delivers a final bump to 24.85 dB and 30.12 dB.

4.3 Beta Decomposition

Leveraging the deformable nature of the Beta Kernel, our framework enables decomposition of both geometry and lighting post-training. This capability arises from the dual role of the Beta Kernel in encoding volumetric splatting and view-dependent color information.

Geometry Decomposition. The Beta Kernel's parameter *b* intrinsically controls the geometry frequency representation of each primitive. Lower b values correspond to lower-frequency primitives, which predominantly capture the fundamental geometric structures of the scene. Conversely, higher b values encapsulate higherfrequency details, such as textures and fine surface variations.

By setting a mask based on the b parameter, we can selectively isolate primitives based on their frequency contributions. We can also do a simple binary split based on a threshold (e.g. the mean of b values across all primitives). Primitives with *b* below the threshold primarily represent the scene's solid geometry, with primitives of b above the threshold provides high-frequency texture details.

Figure 4 illustrates how Beta Kernels separate geometry from texture on real-world datasets. In the "treehill" scene, the bark texture is represented using primitives with beta parameter b. Similarly, in the "garden" scene, detailed textures on the tabletop are removed, preserving the overall table geometry.

Lighting Decomposition. Since Beta Splatting integrates a learnable Phong Reflection Model [Phong 1975], we can extract diffuse component through taking only the base color c_0 . Conversely, only enabling specular component gives the view-dependent specular lighting effects. For comparison, we only enable zero degree of SH for diffuse color while non zero degree for specular effects.

The results in Figure 6 illustrate the effectiveness of this approach. In the "bicycle" scene, the black bench is separated from the white highlights, a result unattainable with Gaussian Splatting. Similarly, in the "bonsai" scene, Beta Splatting successfully reconstructs the bright highlight component of the bonsai container, while Gaussian Splatting fails to capture such extreme view-dependent details.

5 CONCLUSION

We presented Deformable Beta Splatting (DBS), a novel approach that advances real-time radiance field rendering through three key innovations: Deformable Beta Kernels for adaptive geometry representation, Spherical Betas for efficient view-dependent color encoding, and Kernel-Agnostic MCMC that enhances optimization stability and efficiency by relying solely on regularized opacity. These innovations allow DBS to achieve superior visual quality with less memory and computing resources than prior methods.

Limitations. As our framework is rasterization-based, it occasionally produces popping artifacts due to depth approximation inaccuracies during the sorting process. While adaptive, Spherical Beta Functions struggle to model mirror-like reflections and anisotropic specular highlights effectively. In addition, the frustum model causes Beta Kernels to optimize flat geometries for distant backgrounds, which impacts the overall beta geometry distribution. These limitations offer opportunities for further refinement and enhancement of the framework.

ACKNOWLEDGMENTS

The authors would like to thank our primary sponsors of this research: Mr. Clayton Burford of the Battlespace Content Creation (BCC) team at Simulation and Training Technology Center (STTC). This work is supported by University Affiliated Research Center (UARC) award W911NF-14-D-0005, a Toyota Research Institute University 2.0 program, a gift fund from Dolby, and a gift fund from Google DeepMind. Statements and opinions expressed and content included do not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

REFERENCES

- Milena T. Bagdasarian, Paul Knoll, Yi-Hsin Li, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 2025. 3DGS.zip: A survey on 3D Gaussian Splatting Compression Methods. arXiv:2407.09510 [cs.CV] https://arxiv.org/abs/2407.09510
- Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. arXiv:2103.13415 [cs.CV] https://arxiv.org/abs/2103.13415
- Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2023. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. arXiv:2304.06706 [cs.CV] https://arxiv.org/abs/2304.06706
- Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. 2024. Revising Densification in Gaussian Splatting. arXiv:2404.06109 [cs.CV] https://arxiv.org/abs/2404.06109
- Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. Tensorf: Tensorial radiance fields. In European conference on computer vision. Springer, 333–
- Haodong Chen, Runnan Chen, Qiang Qu, Zhaoqing Wang, Tongliang Liu, Xiaoming Chen, and Yuk Ying Chung. 2024. Beyond Gaussians: Fast and High-Fidelity 3D Splatting with Linear Kernels. arXiv:2411.12440 [cs.CV] https://arxiv.org/abs/2411. 12440
- Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. 2023. LucidDreamer: Domain-free Generation of 3D Gaussian Splatting Scenes. arXiv:2311.13384 [cs.CV] https://arxiv.org/abs/2311.13384
- Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. 2024. High-quality Surface Reconstruction using Gaussian Surfels. arXiv:2404.17774 [cs.CV] https://arxiv.org/abs/2404.17774
- Linus Franke, Laura Fink, and Marc Stamminger. 2024. VR-Splatting: Foveated Radiance Field Rendering via 3D Gaussian Splatting and Neural Points. arXiv:2410.17932 [cs.CV] https://arxiv.org/abs/2410.17932
- Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. Fastnerf: High-fidelity neural rendering at 200fps. In Proceedings of the IEEE/CVF international conference on computer vision. 14346–14355.

- Zhiyang Guo, Wengang Zhou, Li Li, Min Wang, and Houqiang Li. 2024. Motion-aware 3D Gaussian Splatting for Efficient Dynamic Scene Reconstruction. arXiv:2403.11447 [cs.CV] https://arxiv.org/abs/2403.11447
- Antoine Guédon and Vincent Lepetit. 2023. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. arXiv:2311.12775 [cs.GR] https://arxiv.org/abs/2311.12775
- Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. 2024. GES: Generalized Exponential Splatting for Efficient Radiance Field Rendering. arXiv:2402.10128 [cs.CV] https://arxiv.org/abs/2402.10128
- Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. ACM Trans. Graph. 37, 6, Article 257 (Dec. 2018), 15 pages. https://doi.org/10.1145/ 3272127.3275084
- Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 2024. 3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes. arXiv:2411.14974 [cs.CV] https://arxiv.org/abs/2411.14974
- Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH '24). ACM, 1–11. https://doi.org/10.1145/3641519.3657428
- Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. 2025. Deformable Radial Kernel Splatting. arXiv:2412.11752 [cs.CV] https://arxiv.org/abs/2412.11752
- Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, and Chenfanfu Jiang. 2024. VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality. arXiv:2401.16663 [cs.HC] https://arxiv.org/abs/2401.16663
- Norman L Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. 1995. Continuous univariate distributions, volume 2. Vol. 289. John wiley & sons.
- Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. arXiv:2308.04079 [cs.GR] https://arxiv.org/abs/2308.04079
- Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Jeff Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 2024. 3D Gaussian Splatting as Markov Chain Monte Carlo. arXiv:2404.09591 [cs.CV] https://arxiv.org/abs/2404.09591
- Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. ACM Transactions on Graphics 36, 4 (2017).
- Jonas Kulhanek and Torsten Sattler. 2024. NerfBaselines: Consistent and Reproducible Evaluation of Novel View Synthesis Methods. arXiv (2024).
- Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. 2024.

 Deblurring 3D Gaussian Splatting. arXiv:2401.00834 [cs.CV] https://arxiv.org/abs/2401.00834
- Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 2024. 3D-HGS: 3D Half-Gaussian Splatting. arXiv:2406.02720 [cs.CV] https://arxiv.org/abs/2406.02720
- Zhihao Liang, Qi Zhang, Wenbo Hu, Ying Feng, Lei Zhu, and Kui Jia. 2024. Analytic-Splatting: Anti-Aliased 3D Gaussian Splatting via Analytic Integration. arXiv:2403.11056 [cs.CV] https://arxiv.org/abs/2403.11056
- Rong Liu, Rui Xu, Yue Hu, Meida Chen, and Andrew Feng. 2024. AtomGS: Atomizing Gaussian Splatting for High-Fidelity Radiance Field. arXiv:2405.12369 [cs.CV] https://arxiv.org/abs/2405.12369
- Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. 2023. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. arXiv:2312.00109 [cs.CV] https://arxiv.org/abs/2312.00109
- Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. 2023. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. arXiv:2308.09713 [cs.CV] https://arxiv.org/abs/2308.09713
- Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 7210–7219.
- Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. Commun. ACM 65, 1 (2021), 99–106.
- Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. 2025. Compact 3D Scene Representation via Self-Organizing Gaussian Grids. In Computer Vision – ECCV 2024. Springer Nature Switzerland, Cham, 18–34. https://doi.org/10.1007/978-3-031-73013-9
- Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics* 41, 4 (July 2022), 1–15. https://doi.org/10.1145/3528223.3530127
- Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico

- Tombari. 2024. RadSplat: Radiance Field-Informed Gaussian Splatting for Robust Real-Time Rendering with 900+ FPS. arXiv:2403.13806 [cs.CV] https://arxiv.org/
- Jeongtaek Oh, Jaeyoung Chung, Dongwoo Lee, and Kyoung Mu Lee. 2024. DeblurGS: Gaussian Splatting for Camera Motion Blur. arXiv:2404.11358 [cs.CV] https: //arxiv.org/abs/2404.11358
- Cheng Peng, Yutao Tang, Yifan Zhou, Nengyu Wang, Xijun Liu, Deming Li, and Rama Chellappa. 2024. BAGS: Blur Agnostic Gaussian Splatting through Multi-Scale Kernel Modeling. arXiv:2403.04926 [cs.CV] https://arxiv.org/abs/2403.04926
- Bui Tuong Phong. 1975. Illumination for computer generated pictures. Commun. ACM 18, 6 (June 1975), 311-317. https://doi.org/10.1145/360825.360839
- Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. 2024. LangSplat: 3D Language Gaussian Splatting. arXiv:2312.16084 [cs.CV] https: //arxiv.org/abs/2312.16084
- Ri-Zhao Qiu, Ge Yang, Weijia Zeng, and Xiaolong Wang. 2024. Splatting: Language-Driven Physics-Based Scene Synthesis and Editing. arXiv:2404.01223 [cs.CV] https://arxiv.org/abs/2404.01223
- Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. 2021. Derf: Decomposed radiance fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 14153-14161.
- Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. 2022. Urban radiance fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 12932-12942.
- Otto Seiskari, Jerry Ylilammi, Valtteri Kaatrasalo, Pekka Rantalankila, Matias Turkulainen, Juho Kannala, Esa Rahtu, and Arno Solin, 2024. Gaussian Splatting on the Move: Blur and Rolling Shutter Compensation for Natural Camera Motion. arXiv:2403.13327 [cs.CV] https://arxiv.org/abs/2403.13327
- Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. 2023. Language Embedded 3D Gaussians for Open-Vocabulary Scene Understanding. arXiv:2311.18482 [cs.CV] https://arxiv.org/abs/2311.18482
- Xiaowei Song, Jv Zheng, Shiran Yuan, Huan ang Gao, Jingwei Zhao, Xiang He, Weihao Gu, and Hao Zhao. 2024. SA-GS: Scale-Adaptive Gaussian Splatting for Training-Free Anti-Aliasing. arXiv:2403.19615 [cs.CV] https://arxiv.org/abs/2403.19615
- Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. arXiv:2111.11215 [cs.CV] https://arxiv.org/abs/2111.11215
- David Svitov, Pietro Morerio, Lourdes Agapito, and Alessio Del Bue. 2024. Bill-Board Splatting (BBSplat): Learnable Textured Primitives for Novel View Synthesis. arXiv:2411.08508 [cs.CV] https://arxiv.org/abs/2411.08508
- $Matthew\ Tancik,\ Vincent\ Casser,\ Xinchen\ Yan,\ Sabeek\ Pradhan,\ Ben\ Mildenhall,\ Pratul\ P$ Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. 2022. Block-nerf: Scalable large scene neural view synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 8248-8258.
- Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. 2024. Dream-Gaussian: Generative Gaussian Splatting for Efficient 3D Content Creation. arXiv:2309.16653 [cs.CV] https://arxiv.org/abs/2309.16653
- Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. 2022. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 5481-5490.
- Chao Wang, Krzysztof Wolski, Bernhard Kerbl, Ana Serrano, Mojtaba Bemana, Hans-Peter Seidel, Karol Myszkowski, and Thomas Leimkühler. 2024. Cinematic Gaussians: Real-Time HDR Radiance Fields with Depth of Field. In Computer Graphics Forum. Wiley Online Library, e15214.
- Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. arXiv preprint arXiv:2106.10689 (2021).
- Sebastian Weiss and Derek Bradley. 2024. Gaussian Billboards: Expressive 2D Gaussian Splatting with Textures. arXiv:2412.12734 [cs.CV] https://arxiv.org/abs/2412.12734
- Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. arXiv:2310.08528 [cs.CV] https://arxiv.org/abs/2310.08528
- Tian-Xing Xu, Wenbo Hu, Yu-Kun Lai, Ying Shan, and Song-Hai Zhang. 2024. Texture-GS: Disentangling the Geometry and Texture for 3D Gaussian Splatting Editing. arXiv:2403.10050 [cs.CV] https://arxiv.org/abs/2403.10050
- Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. 2024. Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering. arXiv:2311.17089 [cs.CV] https://arxiv.org/
- Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. 2024a. Spec-Gaussian: Anisotropic View-Dependent Appearance for 3D Gaussian Splatting. arXiv:2402.15870 [cs.CV] https: //arxiv.org/abs/2402.15870
- Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. 2023. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. arXiv:2309.13101 [cs.CV] https://arxiv.org/abs/2309.13101

- Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. 2024b. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. arXiv:2310.10642 [cs.CV] https://arxiv.org/abs/2310.10642
- Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. 2024a. Gaussian Grouping: Segment and Edit Anything in 3D Scenes. arXiv:2312.00732 [cs.CV] https://arxiv. org/abs/2312.00732
- Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. 2024b. gsplat: An Open-Source Library for Gaussian Splatting. arXiv:2409.06765 [cs.CV] https://arxiv.org/abs/2409.06765
- Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024a. GaussianDreamer: Fast Generation from Text to 3D Gaussians by Bridging 2D and 3D Diffusion Models. arXiv:2310.08529 [cs.CV] https://arxiv.org/abs/2310.08529
- Taoran Yi, Jiemin Fang, Zanwei Zhou, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Xinggang Wang, and Qi Tian. 2024b. Gaussian-DreamerPro: Text to Manipulable 3D Gaussians with Highly Enhanced Quality. arXiv:2406.18462 [cs.CV] https://arxiv.org/abs/2406.18462
- Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. Plenoctrees for real-time rendering of neural radiance fields. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 5752-5761.
- Xiaohang Yu, Zhengxian Yang, Shi Pan, Yuqi Han, Haoxiang Wang, Jun Zhang, Shi Yan, Borong Lin, Lei Yang, Tao Yu, and Lu Fang. 2024. Den-SOFT: Dense Space-Oriented Light Field DataseT for 6-DOF Immersive Experience. arXiv:2403.09973 [cs.CV] https://arxiv.org/abs/2403.09973
- Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2023. Mip-Splatting: Alias-free 3D Gaussian Splatting. arXiv:2311.16493 [cs.CV] https: //arxiv.org/abs/2311.16493
- Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020).
- Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. 2024. Pixel-GS: Density Control with Pixel-aware Gradient for 3D Gaussian Splatting. arXiv:2403.15530 [cs.CV] https://arxiv.org/abs/2403.15530
- Lingzhe Zhao, Peng Wang, and Peidong Liu. 2024. BAD-Gaussians: Bundle Adjusted Deblur Gaussian Splatting. arXiv:2403.11831 [cs.CV] https://arxiv.org/abs/2403.
- Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. 2024. Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields. arXiv:2312.03203 [cs.CV] https://arxiv.org/abs/2312.03203
- Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. 2001. EWA volume splatting. In Proceedings Visualization, 2001. VIS'01. IEEE, 29–538.

Deformable Beta Splatting

Supplementary Material

A 3D SPLATTING KERNELS

A.1 Splatting Condition

In volume rendering and splatting techniques, a continuous **3D kernel function** K(x, y, z) is projected onto a 2D plane to produce a continuous **2D kernel function** K(x, y). This projection is typically performed along one axis (e.g., the *z*-axis) and is represented by the integral:

$$\int_{-\infty}^{\infty} K(x, y, z) dz = K(x, y)$$
 (16)

If K(x, y, z) is a **separable function**, such as:

• Multiplicative form:

$$K(x, y, z) = K(x) \cdot K(y) \cdot K(z) \tag{17}$$

• Additive form:

$$K(x, y, z) = K(x) + K(y) + K(z)$$
 (18)

It is straightforward to derive the corresponding 2D projection. However, when we want the kernel to be **deformable** or to have an **adjustable shape** (e.g., controlling its spread or sharpness), it may be challenging to compute the integral analytically. For example, consider the radially symmetric kernel:

$$K(x, y, z) = 1 - (x^2 + y^2 + z^2)^{\beta}$$
(19)

with $\beta > 0$ and $x^2 + y^2 + z^2 \le 1$. Computing the integral to obtain K(x,y) can be difficult due to the non-separable nature of the kernel and the dependence on β .

A.2 Inverse Splatting

To address this challenge, we can **reverse the process** by first defining the desired **2D kernel** K(r), where $r = \sqrt{x^2 + y^2}$, and then determining whether there exists a corresponding **3D kernel** K(R), with $R = \sqrt{x^2 + y^2 + z^2}$, such that the projection condition is satisfied:

$$K(r) = \int_{-\infty}^{\infty} K(R) \, dz \tag{20}$$

Assuming **radial symmetry** and **finite support** (i.e., $r \le 1$ and $R \le 1$), we can utilize the **inverse Abel transform** to derive the 3D kernel K(R) from the given 2D kernel K(r):

$$K(R) = -\frac{1}{\pi} \int_{R}^{1} \frac{dK(r)}{dr} \cdot \frac{1}{\sqrt{r^{2} - R^{2}}} dr$$
 (21)

This guarantees that any suitably defined 2D kernel admits a multi-view-consistent 3D counterpart.

Example:

Let's take the Beta Kernel as an example:

$$K(r) = (1 - r^2)^{\beta} \tag{22}$$

with $\beta > 0$ and $0 \le r \le 1$. The radial derivative of K(r) is:

$$\frac{dK(r)}{dr} = -2\beta r (1 - r^2)^{\beta - 1}.$$
 (23)

Substituting into the inverse Abel transform, we obtain the 3D kernel:

$$K(R) = \frac{2\beta}{\pi} \int_{R}^{1} \frac{r(1-r^2)^{\beta-1}}{\sqrt{r^2 - R^2}} dr.$$
 (24)

This is the multi-view-consistent geometry we observe in 3D space by defining a 2D Beta Kernel in splatted billboards.

A.3 General Conditions for 2D Kernel Functions

To ensure that the 2D kernel K(r) corresponds to a valid 3D kernel K(R) via the inverse Abel transform and satisfies the projection condition, the following conditions must be met:

- (1) Definition and Range:
 - The kernel is defined for $r \in [0, 1]$.
 - The kernel values are in the range $K(r) \in [0, 1]$.
- (2) Boundary Conditions:
 - Opacity at the center:

$$\lim_{r \to 0^+} K(r) = 1. \tag{25}$$

• Transparency at the boundary:

$$\lim_{r \to 1^{-}} K(r) = 0. \tag{26}$$

- (3) Smoothness and Differentiability:
 - The kernel K(r) is continuously differentiable on (0, 1):

$$K(r) \in C^1(0,1).$$
 (27)

- (4) Integrability:
 - The kernel and its derivative are **absolutely integrable** over (0, 1):

$$\int_0^1 K(r) \, dr < \infty \tag{28}$$

$$\int_0^1 \left| \frac{dK(r)}{dr} \right| dr < \infty \tag{29}$$

- (5) Existence of Corresponding 3D Kernel:
 - The **inverse Abel transform** of K(r) yields a valid 3D kernel K(R) that satisfies the projection condition:

$$K(R) = -\frac{1}{\pi} \int_{R}^{1} \frac{dK(r)}{dr} \cdot \frac{1}{\sqrt{r^{2} - R^{2}}} dr$$
 (30)

By enforcing the above conditions on any radial 2D function, the inverse Abel transform ensures the existence of a unique radial 3D kernel. Hence, one may prescribe any 2D splatting kernel and be guaranteed a corresponding, multi-view-consistent 3D geometry—without having to derive the 3D kernel in closed form.

		Zip-NeRF			3DGS-MCMC			(Ours,30K,sb=2)			(Ours,full,sb=2)		
	Scenes	PSNR [↑]	SSIM [↑]	$LPIPS^{\downarrow}$	PSNR [↑]	SSIM [↑]	$LPIPS^{\downarrow}$	PSNR [↑]	SSIM [↑]	LPIPS↓	PSNR [↑]	SSIM [↑]	LPIPS↓
	bicycle	25.80	0.769	0.208	26.15	0.81	0.18	26.04	0.804	0.169	26.10	0.805	0.170
360	flowers	22.40	0.642	0.273	22.12	0.642	0.31	22.44	0.650	0.308	22.49	0.652	0.306
	garden	28.20	0.860	0.118	28.16	0.89	0.10	28.15	0.882	0.096	28.11	0.881	0.095
	stump	27.55	0.800	0.193	27.80	0.82	0.19	27.56	0.815	0.184	27.50	0.817	0.178
Æ	treehill	23.89	0.681	0.242	23.31	0.66	0.29	23.49	0.676	0.293	23.50	0.680	0.289
Mip-NeRF	room	32.65	0.925	0.196	32.48	0.94	0.25	32.83	0.941	0.168	33.03	0.942	0.165
įφ.	counter	29.38	0.902	0.185	29.51	0.92	0.22	30.36	0.930	0.154	30.70	0.932	0.149
Σ	kitchen	32.50	0.928	0.116	32.27	0.94	0.14	32.61	0.939	0.110	32.84	0.940	0.107
	bonsai	34.46	0.949	0.173	32.88	0.95	0.22	33.90	0.957	0.156	34.46	0.960	0.151
	Mean	28.54	0.828	0.189	28.29	0.84	0.21	28.60	0.844	0.182	28.75	0.845	0.179
	chair	34.84	0.983	0.017	36.51	0.99	0.02	36.74	0.990	0.010	36.70	0.990	0.010
	drums	25.84	0.944	0.050	26.29	0.95	0.04	26.78	0.958	0.033	26.81	0.958	0.032
Synthetic	ficus	33.90	0.985	0.015	35.07	0.99	0.01	36.75	0.990	0.010	36.79	0.990	0.010
the	hotdog	37.14	0.984	0.020	37.82	0.99	0.02	38.85	0.988	0.015	38.95	0.988	0.016
ýn	lego	34.84	0.980	0.019	36.01	0.98	0.02	37.12	0.985	0.014	37.11	0.985	0.014
E)	materials	31.66	0.969	0.032	30.59	0.96	0.04	31.12	0.966	0.032	31.13	0.966	0.031
NeRF!	mic	35.15	0.991	0.007	37.29	0.99	0.01	37.66	0.994	0.005	37.67	0.994	0.005
_	ship	31.38	0.929	0.091	30.82	0.91	0.12	32.13	0.910	0.103	32.15	0.910	0.104
	Mean	33.10	0.971	0.031	33.80	0.97	0.04	34.64	0.973	0.028	34.66	0.973	0.028
	truck				26.11	0.89	0.14	26.44	0.897	0.110	26.59	0.901	0.106
INT	train	-	-	-	22.47	0.83	0.24	23.13	0.839	0.185	23.11	0.845	0.174
Ι	Mean	-	-	-	24.29	0.86	0.19	24.79	0.868	0.147	24.85	0.873	0.140
	drjohnson				29.00	0.89	0.33	29.39	0.908	0.240	29.53	0.910	0.235
DB	playroom	-	-	-	30.33	0.90	0.31	30.82	0.912	0.240	30.71	0.918	0.236
	Mean	-	-	-	29.67	0.90	0.32	30.10	0.910	0.240	30.12	0.914	0.236

Table 4. Quantitative evaluation of rendering results per scene

Metrics	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	Mean
Number of Primitives	6.0M	3.0M	5.0M	4.5M	3.5M	1.5M	1.5M	1.5M	1.5M	3.1M
Training Time (3DGS-MCMC,30K)	43m	26m	39m	35m	29m	27m	29m	27m	25m	31m
Rendering Time (3DGS-MCMC)	18.77ms	11.53ms	15.38ms	15.75ms	11.33ms	8.27ms	10.37ms	9.09ms	8.62ms	12.12ms
Rendering FPS (3DGS-MCMC)	53.30	86.72	65.00	63.54	88.26	120.86	96.43	110.06	116.01	82.46
Memory (3DGS-MCMC)	1395.42 MB	709.54 MB	1182.56 MB	1064.30 MB	827.79 MB	354.77 MB	354.77 MB	354.77 MB	354.77 MB	733.19 MB
Training Time (30K,sb=2)	23m	14m	20m	17m	12m	14m	17m	13m	12m	16m
Training Time (full,sb=2)	24m	11m	25m	14m	12m	19m	32m	28m	32m	22m
Training Steps (full,sb=2)	31.0K	26.5K	25.0K	25.5K	25.5K	48.5K	64.0K	61.5K	84.5	43.5K
Rendering Time (sb=2)	12.70ms	8.13ms	8.71ms	11.33ms	7.42ms	5.03ms	7.69ms	7.04ms	5.04ms	8.12ms
Rendering FPS (sb=2)	78.74	122.93	114.71	88.21	134.82	198.97	130.01	141.93	198.34	123.09
Memory (sb=2)	686.65 MB	343.32 MB	572.21 MB	514.98 MB	400.54 MB	171.66 MB	171.66 MB	171.66 MB	171.66 MB	356.04 MB

Table 5. Quantitative evaluation of training time and rendering efficiency per scene

PER-SCENE RESULTS

Our per-scene evaluation on the Mip-NeRF 360 dataset shows that, on diffuse scenes (e.g., "bicycle," "stump," and "flowers"), Deformable Beta Splatting achieves the comparable performance of prior works, while on scenes with strong specular reflections (e.g., "counter"), it surpasses 3DGS-MCMC and Zip-NeRF by over 1 dB in PSNR. This gain demonstrates the ability of DBS to faithfully reconstruct intricate specular lighting in challenging scenarios.

C KERNEL COMPARISON

In addition to Table 3, we isolate the effect of the splatting kernel through an ablation study, disabling all view-dependent color terms (sh=0, sb=0) and comparing the standard Gaussian kernel (3DGS) to our deformable Beta kernel (DBS). Both models are trained on the Mip-NeRF 360 training set and evaluated on test views under MCMC densification schedules.

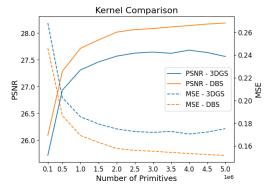


Fig. 7. Reconstruction quality on the Mip-NeRF 360 without viewdependent color-encodings. Deformable Beta Splatting (DBS) consistently outperforms 3D Gaussian Splatting (3DGS), confirming the stronger representation ability of the Beta kernel.

Figure 7 plots PSNR (left axis) and MSE (right axis) as a function of the total number of primitives. Although both kernels improve with more primitives, the Gaussian baseline plateaus around 1.5 M, while the Beta kernel continues to yield gains. Across the entire budget (0.1 M–5 M primitives), DBS consistently achieves higher PSNR and lower MSE, demonstrating that the deformable Beta kernel alone provides superior geometric modeling capacity.

D KERNEL-AGNOSTIC DISTRIBUTION-PRESERVED DENSIFICATION

D.1 Detailed Proof

Assume a primitive defined by a splatting kernel f(x) with opacity o, and it will be cloned N times, Its contribution at a pixel x is given by:

$$\alpha = o \cdot f(x) \tag{31}$$

When cloning this primitive into N copies, each with new opacity o', the combined contribution becomes:

$$\alpha' = 1 - (1 - o' \cdot f(x))^N \tag{32}$$

To preserve its distribution, we require $\alpha = \alpha'$, such that

$$o \cdot f(x) = 1 - (1 - o' \cdot f(x))^N$$
 (33)

Ignoring the kernel momentarily, the condition simplifies to:

$$o' = 1 - \sqrt[N]{1 - o},\tag{34}$$

which aligns with established proof [Bulò et al. 2024; Kheradmand et al. 2024]. Incorporating f(x) complicates preservation beyond the primitive center, especially at boundaries. Prior approaches like 3DGS-MCMC handled boundary preservation via scale adjustments tailored to Gaussian kernels, which are complex and not directly applicable to deformable kernels.

We will show that given a small opacity value, only the opacity adjustment is effective and valid regardless the densified number and kernel function.

For small o, we approximate o' using a Taylor expansion $\sqrt[N]{1-o}\approx 1-\frac{o}{N}$, such that

$$o' = 1 - \sqrt[N]{1 - o} \approx \frac{o}{N} \tag{35}$$

Now consider the densified distribution:

$$\alpha' = 1 - \left(1 - o' \cdot f(x)\right)^N \approx 1 - \left(1 - \frac{o}{N}f(x)\right)^N.$$
 (36)

Since $\frac{o}{N}f(x)$ is also small, applying the binomial approximation:

$$\left(1 - \frac{o}{N}f(x)\right)^{N} \approx 1 - N \cdot \frac{o}{N}f(x) + O(o^{2}) = 1 - of(x) + O(o^{2}), (37)$$

we find

$$\alpha' \approx 1 - \left[1 - of(x) + O(o^2)\right] = of(x) + O(o^2).$$
 (38)

This matches the original distribution:

$$\alpha = o \cdot f(x), \tag{39}$$

up to first-order in o.

The error term $O(o^2)$ becomes negligible as o decreases. Therefore, the difference between original distribution and densified distribution diminishes for smaller o regardless of copy times N and splatting kernel f(x).

D.2 Validity of the Small Opacity Assumption

A natural question arises regarding the role of the small opacity prior used in our framework. At first glance, encouraging primitives to remain semi-transparent may appear counterintuitive: in surface-based representations, higher opacity is typically desirable to achieve clean geometry. However, our approach lies in Volumetric Density field. Volumetric methods-such as NeRF and 3DGS-fundamentally differ from surface-based techniques. In a volumetric framework, each primitive contributes only a small opacity, and the final image emerges from the accumulation of many low-opacity contributions. Within this paradigm, enforcing high perprimitive opacity can improve geometric fidelity and facilitate mesh extraction, but contradicts the core assumptions of smooth transmittance accumulation and degrades visual quality. Notable examples of this trade-off include NeuS [Wang et al. 2021] vs NeRF [Mildenhall et al. 2021] and SuGaR [Guédon and Lepetit 2023] vs 3DGS [Kerbl et al. 2023]. In contrast, applying a moderate opacity prior aligns with the volumetric principle, supports distribution-preserving densification, and further stabilizes the optimization process.

To quantify the impact of the opacity regularization, we analyze its effect both theoretically and empirically. In Figure 8, we visualize the original distribution (blue) and its densified distribution (red) as defined in Equation (33), across different kernel shapes. As opacity decreases, the discrepancy between the two distributions diminishes, with near-perfect alignment observed when the opacity approaches 0.1. This behavior supports our theoretical analysis: under the small-opacity regime, the densified distribution accurately preserves the original, regardless of the number of clones or the kernel function used.

λ_o	PSNR	Opacity
0.000	27.81	0.60 ± 0.05
0.001	28.31	0.26 ± 0.05
0.005	28.61	0.14 ± 0.04
0.010	28.70	$\textbf{0.10} \pm \textbf{0.03}$
0.050	28.69	0.04 ± 0.02
0.100	25.07	0.02 ± 0.02

Table 6. Mean PSNR and Opacity Distribution on the Mip-360 dataset for different λ_0 values.

Complementing this, we conduct an ablation study on the opacity regularization weight λ_o across the Mip-NeRF 360 dataset. We report mean PSNR on held-out test views along with the mean and standard deviation of primitive opacities. As shown in Table 6, omitting the opacity term causes the densification step to naively clone primitives, resulting in over-saturated reconstructions and reducing visual quality. Introducing the opacity prior yields consistent gains in PSNR by reducing both the average opacity and its variance. However, excessively large λ_o values overly suppress opacity and drive primitives toward excessive transparency, again causing damage to the quality of reconstruction. Empirically, we observe that setting $\lambda_o = 0.01$ produces an average opacity around 0.1 and achieves peak PSNR, thereby validating our small opacity assumption.

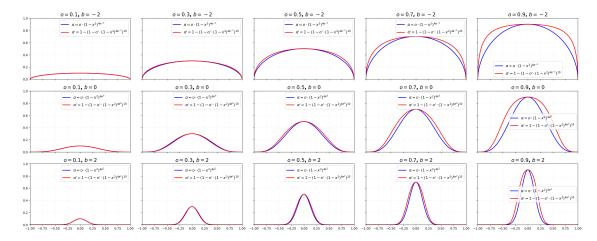


Fig. 8. Kernel-Agnostic Distribution-Preserved Densification illustration.

Method	PSNR↑	SSIM [↑]	LPIPS↓	Num (M) [↓]	Size (MB)↓	Compression Time (s)↓	Compression Ratio [↑]
3DGS	27.20	0.815	0.214	3.35	752.74	-	1
DBS-full	28.75	0.845	0.179	3.11	356.04	-	2.11
DBS-full (compressed)	28.60	0.840	0.182	3.11	63.48	41.70	11.86
DBS-1M	28.42	0.831	0.205	1.00	114.44	-	6.58
DBS-1M (compressed)	28.32	0.828	0.207	1.00	22.32	14.28	33.72
DBS-490K	28.03	0.816	0.232	0.49	56.08	-	13.42
DBS-490K (compressed)	27.80	0.806	0.237	0.49	11.37	9.15	66.20

Table 7. Quantitative evaluation of visual quality and model size on the Mip-360 dataset. More compression comparisons are available at 3DGS.zip [Bagdasarian et al. 2025].

COMPRESSION

In Table 1 and Table 2, we configured DBS-full to use the similar number of primitives as vanilla 3DGS (3.35 M) for a fair comparison. Thanks to the MCMC framework, we can precisely control the target primitive count—so we also evaluate two "pruned" variants, DBS-1M (1 M primitives) and DBS-490 (490 K primitives), to explore the tradeoff between model size and visual quality. After training, we apply a post-training compression pipeline adapted from [Morgenstern et al. 2025] to further reduce each model's on-disk footprint. Specifically, we firstly sort and reorder the parameters so that similar values are adjacent, maximizing the efficiency of lossless PNG compression. Next, we quantize most parameters from 32-bit floats down to 8-bit, allowing us to save them into PNG images. However, because we observed that reducing the precision of the 3D position to 8 bits causes a severe degradation in rendering quality, we preserve those values at full 32-bit precision, distributing them across four separate images, while all other parameters are stored as 8-bit PNGs.

We conduct these experiments on the Mip-360 dataset and report mean metrics across nine scenes in Table 7. As the model is pruned from 3.11 M primitives (DBS-full) to 1 M (DBS-1M) and then to 0.49 M (DBS-490K), we observe a graceful degradation in rendering fidelity: mean PSNR decreases from 28.75 dB to 28.42 dB to 28.03 dB, SSIM from 0.845 to 0.831 to 0.816, and LPIPS increases from 0.179 to 0.205 to 0.232. In parallel, the raw on-disk footprint contracts substantially-from 356 MB (2.11x relative to vanilla 3DGS) to 114 MB (6.58x) to 56 MB (13.42x). Applying uniform post-training PNG compression further amplifies these storage savings by approximately 5x-6x: final sizes become 63 MB, 22 MB, and 11 MB, corresponding to overall compression ratios of 11.9x, 33.7x, and 66.2x relative to 3DGS. Crucially, this additional compression incurs only minor quality losses (PSNR drops of 0.15 dB, 0.10 dB, and 0.23 dB) and remains computationally practical, requiring only 41.7 s, 14.3 s, and 9.2 s per scene, respectively. The naively compressed DBS-490K matches vanilla 3DGS's visual quality in only ~10 MB.

HYPERPARAMETERS

The training process is governed by several hyperparameters. The initial learning rate for position optimization is set to $lr_{\mu}^{init} = 0.00016$, which decays to a final value of $lr_{\mu}^{\text{final}} = 0.0000016$. A delay multiplier of 0.01 is applied, with a maximum number of steps for position learning rate decay of 30,000. The base color learning rate is lr_{c_0} = 0.0025, and the same value is used for the Spherical Beta parameters learning rate ($\ln_{b_m,\hat{R}_m,c_m}=0.0025$). The opacity learning rate is relatively high, set to $\ln_o=0.05$, while the beta learning rate is $lr_b = 0.001$. The scaling and rotation learning rates are $lr_s = 0.005$ and $lr_q = 0.001$, respectively.

Furthermore, the loss function is regularized using a DSSIM loss term with a weight of $\lambda_{SSIM} = 0.2$. Densification occurs every 100 iterations, starting from 500 and continuing until 25,000. The background is not randomized during training, and a learning rate noise of $\lambda_{lr} = 5 \times 10^4$ is used. The regularization terms for scale and opacity are $\lambda_{\Sigma} = 0.01$ and $\lambda_{o} = 0.01$, respectively.