DARB-Splatting: Generalizing Splatting with Decaying Anisotropic Radial Basis Functions

Vishagar Arunan[1] Saeedha Nazar[1] Hashiru Pramuditha[1] Vinasirajan Viruthshaan[1]
Sameera Ramasinghe[2] Simon Lucey[2] Ranga Rodrigo[1]

[1]University of Moratuwa [2]University of Adelaide

Abstract

Splatting-based 3D reconstruction methods have gained popularity with the advent of 3D Gaussian Splatting, efficiently synthesizing high-quality novel views. These methods commonly resort to using exponential family functions, such as the Gaussian function, as reconstruction kernels due to their anisotropic nature, ease of projection, and differentiability in rasterization. However, the field remains restricted to variations within the exponential family, leaving generalized reconstruction kernels largely underexplored, partly due to the lack of easy integrability in 3D to 2D projections. In this light, we show that a class of decaying anisotropic radial basis functions (DARBFs), which are non-negative functions of the Mahalanobis distance, supports splatting by approximating the Gaussian function's closed-form integration advantage. With this fresh perspective, we demonstrate up to 34% faster convergence during training and a 15% reduction in memory consumption across various DARB reconstruction kernels, while maintaining comparable PSNR, SSIM, and LPIPS results. We will make the code available.

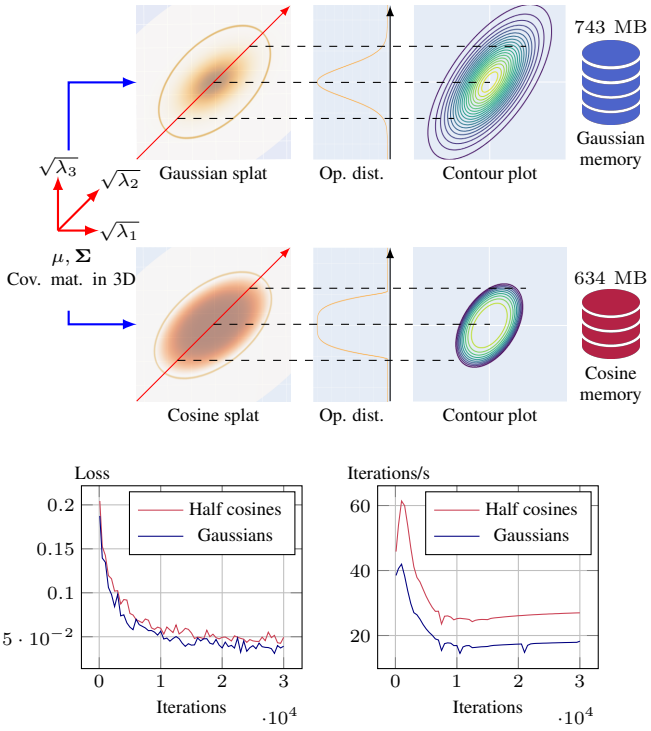


Figure 1. This figure compares two different splat functions—Gaussian and cosine (specifically, half-cosine squared)—initialized for the same 3D covariance matrix, $\mathbf{\Sigma}$. Cosine functions, defined as $\cos\left(\frac{d_M^2}{\xi}\right)$ for $d_M^2 < \frac{\xi\pi}{2}$ where $\xi(>0)$, have finite support and exhibit effective reconstruction performance comparable to Gaussians. Notably, the cosine-based approach enhances training speed by 34% and reduces memory usage by 15%, providing a more memory-efficient alternative.

1. Introduction

Splatting plays a pivotal role in modern 3D reconstruction, enabling the representation and rendering of 3D points without relying on explicit surface meshes. The recent 3D Gaussian representation, combined with splatting-based rendering in the 3D Gaussian Splatting (3DGS) [20] has gained widespread attention due to its state-of-the-art (SOTA) visual quality, real-time rendering, and reduction in training times. Since its inception, 3DGS has seen expanding applications in industry, spanning 3D web viewers, 3D scanning, VR platforms, and more, catering to large user bases. As demand continues to rise, so does the need to improve efficiency to conserve computational resources.

3DGS represents a 3D scene as a dynamically densified radiance field of 3D Gaussian primitives (ellipsoids) that act as reconstruction kernels. In theory, these ellipsoids are integrated along the projection direction onto the 2D image plane, resulting in 2D Gaussians or splats, a function known as the footprint function or splatting function. This process exploits an interesting property of Gaussians: 2×2

covariance matrix of the 2D Gaussian can be obtained by skipping the third row and column of the 3×3 covariance matrix of the 3D Gaussian [72]. This bypasses costly integration and efficiently simplifies the rendering in practical implementation. The footprint function then models the spread of each splat's opacity, eventually contributing to the final pixel color. This Gaussian-based blending effect—a smooth, continuous interpolation between splats—inspires the method's name, "3D Gaussian Splatting." Numerous subsequent papers [7, 8, 25, 63, 69] build upon this principle, thereby restricting to exponential family functions (*e.g.*, Gaussian functions) to spread each sample in image space.

In splatting, the Gaussian kernel is commonly used, with some variations within the exponential family (*e.g.*, super-Gaussian [15], half-Gaussian [28], Gaussian-Hermite [68] kernels). However, classical signal processing and sampling theory establish that while Gaussians are useful, they are not the most effective interpolators. Especially when determining the pixel color, as in 3DGS, we argue that the Gaussian function may not be the only possible interpolator. We see other functions outperforming them in various contexts, with a notable example being JPEG compression [48, 55], which leverages Discrete Cosine Transforms (DCTs) instead of Gaussians functions for image representation. Additionally, Saratchandran *et al.* [43] demonstrate that the sinc activation surpasses $\mathrm{Gaussian}$ activation in implicit neural representations [18, 21, 33, 50], while also suggesting several alternative activation functions. This raises the question: *Should we limit ourselves to Gaussians or the exponential family alone?* This remains an underexplored area in the computer vision community, partly because costly integration processes make it computationally inefficient to use functions outside of Gaussians.

To address this shortcoming, we generalize these kernels by introducing a broader class of functions, namely, Decaying Anisotropic Radial Basis Functions (DARBFs) that are non-negative (*e.g.*, modified raised cosine, half-cosine, sinc functions and *etc*.) that include but are not limited to exponential functions. These non-negative DARBFs achieve comparable reconstruction quality while significantly improving training time, along with modest reductions in memory usage (*e.g.*, half-cosine squares in Table 3 and Fig. 1). These DARBFs support anisotropic behavior as they rely on the Mahalanobis distance, and are differentiable, thereby supporting differentiable rendering (Sec. 3.1). A key innovation in our approach is a **novel correction factor that preserves the computational efficiency** of 3DGS even with alternative kernels by approximating the Gaussian's closed-form integration shortcut. This allows for effective splatting, yielding novel views, along with subtle improvements in visual quality for some functions (*e.g.*, raised cosines in Fig. 5). To the best of our knowledge, our method represents one of the first modern generalizations expanding splatting to non-exponential functions, and offering high-quality rendering as splatting techniques scale to repetitive industrial applications, where computational savings are increasingly critical.

The contributions of our paper are as follows:

- We present a unified approach where the Gaussian function is merely a special case within the broader class of DARBFs, which can be splatted.
- We leverage the DARBF family, and demonstrate reconstruction kernels achieving up to 34% faster convergence, a 15% reduced memory footprint with on-par PSNR, and enhanced visual quality with finer details in some kernels.
- We introduce a computationally feasible method to approximate the Gaussian closed-form integration advantage when implementing alternative kernels, facilitated by our novel correction factor along with CUDA-based backpropagation codes.

2. Related Work

Radiance Field Representation. Light fields [14, 27] were the foundation of early Novel View Synthesis (NVS) techniques capturing radiance in static scenes. The successful implementation of Structure-from-Motion (SfM) [49] enabled NVS to use a collection of images to capture real-world details that were previously impossible to model manually using meshes. This led to the development of the efficient COLMAP pipeline [44], which is now widely utilized in 3D reconstruction to generate initial camera poses and sparse point clouds (*e.g.*, [16, 20, 33, 70]).

Radiance field methods, notably Neural Radiance Fields (NeRFs) [33], utilize neural networks to model radiance as a continuous function in space, thereby enabling NVS. NeRF techniques generate photorealistic novel views [1–3, 32, 34, 53] through volumetric rendering, integrating color and density along rays. Recent advancements in NeRF improve training [6, 35, 60, 66] and inference times [13, 29, 30, 39, 40]; introduce depth-supervised NeRFs [10, 42] and deformable NeRFs [36–38, 71]; and enable scene editing [31, 45, 51, 62], among other capabilities. While 3DGS follows a similar image formation model to NeRFs, it diverges by using point-based rendering over volumetric rendering. This approach, leveraging anisotropic splatting, has rapidly integrated into diverse applications within a year's span, with recent work excelling [11] in physics-based simulations [23, 52, 58, 59], manipulation [7, 19, 26, 46, 67], generation [9, 41, 54, 65] and perception [47, 61, 63], among other areas. Similar to 3DGS, our approach follows point-based rendering but does not restrict points to being represented as 3D Gaussians, as Saratchandran *et al.* [43] demonstrate for NeRFs. Instead, we provide DARBFs, a class of functions to choose from to represent all points.

Splatting. Splatting, introduced by Westover [56], en-

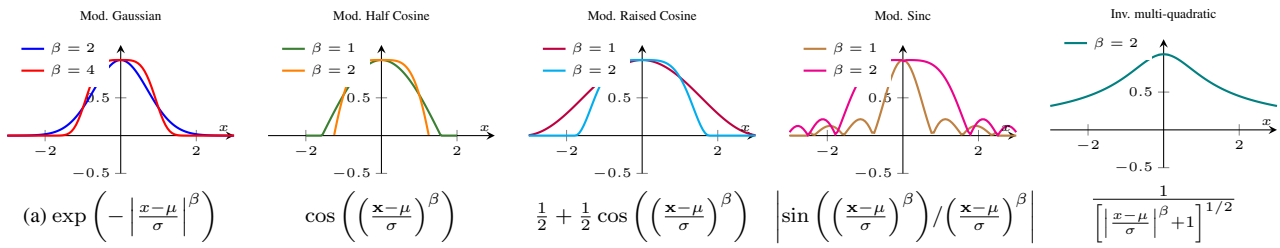


Figure 2. Overview of decaying anisotropic radial basis functions (DARBFs) and their respective 1D curves. These functions decay with distance and vary in their sensitivity to direction, making them effective for capturing anisotropic features in spatial data.

ables point-based rendering of 3D data, such as sparse SfM points [49] obtained from the COLMAP pipeline [44], without requiring mesh-like connectivity. Each particle's position and shape are represented by a volume [56] that serves as a reconstruction kernel [72], which is projected onto the image plane through a "footprint function," or "splat function," to spread, or splat each particle's intensity and color across a localized region. Early splatting techniques [57] commonly used spherical kernels for their radial symmetry to simplify calculations, although they struggled with perspective projections. To address this, elliptical kernels [57, 72], projecting elliptical footprints on the image plane were used to better approximate elongated features through anisotropic properties, ultimately enhancing rendering quality. Westover [57], further experimented with different reconstruction kernels, namely sinc, cone, Gaussian, and bilinear functions without focusing on a specific class of functions. EWA Splatting [72] further refined this approach by selecting a Gaussian reconstruction kernel in 3D, which projects as an elliptical Gaussian footprint on the image plane.

This laid the groundwork for 3DGS to select the Gaussian function. Although the Gaussian function has been extensively used to produce results in the past, other anisotropic radial basis functions exist that can serve as splats, but they lack exploration.

Reconstruction Kernel Modifications. Although 3DGS claims SOTA performance, recent work has shown further improvements by refining the reconstruction kernel. For instance, GES [15] introduces the generalized exponential function, or super Gaussian, by incorporating a learnable shape parameter for each point. However, GES does not modify the splatting function directly within the CUDA rasterizer; instead, it only approximates its effects by adjusting the scaling matrix and loss function in PyTorch, failing to fully demonstrate its superiority [68]. In contrast, we directly modify the CUDA rasterizer's splatting function to support various DARBFs, achieving superior performance in specific cases compared to 3DGS.

Concurrent work such as 3D-HGS [28], splits the 3D Gaussian reconstruction kernel into two halves, but this approach introduces additional parameters into the CUDA rasterizer, leading to increased computational costs. Similarly, 2DGH [68] extends the Gaussian function by incorporating Hermite polynomials into a Gaussian-Hermite reconstruction kernel. Although this kernel more sharply captures edges, it incurs a high memory overhead due to the increased number of parameters per primitive. In contrast, DARBF Splatting achieves efficient performance by avoiding additional parameters while still enabling flexibility in kernel choice. Furthermore, previous works are mere variants of the Gaussian kernel, remaining confined to traditional Gaussian Splatting methods. Our approach breaks this constraint, generalizing the reconstruction kernel across the DARBF class and opening new avenues for high-quality reconstruction.

3. Preliminaries

3.1. Radial Basis Functions

Radial basis functions (RBFs) serve as a fundamental class of mathematical functions where the function value solely depends on the distance from a center point. However, isotropic RBFs, which are radially symmetric, often fall short in capturing the local geometric details, such as sharp edges, flat regions, or anisotropic features that vary directionally. This limitation [5] results in isotropic RBFs inaccurately modeling directionally varying local structures.

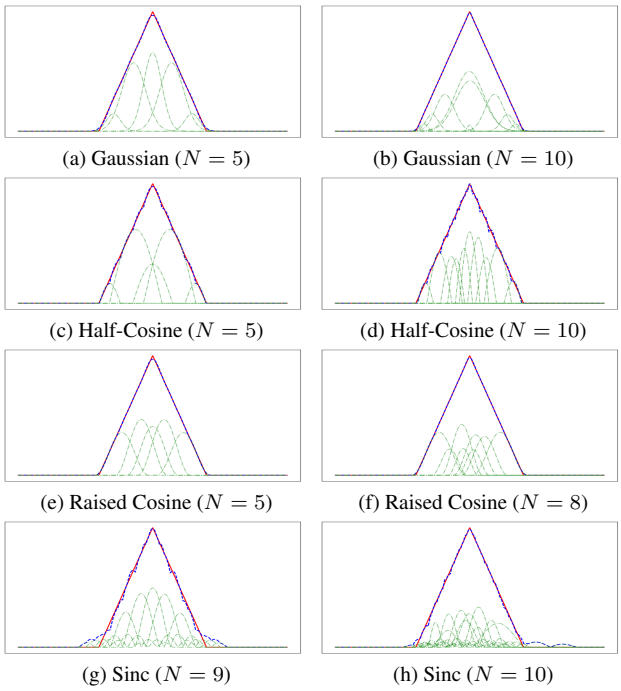
To address this issue, Anisotropic Radial Basis Functions (ARBFs) with a decaying nature, namely Decaying ARBFs (DARBFs), are used, as we are interested in splatting, where splats decays spatially. This extends the traditional RBF framework by allowing each function's influence to vary along different axes, achieved by incorporating the Mahalanobis distance. For a particular point $\mathbf{x} \in \mathbb{R}^3$, the Mahalanobis distance (d_M), calculated from the center $\mu \in \mathbb{R}^3$ of a 3D RBF, is given by:

$$d_M = \left[(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x}-\mu)\right]^{\frac{1}{2}}, \tag{1}$$

where $\mathbf{\Sigma}$ denotes the 3×3 covariance matrix. This radially dependent anisotropy naturally supports smooth interpolation and plays a pivotal role in 3D reconstruction.

3.2. Assessing DARBFs in Simulations

We evaluate and compare the performance of our DARB reconstruction kernels against conventional Gaussian kernels through simulations in lower-dimensional settings. We generated various synthetic 1D functions across a specified range, including *square, exponential, truncated-sinusoid, Gaussian, and triangular pulses*, as well as some irregular functions to simulate signal variability. These signals were reconstructed using Gaussian functions and various RBFs to evaluate the number of primitives required for accurate reconstruction and the cost difference between original and reconstructed signals.



(a) Gaussian ($N = 5$) (b) Gaussian ($N = 10$)

(c) Half-Cosine ($N = 5$) (d) Half-Cosine ($N = 10$)

(e) Raised Cosine ($N = 5$) (f) Raised Cosine ($N = 8$)

(g) Sinc ($N = 9$) (h) Sinc ($N = 10$)

Figure 3. Examples of signal reconstruction through backpropagation using various RBFs. Here the target, reconstructed and individual components are represented by red, blue and green plots, respectively. We achieve competitive results in convergence speed and computational cost. Our raised cosine kernels, for example, achieve convergence faster with fewer components ($N = 8$) and at a lower cost of 0.00004 compared to Gaussian kernels, which converge slightly slower at $N = 10$ with a comparable cost of 0.0001 in terms of mean squared error.

This pipeline optimizes the number of primitives needed for signal approximation across different reconstruction kernels. Our approach focuses on three key parameters: position, covariance (for signal spread), and amplitude (for signal strength). The model configurations comprises a mixture network of *Gaussian, half-cosine, raised-cosine, and modular sinc components*, each with a predefined variable number of components. For an extensive analysis of RBFs functions, refer our Supplementary Material.

The model was trained for a fixed number of epochs with respect to a mean squared error loss function, which minimized reconstruction error between the predicted and target signals. Unlike the original Gaussian Splatting algorithm [20], which calculates loss in the projected 2D image space with reconstruction in 3D space, we simplified by calculating the loss and performing the reconstruction directly in 1D because, 1D signal projection cannot be represented in 0D, as this would be meaningless. Finally, a parameter sweep was conducted to identify the optimal configuration with the lowest recorded loss, indicating the optimal number of components.

Table 1. Comparison of simulation results with Gaussian components vs. other reconstruction kernel components. Note that * indicates that we achieved better reconstruction with fewer components ($N = 8$) in terms of mean squared error, using our modified raised cosine pulses compared to Gaussians.

Function	**Loss** ($N = 5$)	**Loss** ($N = 10$)
Gaussian	0.0002	0.0001
Modified half cosine	0.0014	0.0003
Modified raised cosine	0.0002	0.00004*
Modified sinc (modulus)	0.0030	0.0002

The simulation results indicate that, like Gaussian functions, certain RBFs, can perform comparably or even outperform Gaussians in specific cases. Notably, raised cosines achieved lower reconstruction loss than Gaussians while requiring fewer primitives. These preliminary findings suggest promising alternatives to Gaussian functions and motivate further exploration of these RBFs. Building on this insight, we aim to incorporate these alternative functions into the Gaussian Splatting algorithm, introducing minor adjustments in domain constraints and backpropagation.

4. Method

Next, we will elaborate on the use of DARBFs for 3D reconstruction tasks. We present DARBFs as a plug-and-play replacement for existing Gaussian kernels, offering improvements in aspects such as training time and memory efficiency. The pipeline for DARB splatting is presented in Fig. 4. Splatting-based reconstruction methods such as 3DGS [20] obtain the 3D representation of a scene by placing anisotropic Gaussian kernels at 3D points proposed by a SfM pipeline such as COLMAP [44]. The 3DGS Gaussian kernel gets transformed onto the camera frame and projected to the corresponding 2D plane of existing views, resulting in splats. The error between the pixels of these views and blended (rasterized) splats generates the backpropagation signal to optimize the parameters of the 3D Gaussians. The four properties that drive this pipeline are 1. the differentiability of this rasterization, 2. the anisotropic nature of the 3D kernel function (hence the anisotropic nature of the splats), 3. easy projectability, and 4. rapid decay. As seen

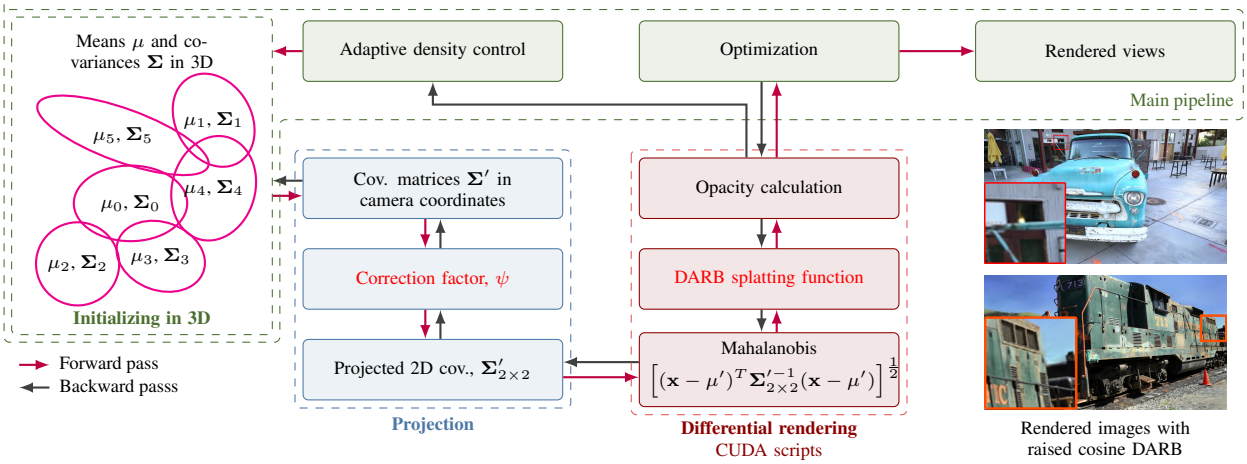


Figure 4. **Block diagram:** We use the standard optimization pipeline from 3DGS [20] with modifications, introducing a correction factor (ψ) to obtain the projected 2D covariance matrix ($\Sigma'_{2\times2}$) compatible with splatting for DARBFs within the existing framework. Our changes within the pipeline are highlighted in red text.

in [24], the reconstruction kernel function being a function of the Mahalanobis distance d_M (Eq. 1), yields the first two properties. A class of functions that display the four properties is non-negative decaying anisotropic radial basis functions (DARBFs) that are functions of Mahalanobis distance [4] d_M (Eq. 1). In Sec. 4.1 we propose a method to approximate the projection of DARBFs, thereby satisfying the third property.

4.1. DARB-Splatting

DARB Representation. Similar to 3DGS, we use the same parameterization to represent 3D DARBFs, as the Gaussian function is an instance of the DARBF class. In Table 2, we show the generic mathematical expressions of these reconstruction kernel functions, and Fig. 2 shows their 1D plots. Only some members of the class are shown here, while a broader view is provided in the *Supplementary Material*. Similar to the Gaussian, most of the chosen DARBFs are strictly decaying, meaning their envelopes decay, which makes them suitable as energy functions. The envelope of the sinc function, for instance, decays (square-integrable, hence an energy function). To optimize performance, we restricted the spread of most functions to a single pulse (central lobe), ensuring finite support along the reconstruction axes. However, we also observed that comparable results can be achieved with two or more pulses, albeit with a slight trade-off in visual quality (refer *Supplementary Material*).

DARB Projection. To project 3D DARBs to DARB-splats, we use the method proposed by EWA Splatting [64, 72]. To project 3D mean (μ) in world coordinate system onto the 2D image plane, we use the conventional perspective projection. However, the same projection cannot be used to project the 3D covariance (Σ) in world coordinates to the 2D covariance ($\Sigma'_{2\times2}$) in image space. Hence, first, the 3D covariance (Σ) is projected onto camera coordinate frame which results in a projected 3×3 covariance matrix (Σ'). Given a viewing transformation W, the projected covariance matrix Σ' in camera coordinate frame can be obtained as follows:

$$\Sigma' = JW\Sigma W^T J^T \tag{2}$$

where J denotes the Jacobian of the affine approximation of the projective transformation. Importantly, integrating a normalized 3D Gaussian along one coordinate axis results in a normalized 2D Gaussian itself. From that, the EWA Splatting [72] shows that the 2D covariance matrix ($\Sigma'_{2\times2}$) of the 2D Gaussian can be easily obtained by taking the sub-matrix of the 3D covariance matrix (Σ'), specifically

Table 2. Overview of decaying anisotropic radial basis functions (DARBFs) in 3D, with essential limits to consider for bounded function applications in 3D reconstruction. Incorporating these limits on Mahalanobis distance d_M (defined in Eq. 1) ensures precise control over the main lobe of each function and influence, similar to the 3D Gaussian, for capturing localized features in complex 3D surfaces. Here, the scaling factor $\xi(>0)$ controls the spread of the covariance while β decides the roll-off of the main lobe of each DARBF.

Function	Expression	Domain
Modified Gaussian	$\exp\left(-\frac{1}{\xi}(d_M)^\beta\right)$	$d_M \geq 0$
Modified half cosine	$\cos\left(\frac{1}{\xi}(d_M)^\beta\right)$	$\frac{1}{\xi}(d_M)^\beta \leq \frac{\pi}{2}$
Modified raised cosine	$0.5 + 0.5 \cdot \cos\left(\frac{1}{\xi}(d_M)^\beta\right)$	$\frac{1}{\xi}(d_M)^\beta \leq n\pi$
Modified sinc (modulus)	$\frac{\left\lvert\sin\left(\frac{1}{\xi}(d_M)^\beta\right)\right\rvert}{\frac{1}{\xi}(d_M)^\beta}$	$\frac{1}{\xi}(d_M)^\beta \leq \frac{(n+1)\pi}{2}$
Inv. multi-quadratic	$\frac{1}{\left[\frac{1}{\xi}(d_M)^2+1\right]^{\frac{1}{2}}}$	$d_M \geq 0$

by skipping the third row and third column (Eq. 4).

However, this sub-matrix shortcut applies only to Gaussian reconstruction kernels due to their inherent properties. Our core argument questions whether the integration described above is essential for the splatting process. While the Gaussian's closed form integration is convenient, we propose interpreting the 3D covariance as a parameter that represents the 2D covariances from all viewing directions. When viewing a 3D Gaussian from a particular direction, we may think that the opacity would be distributed across the volume defined by each 3D Gaussian, and that alpha blending composites these overlapping volumes. In practice, however, the opacity distribution takes place after projecting 3D Gaussians to 2D splats. Thus, when a ray approaches, it only considers the Gaussian value derived from the 2D covariance of the splat.

As a simple example, consider two 3D Gaussians that are nearly identical in all properties except for the variance along the z-axis ($(\sigma'_z)^2$). When these two Gaussians are projected into 2D, we expect the 3D Gaussian with a higher σ'_z to have greater opacity values. In contrast, 3DGS yields a similar matrix ($\Sigma'_{2\times2}$) for both covariances, while exerting the same influence on opacity distribution along the splats. This further strengthens our argument for treating Σ' as a parameter to represent $\Sigma'_{2\times2}$.

Generally, for DARBFs, integrating a 3D DARB along a certain axis does not yield the same DARBF in 2D, nor does it result in a closed-form solution. Therefore, we established the relationship between Σ' and $\Sigma'_{2\times2}$ for DARBFs through simulations. For clarity, we briefly provide the derivations for the half-cosine squared splat here, whereas detailed derivations and corresponding CUDA script modifications for all DARBFs are provided in the *Supplementary Material*. The 3D half-cosine squared kernels is as follows:

$$P_{i,j,k} = \begin{cases} \cos\left(\frac{\pi}{18}(d_M)^2\right), & \text{if } (d_M)^2 < 9, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

where d_M (Eq. 1) is the Mahalanobis distance. We then calculate the projected $\Sigma'_{2\times2}$ by summing values along a certain axis, similar to integrating along the same axis. Through experimentation with various values, we determined that the estimated $\Sigma'_{2\times2}$ 2D covariance matrix is also symmetric, due to the strict decomposition of the 3D covariance matrix to maintain a positive definite matrix, independent of the third row and column of Σ'. Additionally, we obtain $\Sigma'_{2\times2}$ by multiplying Σ' with a correction factor ψ (Eq. 4), which we estimate separately for each reconstruction kernel using Monte Carlo experiments. For half-cosine squares we estimated $\psi = 1.36$.

$$\Sigma' = \begin{pmatrix} a & b & c \\ b & d & e \\ e & d & f \end{pmatrix} \rightarrow \psi \begin{pmatrix} a & b \\ b & d \end{pmatrix} = \Sigma'_{2\times2}. \tag{4}$$

Similarly, based on such empirical results, we modified the existing CUDA scripts of 3DGS [20] to implement each DARBF separately with its respective ψ value to ensure comparable results. Additionally, we introduce a scaling factor ξ to match the extent of our function closely with the Gaussian, allowing a fair comparison of DARBF performance. Although Gaussian functions are spatially infinite, 3DGS [20] uses a bounded Gaussian to limit the opacity distribution in 2D, thereby reducing unnecessary processing time. For projected 2D splats, they calculate the maximum radius as $R = 3 \cdot \sqrt{\max\{\lambda_1, \lambda_2\}}$, where λ_1 and λ_2 denote the eigenvalues of the 2D covariance matrix $\Sigma'_{2\times2}$. Using this radius R, we obtain a similar extent for DARBFs as Gaussians with the help of ξ. Taking that into consideration, the 2D half-cosine squared splat is defined as follows:

$$w = \cos\left(\frac{(x-\boldsymbol{\mu'})^T\left(\Sigma'_{2\times2}\right)^{-1}(x-\boldsymbol{\mu'})}{\xi}\right), \tag{5}$$

where $x \in \mathbb{R}^2$ is the position vector, $\mu' \in \mathbb{R}^2$ is the projected mean, and $\Sigma'_{2\times2} \in \mathbb{R}^{2\times2}$ represents the covariance matrix. Here, R_{cosine} will be $\frac{\sqrt{2\pi\xi\max(\lambda_1,\lambda_2)}}{2}$, indicating 100% extent, as we consider the entire central lobe. However, for a 2D Gaussian with an extent of 99.7% (up to three standard deviations), the radius R_{Gaussian} is given by $3\sqrt{\max(\lambda_1,\lambda_2)}$. To match the extent of both functions, we select $\xi(>0)$ such that $\frac{\sqrt{2\pi\xi\max(\lambda_1,\lambda_2)}}{2} = 3\sqrt{\max(\lambda_1,\lambda_2)}$, which implies $\xi = \frac{18}{\pi}$.

Within this bounded region, we use our footprint function (Eq. 5) to model the opacity, similar to the approach in 3DGS. Following this, we apply alpha blending to determine the composite opacity [20, 64] for each pixel, eventually contributing to the final color of the 2D rendered image.

4.2. Backpropagation and Error Calculation

To support backpropagation, we must account for the differentiability of DARBFs. As usual, the backpropagation process begins with a loss function that combines the same losses discussed in the original work [20]: an $\mathcal{L}_1$ loss and an SSIM (Structural Similarity Index Measure) loss, which guide the 3D DARBs to optimize its parameters.

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} \tag{6}$$

where λ denotes each loss term's contribution to the final image loss. After computing the loss, we perform backpropagation, where we explicitly modify the differentiable Gaussian rasterizer from 3DGS [20] to support respective gradient terms for each DARBF. The following example demonstrates these modifications for the 3D half-cosine squared function (Eq. 5) with $\xi = \frac{18}{\pi}$.

$$\frac{dw}{d(x-\boldsymbol{\mu'})} = \frac{-2\left(\Sigma'_{2\times2}\right)^{-1}(x-\mu')\sin\left(\frac{(x-\boldsymbol{\mu'})^T(\Sigma'_{2\times2})^{-1}(x-\boldsymbol{\mu'})}{\xi}\right)}{\xi} \tag{7}$$

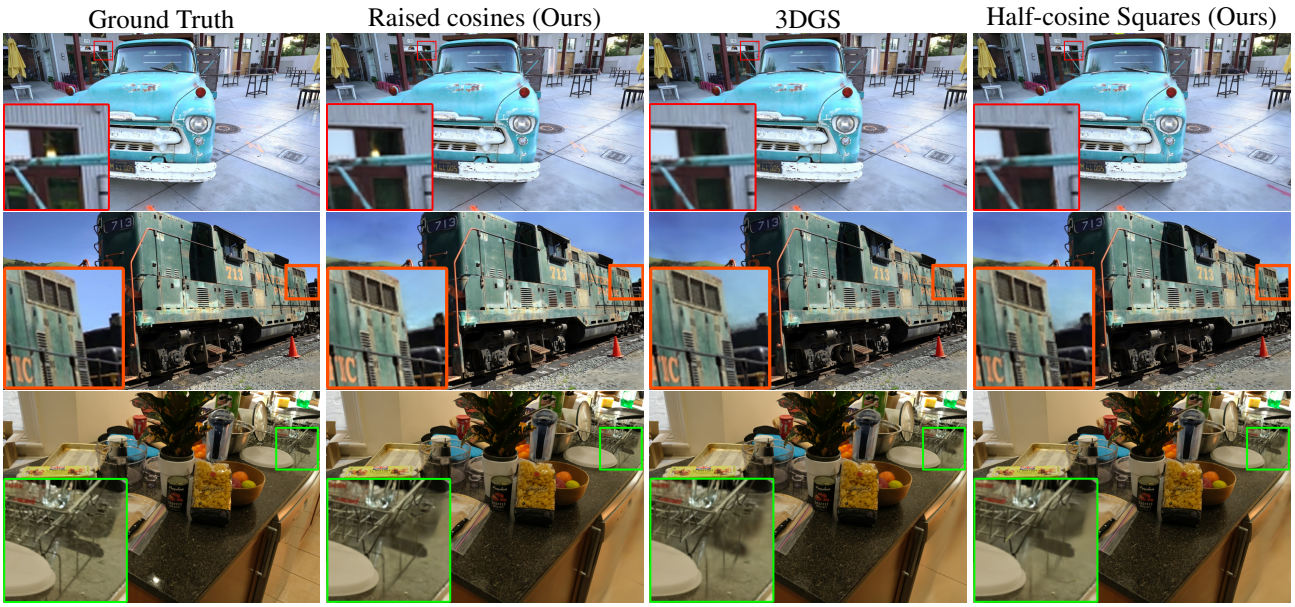


Figure 5. **Visual comparison on NVS.** We present comparisons between our proposed methods (Raised cosine and half-cosine squares) and established baseline (3DGS) alongside their respective ground truth images. The displayed scenes are ordered as follows: TRUCK and TRAIN from Tanks&Temples [22] dataset; COUNTER from Mip-NeRF360 [2] dataset. We highlight subtle improvements in our approach, such as the appearance of a light bulb in the rendered image of the first row, which is absent in the splitting algorithm. Additionally, the edges of the train and the shadows of the rack in the images on the second and third rows, respectively, are noticeably sharper, whereas these details appear blurred in the original splitting algorithm. Note that in addition to improved reconstruction, we also achieve significantly better training and memory efficiency with these alternative primitives (see Table 3).

$$\frac{dw}{d((\Sigma'_{2\times2})^{-1})} = \frac{-(x-\mu')\ (x-\boldsymbol{\mu'})^T\ \sin\left(\frac{(x-\boldsymbol{\mu'})^T(\Sigma'_{2\times2})^{-1}(x-\boldsymbol{\mu'})}{\xi}\right)}{\xi} \tag{8}$$

5. Experiments

Experimental Settings. We anchor our contributions on the recently *updated codebase* of 3DGS [20], adjusting the CUDA scripts to support a range of DARBFs. For a fair comparison, we use the same testing scenes as the 3DGS paper, including both bounded indoor and large outdoor environment scenes from various datasets [2, 17, 22]. We utilized the same COLMAP [44] initialization provided by the official dataset for all tests conducted to ensure fairness. Furthermore, we retained the original hyperparameters, adjusting only the opacity learning rate to 0.02 for improved results. All experiments and evaluations were conducted, and further verified on a single NVIDIA GeForce RTX 4090 GPU.

We compare our work with SOTA splatting-based methods that use exponential family reconstruction kernels, such as the *original* 3DGS, *updated codebase* of 3DGS, and GES [15] as benchmarks. We also evaluated the DARBFs using the standard and frequently used PSNR, LPIPS and SSIM metrics similar to these benchmarks (Table 3).

5.1. Results

Table 3 summarizes the comparative analysis across various datasets alongside our benchmarks. It shows that different functions perform better quantitatively in various aspects, such as training time, memory efficiency, and visual quality. Results show that the modified raised cosine function with $\xi = \frac{2.5}{\pi}$ achieves on-par results in terms of PSNR, LPIPS and SSIM metrics with the SOTA *original* and *updated codebase* of 3DGS [20], further validating our 1D simulation results discussed in Sec. 3.2. Additionally, it demonstrates that certain DARBFs, despite not belonging to the exponential family, are able to compete effectively with our previous related work benchmark [15].

Half-cosine squares with $\xi = \frac{18}{\pi}$, achieve approximately a 15% reduction in training time compared to Gaussians (*updated codebase*) on average across all scenes, with a notably modest trade-off in visual quality of less than 0.5 dB on average. This is because a half-cosine square spans a larger region compared to a Gaussian. With precomputed scaling factors for each DARBF, there is no additional computational overhead in the rendering pipeline, resulting in comparable rendering times to Gaussians. Fig. 5 shows that raised cosine captures fine details better than Gaussians, and Table 4 presents a comparative analysis of our method across various 3D reconstruction methods.

Function	Mip-NeRF360					Tanks & Temples					Deep Blending				
	SSIM↑	PSNR↑	LPIPS↓	Train	Memory	SSIM↑	PSNR↑	LPIPS↓	Train	Memory	SSIM↑	PSNR↑	LPIPS↓	Train	Memory
3DGS (7k) [20]	0.770	25.60	0.279	4m 43s*	523 MB	0.767	21.20	0.28	5m 05s*	270 MB	0.875	27.78	0.317	3m 22s*	368 MB
3DGS (30K) [20]	0.815	27.21	0.214	30m 33s*	734 MB	0.841	23.14	0.183	19m 46s*	411 MB	0.903	29.41	0.243	26m 29s*	676 MB
GES [15]	0.794	26.91	0.250	23m 31s*	377 MB	0.836	23.35	0.198	15m 26s*	222 MB	0.901	29.68	0.252	22m 44s*	399 MB
3DGS-updated (7K)	0.769	25.95	0.281	3m 24s	504 MB	0.781	21.78	0.261	2m 02s	293 MB	0.879	28.42	0.305	3m 14s	462 MB
3DGS-updated (30K)	0.813	27.45	0.218	19m 06s	633 MB	0.847	23.77	0.173	11m 24s	371 MB	0.901	29.66	0.242	20m 12s	742 MB
Raised cosine (7K)	0.773	26.01	0.273	3m 13s	513 MB	0.788	21.88	0.251	1m 59s	304 MB	0.882	28.47	0.300	2m 57s	401 MB
Raised cosine (30K)	0.813	27.45	0.214	19m 36s	645 MB	0.851	23.64	0.166	12m 06s	364 MB	0.901	29.63	0.240	20m 01s	767 MB
Half cosine (7K)	0.737	25.46	0.316	2m 52s	400 MB	0.749	21.071	0.295	1m 42s	244 MB	0.872	27.938	0.320	2m 42s	355 MB
Half cosine (30K)	0.790	27.04	0.247	16m 15s	524 MB	0.824	23.108	0.201	9m 11s	338 MB	0.900	29.38	0.253	17m 35s	634 MB
Modular Sinc (7K)	0.751	25.74	0.302	3m 20s	439 MB	0.767	21.59	0.276	2m 04s	255 MB	0.878	28.39	0.308	3m 17s	399 MB
Modular Sinc (30K)	0.801	27.30	0.234	18m 50s	622 MB	0.836	23.51	0.189	16m 08s	333 MB	0.901	29.66	0.247	20m 47s	682 MB

Table 3. **Splatting results:** The best values are highlighted in red and second-best values in orange, the third-best values in light orange, and the fourth-best in yellow. Note that, * denotes the difference in computation power that is used to implement our model and previous models (3DGS [20], GES [15]), hence we used a speed-up factor *(see Supplementary Materials)*, to get fair results regarding training time which is available in the original publication. Our DARBFs were implemented based on the *updated* 3DGS codebase. Our DARB splatting with raised cosine function was able to achieve on-par performance regarding PSNR, SSIM, LPIPS and training duration. Also, half cosine square shows a significant reduction in training time. We evaluate the performance of each model by considering the results at 7K and 30K iteration checkpoints.

Table 4. Comparison of various 3D reconstruction methods, showing that our approach outperforms prior methods in PSNR, LPIPS, and training efficiency metrics. Note that the benchmark values for 3DGS and GES are sourced directly from their respective papers, and therefore may not represent an entirely fair comparison.

Method	SSIM↑	PSNR↑	LPIPS↓	Train↓
Plenoxels [12]	0.626	23.08	0.463	19m*
INGP [35]	0.699	25.59	0.331	5.5m*
Mip-NeRF360 [2]	0.792	27.69	0.237	48h
3DGS [20]	0.815	27.21	0.214	30m*
GES [15]	0.794	26.91	0.250	23m*
DARBS (RC)	0.813	27.45	0.214	19min
DARBS (HC)	0.790	27.04	0.247	16min

5.2. Ablation Study

Decaying ARBFs. We validated various reconstruction kernels through low-dimensional simulations (Sec. 3.2), showing that normalized, decaying functions focus energy near the origin (Sec. 4.1), ensuring each sample's influence on the reconstruction is strongest its point, aiding accurate signal reconstruction. In contrast, the instability caused by unbounded growing functions led to their omission in favor of DARBFs.

Correction factor. The existing pipeline was built upon the prominent property of Gaussians, where integrating along a specific axis results in the same function, in a lower dimension. However, DARBFs generally do not exhibit this property. To incorporate DARBFs into the original pipeline, we introduced a correction factor ψ to approximate the projected covariance (Sec. 4.1). As shown in Table 5, this modification significantly enhanced the performance of DARBFs, achieving comparable results across all scenes in the Mip-NeRF 360 dataset on average for raised cosines, relative to the *updated* 3DGS codebase, implying outperformance of the same in the *original* 3DGS results.

Table 5. Comparison of PSNR, SSIM, and LPIPS metrics for Mip-NeRF 360 scenes under the influence of ψ.We claim that incorporating ψ not only enhances the performance of our kernels but also serves as the pivotal factor in surpassing the 3DGS results.

	with ψ			without ψ		
Scenes	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
3DGS-updated	—	—	—	27.45	0.81	0.22
Raised Cosine	27.45	0.81	0.21	26.69	0.78	0.25
Half Cosine	27.04	0.79	0.25	25.89	0.76	0.28

6. Conclusion and Discussion

To the best of our knowledge, we are the first to generalize splatting techniques with DARB-Splatting, extending beyond the conventional exponential family. In introducing this new class of functions, DARBFs, we highlight the distinct performances of each function. We establish the relationships between 3D covariance and 2D projected covariance through Monte Carlo experiments, providing an effective approach for understanding covariance transformations under projection. Our modified CUDA codes for each DARBF are available to facilitate further research and exploration in this area.

Limitation. While we push the boundaries of splatting functions with DARBFs, the useful properties and capabilities of each function for 3D reconstruction, within a classified mathematical framework, remain to be fully explored. Future research could explore utility applications and the incorporation of existing signal reconstruction algorithms, such as the Gram-Schmidt process, with DARBFs.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5835–5844, 2021. 2

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, pages 5470–5479, 2022. 7, 8

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *CVPR*, pages 19697–19705, 2023. 2

[4] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Springer, 2006. 5

[5] G. Casciola, D. Lazzaro, L.B. Montefusco, and S. Morigi. Shape preserving surface reconstruction using locally anisotropic radial basis function interpolants. *Computers & Mathematics with Applications*, 51(8):1185–1198, 2006. Radial Basis Functions and Related Multivariate Meshfree Approximation Methods: Theory and Applications. 3

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, pages 333–350. Springer, 2022. 2

[7] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *CVPR*, pages 21476–21485, 2024. 2

[8] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3D gaussian splatting from sparse multi-view images. In *ECCV*, pages 370–386, 2025. 2

[9] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes, 2023. 2

[10] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, pages 12882–12891, 2022. 2

[11] Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3d gaussian splatting as new era: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2

[12] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5501–5510, 2022. 8

[13] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, pages 14346–14355, 2021. 2

[14] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, page 43–54, New York, NY, USA, 1996. Association for Computing Machinery. 2

[15] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. Ges: Generalized exponential splatting for efficient radiance field rendering. In *CVPR*, pages 19812–19822, 2024. 2, 3, 7, 8

[16] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *CVPR*, pages 19740–19750, 2023. 2

[17] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 37(6):257:1–257:15, 2018. 7

[18] Tongyan Hua and Lin Wang. Benchmarking implicit neural representation and geometric rendering in real-time rgb-d slam. In *CVPR*, pages 21346–21356, 2024. 2

[19] Jiajun Huang, Hongchuan Yu, Jianjun Zhang, and Hammadi Nait-Charif. Point'n move: Interactive scene object manipulation on gaussian splatting radiance fields. *IET Image Processing*, 2024. 2

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4), 2023. 1, 2, 4, 5, 6, 7, 8

[21] Muhammad Osama Khan and Yi Fang. Implicit neural representations for medical imaging segmentation. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022*, pages 433–443, 2022. 2

[22] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 7

[23] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. In *ECCV*, pages 252–269. Springer, 2025. 2

[24] Christoph Lassner and Michael Zollhöfer. Pulsar: Efficient sphere-based neural rendering. In *CVPR*, 2021. 5

[25] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *CVPR*, pages 21719–21728, 2024. 2

[26] Jiahui Lei, Yufu Wang, Georgios Pavlakos, Lingjie Liu, and Kostas Daniilidis. Gart: Gaussian articulated template models. In *CVPR*, pages 19876–19887, 2024. 2

[27] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, page 31–42, New York, NY, USA, 1996. Association for Computing Machinery. 2

[28] Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 3d-hgs: 3d half-gaussian splatting. *arXiv preprint arXiv:2406.02720*, 2024. 2, 3

[29] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *CVPR*, pages 14556–14565, 2021. 2

[30] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. pages 15651–15663, 2020. 2

[31] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *CVPR*, pages 5773–5783, 2021. 2

[32] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, pages 7210–7219, 2021. 2

[33] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[34] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *CVPR*, pages 16190–16199, 2022. 2

[35] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):1–15, 2022. 2, 8

[36] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *CVPR*, pages 5865–5874, 2021. 2

[37] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

[38] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, pages 10318–10327, 2021. 2

[39] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *CVPR*, pages 14153–14161, 2021. 2

[40] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *CVPR*, pages 14335–14345, 2021. 2

[41] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting, 2024. 2

[42] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, pages 12892–12901, 2022. 2

[43] Hemanth Saratchandran, Sameera Ramasinghe, Violetta Shevchenko, Alexander Long, and Simon Lucey. A sampling theory perspective on activations for implicit neural representations, 2024. 2

[44] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. 2, 3, 4, 7

[45] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, pages 20154–20166, 2020. 2

[46] Ruizhi Shao, Jingxiang Sun, Cheng Peng, Zerong Zheng, Boyao Zhou, Hongwen Zhang, and Yebin Liu. Control4d: Dynamic portrait editing by learning 4d gan from 2d diffusion-based editor. *arXiv preprint arXiv:2305.20082*, 2 (6):16, 2023. 2

[47] Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. Language embedded 3d gaussians for open-vocabulary scene understanding, 2023. 2

[48] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001. 2

[49] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006. 2, 3

[50] Kun Su, Mingfei Chen, and Eli Shlizerman. Inras: Implicit neural representation for audio scenes. In *NeurIPS*, pages 8144–8158, 2022. 2

[51] Jingxiang Sun, Xuan Wang, Yong Zhang, Xiaoyu Li, Qi Zhang, Yebin Liu, and Jue Wang. Fenerf: Face editing in neural radiance fields. In *CVPR*, pages 7672–7682, 2022. 2

[52] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *CVPR*, pages 20675–20685, 2024. 2

[53] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *CVPR*, pages 8248–8258, 2022. 2

[54] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation, 2024. 2

[55] G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992. 2

[56] Lee Westover. Interactive volume rendering. In *Proceedings of the 1989 Chapel Hill Workshop on Volume Visualization*, page 9–16, New York, NY, USA, 1989. Association for Computing Machinery. 2, 3

[57] Lee Westover. Footprint evaluation for volume rendering. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, page 367–376, New York, NY, USA, 1990. Association for Computing Machinery. 3

[58] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *CVPR*, pages 20310–20320, 2024. 2

[59] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *CVPR*, pages 4389–4398, 2024. 2

[60] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, pages 5438–5448, 2022. 2

[61] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *CVPR*, pages 19595–19604, 2024. 2

[62] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *CVPR*, pages 13779–13788, 2021. 2

[63] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *ECCV*, 2024. 2

[64] Vickie Ye and Angjoo Kanazawa. Mathematical supplement for the `gsplat` library, 2023. 5, 6

[65] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2D and 3D diffusion models. In *CVPR*, pages 6796–6807, 2024. 2

[66] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *CVPR*, pages 5752–5761, 2021. 2

[67] Heng Yu, Joel Julin, Zoltán A. Milacski, Koichiro Niinuma, and László A. Jeni. Cogs: Controllable gaussian splatting. In *CVPR*, pages 21624–21633, 2024. 2

[68] Ruihan Yu, Tianyu Huang, Jingwang Ling, and Feng Xu. 2dgh: 2d gaussian-hermite splatting for high-quality rendering and better geometry reconstruction. *arXiv preprint arXiv:2408.16982*, 2024. 2, 3

[69] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3D Gaussian splatting. In *CVPR*, pages 19447–19456, 2024. 2

[70] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *CVPR*, pages 19447–19456, 2024. 2

[71] Fuqiang Zhao, Wei Yang, Jiakai Zhang, Pei Lin, Yingliang Zhang, Jingyi Yu, and Lan Xu. Humannerf: Efficiently generated human radiance field from sparse inputs. In *CVPR*, pages 7743–7753, 2022. 2

[72] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 2, 3, 5