Deformable Radial Kernel Splatting

Yi-Hua Huang[1] **Ming-Xian Lin**[1] **Yang-Tian Sun**[1] **Ziyi Yang**[1] **Xiaoyang Lyu**[1]
Yan-Pei Cao[2†] **Xiaojuan Qi**[1†]
[1] The University of Hong Kong [2] VAST

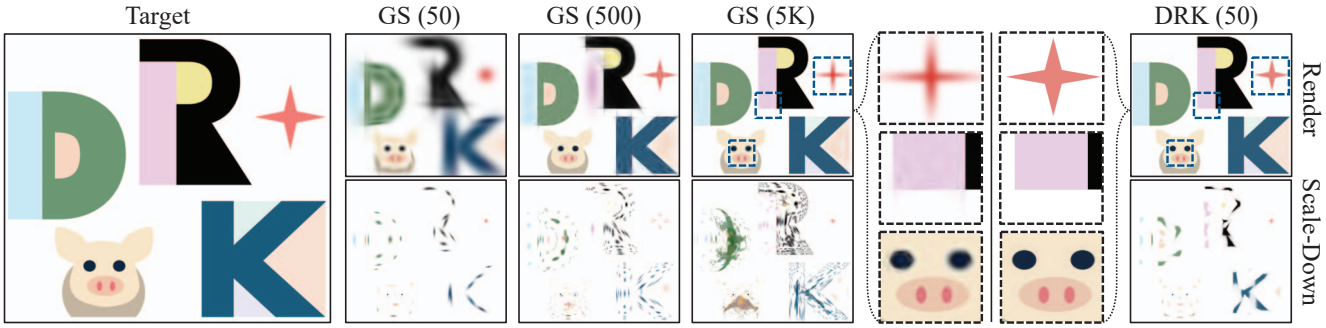

Figure 1. **Gaussian Splatting (GS) vs. Deformable Radial Kernel (DRK) Splatting**: GS requires thousands of Gaussians to approximate detailed textures and shapes. In contrast, DRK efficiently fits the target pattern with just 30 primitives, achieving superior results.

Abstract

Recently, Gaussian splatting has emerged as a robust technique for representing 3D scenes, enabling real-time rasterization and high-fidelity rendering. However, Gaussians' inherent radial symmetry and smoothness constraints limit their ability to represent complex shapes, often requiring thousands of primitives to approximate detailed geometry. We introduce Deformable Radial Kernel (DRK), which extends Gaussian splatting into a more general and flexible framework. Through learnable radial bases with adjustable angles and scales, DRK efficiently models diverse shape primitives while enabling precise control over edge sharpness and boundary curvature. iven DRK's planar nature, we further develop accurate ray-primitive intersection computation for depth sorting and introduce efficient kernel culling strategies for improved rasterization efficiency. Extensive experiments demonstrate that DRK outperforms existing methods in both representation efficiency and rendering quality, achieving state-of-the-art performance while dramatically reducing primitive count.

1. Introduction

3D Gaussian Splatting [27] (3D-GS) has emerged as a leading 3D representation method thanks to its rapid rasterization and superior rendering quality. Its key advantages– explicit point-based representation, flexibility for manipulation, and MLP-free rendering– have positioned it as a dominant approach in novel view synthesis. However, this success raises an intriguing question: *Is the Gaussian kernel truly optimal for representing 3D scenes?*

Natural scenes often consist of primitives with diverse shapes (see Fig. 2), such as rectangles, triangles, and ellipses, which cannot be fully captured by Gaussians alone. Moreover, the inherently smooth nature of Gaussian kernels makes them less effective at representing sharp transitions. As a result, densely packed or numerous Gaussians are frequently required to accurately model scene primitives with non-Gaussian geometries or sharp boundaries (see Fig. 1), leading to inefficiencies in both computation and memory.

Recent works have explored modifications to the Gaussian kernel to better handle discontinuities. For example, GES [20] adaptively adjusts the exponent values to control the sharpness of Gaussians, while still maintaining rotational symmetry, which limits its representational flexibility. DisC-GS [43] and 3D-HGS [30] introduce curve- and half-cutting techniques to the Gaussian kernel, respectively, improving its ability to capture discontinuous transitions. However, these cutting-based approaches remain fundamentally constrained by the inherent smoothness of the Gaussian kernel. Another line of research [8, 16, 19, 22] leverages 2D Gaussians to represent 3D scenes. While this approach offers better surface reconstruction compared to 3DGS, it is still constrained by the inherent limitations of the Gaussian framework, which restricts the flexibility

†Corresponding Author
Project page: https://yihua7.github.io/DRK-web/.

of the representation. Notably, the concurrent work 3D-CS [21] employs novel 3D smooth convex shapes as primitives for geometry modeling with satisfactory results.

In this work, our objective is to design new shape primitives that can adapt to complex 3D scenes. Motivated by the success of 2D Gaussian Splatting (2D-GS), we focus on developing adaptive 2D planar kernels. Specifically, we introduce the 2D Deformable Radial Kernel (*DRK*), a flexible planar primitive for representing 3D scenes (see Fig. 4). At its core, DRK extends traditional radial bases with learnable parameters that enable precise control over shape deformation through three key features: 1) multiple radial bases with learnable polar angles and scale parameters allow the kernel to adaptively stretch, rotate, and shrink, providing significantly more flexibility than fixed Gaussian functions; 2) a hybrid distance metric combining $L1$ and $L2$ norms with learnable weights enables DRK to naturally represent both curved surfaces and sharp geometric features; and 3) an adaptive piecewise linear remapping function provides fine-grained control over value distributions, particularly enhancing the representation of sharp boundaries. Despite its enhanced expressiveness, DRK maintains a simple parametric form that is easy to optimize and efficient to render. The DRK determines the location and orientation of the tangent plane through its center and rotation parameters, building upon these radial bases as its structural foundation to conform to target shapes, bypassing the geometric and smoothness constraints typically imposed by Gaussian functions. As shown in Fig. 2, a single DRK primitive can flexibly model a wide range of shape primitives, whereas many Gaussians are often required to represent similar shapes. The formulation is also backwards-compatible – DRK naturally reduces to a Gaussian kernel when using two orthogonal major bases with different decay rates, while achieving superior rendering quality with comparable computational efficiency.

We experimentally validate *DRK* on our proposed dataset (DiverseScene) covering various scenarios, including rich textures, fine geometry, specular effects, and large spatial scale. We also evaluate *DRK* on unbounded Mip-NeRF 360 [2] datasets. Experimental results demonstrate our method has advantages in preserving visual details more effectively and efficiently.

Overall, our work makes the following contributions:

- We introduce Deformable Radial Kernel (*DRK*), a novel primitive that generalizes conventional Gaussian splatting. DRK enables flexible shape modeling through learnable radial bases, hybrid distance metrics, and adaptive sharpness control, significantly reducing the number of primitives needed for high-quality scene representation.
- We develop a differentiable and efficient rendering pipeline for DRK, featuring polygon-based tile culling and cache-sorting strategies to enhance both rendering efficiency and view consistency.
- We created DiverseScenes, a new dataset covering rich textures, fine geometry, specular effects, and large scenes for more effectively evaluating the ability to model complex scenarios of different algorithms.

2. Related Work

2.1. Novel View Synthesis

Novel view synthesis from multi-view images represents a fundamental challenge in computer graphics and computer vision. The field has evolved through increasingly sophisticated 3D scene representations. Traditional light field approaches [10, 18, 29] attempt to model the entire scene through a single 4D function of ray-slice intersections, but this implicit representation struggles with sparse inputs and complex geometry. Advancing beyond single-function modeling, Multi-Plane-Images (MPI) [38, 51, 55, 58, 69] introduced a layered representation using multiple depth planes, offering better 3D structure modeling. However, its discrete depth layering limits the handling of large viewpoint changes, particularly in rotational views. To achieve continuous depth modeling, subsequent works adopted explicit 3D proxies including meshes [5, 11, 35, 56, 59], point clouds [1, 7, 37, 41, 57], and voxels [14, 32, 48, 49]. These representations offer more flexible geometry modeling, with point-based methods further enhanced by neural networks [45, 46] for improved stability.

Most recently, continuous neural representations have emerged as a powerful alternative, replacing discrete geometric proxies with MLPs that directly map 3D coordinates to scene properties. NeRF [39] pioneered this approach with radiance fields, spawning numerous extensions for faster inference [15, 26, 40], large-scale scenes [2, 17, 53, 61, 66], dynamic content [31, 42, 50], reflection modeling [3, 24, 28, 62], and stylization [9, 13, 23, 67].

2.2. Gaussian Splatting

Point-based rendering approaches have made the rendering process of point clouds differentiable, enabling end-to-end training for novel view synthesis. 3D Gaussian Splatting (3D-GS) [27] introduced scene modeling using 3D Gaussian distributions, compositing the appearance of intersected Gaussians along viewing rays through α-blending. This seminal work has inspired numerous extensions across various applications, including dynamic scene view synthesis [25, 34, 65], 3D content generation [54, 70], geometry reconstruction [22, 36], video representation [52] and neural network-enhanced high-fidelity view synthesis [33, 64]. The flexible representation, computational efficiency, and superior rendering quality of 3D-GS have established it as a prominent approach in the field [60].

The key innovation of 3D-GS lies in its use of Gaussian

kernels for point effect decay, where the continuous nature of these functions facilitates the optimization of discrete representations. Subsequent research has focused on both constraining and extending these Gaussian kernels for specific applications. Notable developments include the constraint of 3D Gaussians to 2D space [8, 19, 22], yielding planar representations that better approximate scene surfaces for precise geometry reconstruction. Other works have extended the framework to 4D space [12, 63], enabling temporal dynamic modeling through 3D slicing of 4D Gaussians for view synthesis. To address limitations in representing sharp edges, researchers have proposed modifications to the Gaussian shape using curve-cutting [43], hemispheric truncation [30], and novel 3D convex shapes [21].

3. Preliminaries

In this section, we first review 3D Gaussian Splatting (3D-GS) and 2D Gaussian Splatting (2D-GS), and their mathematical formulations (Sec. 3.1 and Sec. 3.2). We then introduce our general kernel splatting framework that extends beyond traditional Gaussian primitives (Sec. 3.3). Finally, we analyze the inherent limitations of Gaussian-based representations, motivating the need for our proposed deformable radial kernel formulation (Sec. 3.4).

3.1. 3D Gaussian Splatting (3D-GS)

3D Gaussian Splatting represents scenes using colored 3D Gaussians [27]. Each Gaussian G is parameterized by a 3D center μ, covariance matrix $\Sigma = RSS^TR^T$ (where R is a rotation matrix from quaternion $q \in \mathbf{SO}(3)$ and S is a scaling matrix from vector s), opacity o, and spherical harmonic coefficients sh for view-dependent appearance. A scene is represented as $\mathcal{G} = \{G_j : \mu_j, q_j, s_j, o_j, sh_j\}$.

Rendering involves projecting Gaussians onto the image plane with 2D covariance $\Sigma' = JW\Sigma W^TJ^T$ and center $\mu' = JW\mu$. The pixel color $C(u)$ is computed via neural point-based α-blending:

$$C(u) = \sum_{i \in N} T_i \alpha_i \mathcal{SH}(sh_i, v_i), \text{ where } T_i = \Pi_{j=1}^{i-1}(1-\alpha_j), \quad (1)$$

where $\mathcal{SH}$ evaluates spherical harmonics with view direction v_i, and α_i is:

$$\alpha_i = o_i e^{-\frac{1}{2}(p-\mu_i')^T \Sigma_i' (p-\mu_i')}. \quad (2)$$

By optimizing these Gaussian parameters and adaptively adjusting their density, 3D-GS achieves high-quality scene representation with real-time rendering capabilities.

3.2. 2D Gaussian Splatting (2D-GS)

Recognizing that real-world scenes primarily consist of surface structures where 3D Gaussians naturally compress into planar formations, recent works [8, 22] proposed 2D Gaussian Splatting (2D-GS). The opacity α is computed using local coordinates (u, v) on the tangent plane:

$$\alpha = o \cdot \exp\left(-\frac{1}{2}\left(\frac{u^2}{s_u^2} + \frac{v^2}{s_v^2}\right)\right), \quad (3)$$

where s_u^2 and s_v^2 denote the variances along the U and V axes. This surface-centric formulation aligns with the success of polygon meshes in 3D modeling, offering a more compact scene representation.

3.3. General Kernel Splatting

We extend conventional 2D Gaussians by introducing a more general planar kernel splatting formulation (Fig. 3). Given a ray from position $r_o \in \mathbb{R}^3$ with direction $r_d \in \mathbb{R}^3$ intersecting a tangent plane centered at μ with rotation $R \in SO(3)$, the intersection point is:

$$i = r_o + \frac{(\mu - r_o)^T R_z}{r_d^T R_z} r_d, \quad (4)$$

where R_z is the plane normal. The local UV coordinates at this intersection are:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} R_x^T \\ R_y^T \end{pmatrix} (i - \mu), \quad (5)$$

where R_x^T and R_y^T are the first and second rows of R. The kernel splatting function then maps these UV coordinates to a density value α, determining the primitive's shape.

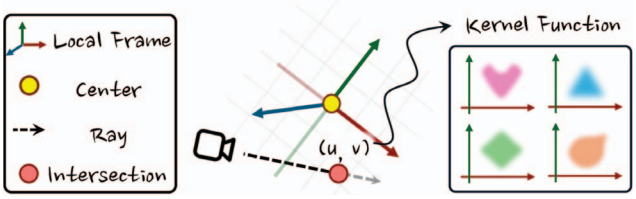

Figure 3. Illustration of general planar kernel splatting: UV coordinates of ray-plane intersections are mapped to density values via a kernel function that determines the primitive's shape.

3.4. Discussions

While 3D-GS demonstrates remarkable efficacy, our analysis reveals several inherent limitations. *First*, screen-space projection of Gaussians yields 2D elliptical distributions with **rotational symmetry**, constraining their ability to approximate diverse primitive shapes. *Second*, their **conic curve** boundaries (derived by the $L2$ norm) challenge the representation of linear edges and intermediate edge forms. *Third*, Gaussian distributions inherently couple **decay rate** with spatial extent through the covariance matrix—sharp features necessitate narrow distributions, making it challenging to simultaneously capture abrupt transitions and extended spatial regions.

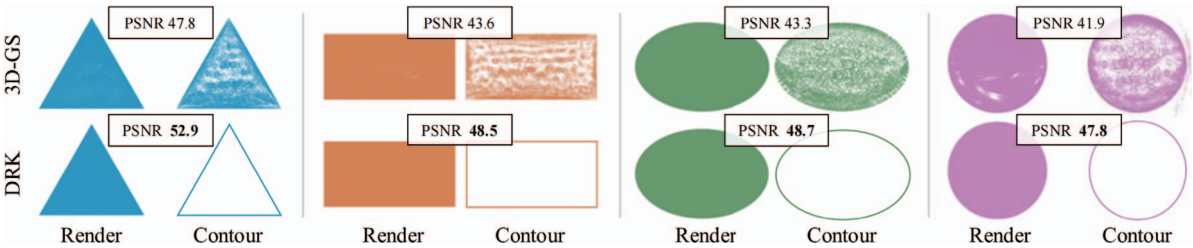

Figure 2. Comparison of 3D-GS versus a single *DRK*: *DRK* achieves superior geometric fidelity with just one primitive compared to multiple Gaussians. We visualize the contours of 3D-GS and *DRK* to better illustrate primitive count, scale, and position.

These constraints necessitate millions of fine-grained Gaussians to approximate arbitrary shapes, leading to over-parameterization while still failing to achieve perfect fidelity under practical constraints. As shown in Fig. 2, even elementary shapes like triangles and rectangles—which can be described concisely—require numerous Gaussians for approximation, exhibiting artifacts like interior inconsistencies and external floating points (last column). Our deformable radial kernel (DRK) addresses these limitations through radial basis functions with varying lengths s_k and polar angles θ_k, controlled by parameters η and τ for contour curvature and sharpness (Fig. 4).

4. Deformable Radial Kernel Splatting

Deformable Radial Kernel (DRK) is a novel primitive for 3D scene representation characterized by a set of parameters $\Theta = \{\mu, q, s_k, \theta_k, \eta, \tau, o, sh\}$, where:

- $\mu \in \mathbb{R}^3$, $q \in SO(3)$, o, and sh follow 3D-GS for position, orientation, opacity, and view-dependent appearance
- $\{s_k, \theta_k\}_{k=1}^K$ define radial basis lengths and angles that control the kernel shape
- $\eta \in (0, 1)$ and $\tau \in (-1, 1)$ control boundary curvature and sharpness, respectively

In the following sections, we detail each component of DRK. First, we introduce the radial basis formulation (Sec. 4.1), showing that traditional 2D Gaussians are a special case within our framework. Then, we discuss $L1\&L2$ norm blending and edge sharpening in Sec. 4.2 and Sec. 4.3, which enable flexible control of boundary curvature and sharpness. Finally, in Sec. 4.4, we explore efficient rasterization strategies specifically tailored for DRK, including low-pass filtering, kernel culling, and cache sorting.

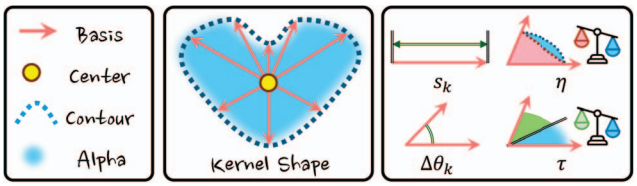

Figure 4. *DRK* defines a deformable shape using radial basis functions characterized by lengths s_k and polar angles θ_k, with parameters η and τ governing the shape's curvature and sharpness respectively.

4.1. Radial Basis

Our radial basis defines a kernel's shape through K control points in polar coordinates $\{s_k, \theta_k\}$, where $s_k > 0$ represents each radial length and angles are ordered as $0 < \theta_0 < \ldots < \theta_K \leq 2\pi$. For any point with local tangent coordinates (u, v), we compute its polar representation using $L2$ norm $r_2 = \sqrt{u^2 + v^2}$ and angle $\theta = \arccos(u/r_2)$. Given a point lying between basis k and $k+1$ (i.e., $\theta_k < \theta \leq \theta_{k+1}$), we define its relative angular distance as $\Delta\theta_k = \frac{(\theta - \theta_k)\pi}{\theta_{k+1} - \theta_k}$. The kernel function is then formulated as:

$$\alpha = o \cdot \exp\left(-\frac{r_2^2}{2}\left(\frac{1 + \cos(\Delta\theta_k)}{2s_k^2} + \frac{1 - \cos(\Delta\theta_k)}{2s_{k+1}^2}\right)\right). \quad (6)$$

This formulation smoothly interpolates between adjacent radial bases using cosine weighting terms, ensuring continuous transitions in the kernel's shape.

2D Gaussian as a Special Case. Our formulation generalizes 2D Gaussian kernels. By setting $K = 4$ with $\theta_k = \frac{k\pi}{2}$ and $s_0 = s_2 = s_u$, $s_1 = s_3 = s_v$, we can derive the 2D Gaussian form in Eq. (3). For points in the first UV quadrant where $\Delta\theta_k = 2\theta$:

$$\begin{aligned}\alpha &= o \cdot \exp\left(-\frac{r_2^2}{2}\left(\frac{\cos^2\theta}{s_0^2} + \frac{\cos^2\theta}{s_1^2}\right)\right) \\ &= o \cdot \exp\left(-\frac{1}{2}\left(\frac{u^2}{s_0^2} + \frac{v^2}{s_1^2}\right)\right),\end{aligned}$$

where we use the identity $\cos(\Delta\theta_k) = \cos(2\theta) = 2\cos^2\theta - 1$. The derivation extends to other quadrants by symmetry.

4.2. $L1\&L2$ Norm Blending

The scaling term $\frac{1}{\bar{s}^2} = \left(\frac{1+\cos(\Delta\theta_k)}{2s_k^2} + \frac{1-\cos(\Delta\theta_k)}{2s_{k+1}^2}\right)$ in Eq. (6) enforces smooth contours at radial endpoints, where $\frac{d\bar{s}}{d\Delta\theta_k} = 0$ at $\Delta\theta_k = 0$ or π. However, since radials converge at the kernel center and are not parallel, this $L2$ formulation inherently produces conic curves, limiting its ability to represent **straight edges** common in man-made environments.

To address this limitation, we incorporate $L1$ norm into our *DRK* formulation. For a point (u, v) between radial

endpoints $e_i = (s_i \cos(\theta_i), s_i \sin(\theta_i))$ and e_{i+1}, we compute its $L1$ norm as:

$$r_1 = \left\| \begin{pmatrix} e_i & e_{i+1} \end{pmatrix}^{-1} \begin{pmatrix} u \\ v \end{pmatrix} \right\|_1 . \tag{7}$$

This transformation maps the diamond-shaped $L1$ unit ball ($|x| + |y| = 1, 0 \leq x, y \leq 1$) to the segment between adjacent endpoints, enabling straight-line boundaries.

We then introduce a blending weight $\eta \in (0, 1)$ to smoothly interpolate between $L1$ and $L2$ norms, yielding our complete kernel function:

$$\alpha = o \cdot \exp\left(-\frac{1}{2}\left(\eta r_1^2 + (1-\eta)\frac{r_2^2}{\bar{s}^2}\right)\right). \tag{8}$$

Note that r_1^2 requires no explicit scaling term as it is inherently scaled through the inverse transformation in Eq. (7). Please see Fig. 4 (right) for illustration.

4.3. Edge Sharpening

As shown in Fig. 2, Gaussian functions inherently couple scale and edge sharpness through their variance parameter, making it challenging to achieve precise shape approximation even with thousands of Gaussians. To decouple these properties, we introduce a sharpening coefficient $\tau \in (-1, 1)$ that adaptively adjusts edge transitions. Given the exponential term g (i.e., the density value before opacity scaling) in Eq. (8), our sharpening function is:

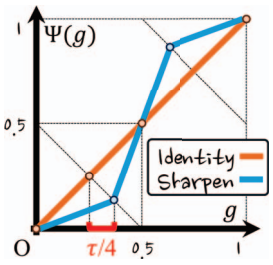

Figure 5. Sharpening function illustration.

$$\Psi(g) = \begin{cases} \frac{1-\tau}{1+\tau} g & \text{if} \quad 0 \leq g < \frac{1+\tau}{4}, \\ \frac{1+\tau}{1-\tau} g - \frac{\tau}{1-\tau} & \text{if} \quad \frac{1+\tau}{4} \leq g < \frac{3-\tau}{4}, \\ \frac{1-\tau}{1+\tau} g + \frac{2\tau}{1+\tau} & \text{if} \quad \frac{3-\tau}{4} \leq g \leq 1. \end{cases} \tag{9}$$

As illustrated in Fig. 5, this piecewise function reshapes the alpha values toward 0 or 1, creating sharper edge transitions while maintaining the kernel's spatial extent. The final kernel opacity is computed as $\alpha = o \cdot \Psi(g)$.

4.4. Rasterization

Low-pass Filtering. Camera-captured images represent discretely sampled signals from scene rays, inherently limiting the maximum recoverable frequency of 3D content. Without proper frequency constraints, optimization tends to generate small, floating primitives that overfit to individual training views. These primitives become too fine-grained (smaller than ray coverage) and only contribute to specific rays, resulting in poor generalization.

While 3D-GS leverages EWA splatting for closed-form Gaussian low-pass filtering, *DRK* requires an alternative approach. Following Botsch et al. [4], we approximate filtering by taking the maximum between the kernel function and a view-dependent low-pass filter. Since *DRK* represents planar primitives, its frequency should be bounded on its tangent plane while potentially unbounded when viewed orthogonally, so we scale the filter size by the cosine of the view angle:

$$\tilde{\alpha} = \max(\alpha, o \cdot \exp(-\frac{1}{2|r_d \cdot R_z|^2}(\frac{\Delta p_w^2}{s_l^2} + \frac{\Delta p_h^2}{s_l^2}))), \tag{10}$$

where Δp_h and Δp_w are image space distances between the pixel and the projected kernel center, $r_d \cdot R_z$ accounts for view-dependent scaling, and s_l controls the filter radius.

Tile Culling. Our rendering pipeline, based on 3D-GS, performs image rendering using 2D thread grids on CUDA. The process begins by dividing images into 16×16 tiles, with each tile corresponding to potentially intersecting primitives (kernels or 3D Gaussians). While 3D-GS projects the major axis of a 3D Gaussian onto the image plane to create a square axis-aligned bounding box (AABB) for tile coverage, this method can be computationally inefficient for primitives that project as highly anisotropic or elongated shapes, as shown in Fig. 6. This inefficiency becomes particularly evident in *DRK* due to its diverse shape representations. To address this, we introduce a more precise radius calculation and an improved tile culling strategy.

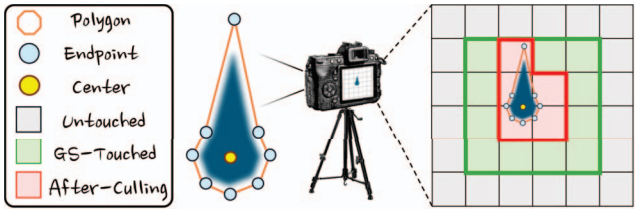

Figure 6. Polygon-based tile culling using radial basis endpoints enables efficient rasterization by eliminating untouched tiles.

The effective range of *DRK* is governed by its radial basis, whose endpoints define the kernel's boundary. Following the 3-σ principle, we define the boundary radius s_k^c, but extend it to account for opacity o and sharpness τ parameters. The calibrated radial length s_k^c is defined as:

$$s_k^c = s_k \sqrt{-\log\left(\Psi^{-1}(\frac{e^{-3^2}}{o})\right)}. \tag{11}$$

The derivation details are provided in the Supplementary Material. Using this calibrated length, we calculate each radial basis endpoint v_k:

$$v_k = \mu + s_k^c(\cos\theta_k R_x + \sin\theta_k R_y). \tag{12}$$

Methods	Simple			Texture			Geometry			Specular			Large			Average		
	PSNR	*LPIPS*	*SSIM*	*PSNR*	*LPIPS*	*SSIM*	*PSNR*	*LPIPS*	*SSIM*	*PSNR*	*LPIPS*	*SSIM*	*PSNR*	*LPIPS*	*SSIM*	*PSNR*	*LPIPS*	*SSIM*
2D-GS	42.10	.0209	.9969	42.59	.0624	.9834	29.90	.0513	.9700	27.18	.0570	.9550	27.80	.2500	.8388	33.92	.0881	.9514
3D-GS	39.51	.0311	.9941	44.57	.0628	.9968	29.14	.0552	.9658	27.16	.0758	.9523	32.19	.2054	.9017	34.41	.0861	.9621
3D-HGS	41.41	.0199	.9931	46.19	.0397	.9912	30.42	.0406	.9598	27.27	.0512	.9447	33.17	.1672	.8721	35.68	.0637	.9521
GES	40.95	.0266	.9955	45.23	.0579	.9977	30.36	.0468	.9704	27.04	.0632	.9550	31.67	.2074	.8987	35.05	.0804	.9634
DRK (S2)	40.34	.0283	.9946	47.67	.0537	.9976	30.68	.0477	.9709	27.07	.0704	.9570	31.57	.2090	.8985	35.03	.0823	.9637
DRK (S1)	42.24	.0168	.9974	47.86	.0410	.9988	31.67	.0394	.9788	27.77	.0631	.9603	33.78	.1685	.9196	36.62	.0668	.9701
DRK	43.46	.0105	.9985	49.08	.0396	.9990	31.93	.0366	.9803	28.19	.0592	.9615	35.28	.1348	.9372	37.58	.0564	.9752

Table 1. Rendering quality evaluation on DiverseScene datasets across various categories

	2DGS	3DGS	3DHGS	GES	*DRK*(S2)	*DRK*(S1)	*DRK*
Num(↓)	359K	336K	373K	330K	42 K	109K	260K
MB(↓)	83.6	79.7	89.6	78.1	12.3	32.1	76.6
FPS(↑)	251.3	247.1	154.5	227.4	234.9	119.2	77.5

Table 2. Evaluation results of average primitive number (Num), model size (MB), and rendering speed (FPS) on DiverseScene.

These endpoints form a polygon that accurately approximates the kernel's boundary. By projecting this polygon onto the image plane, we can efficiently identify and cull tiles that don't intersect with the kernel, making the rasterization more efficient.

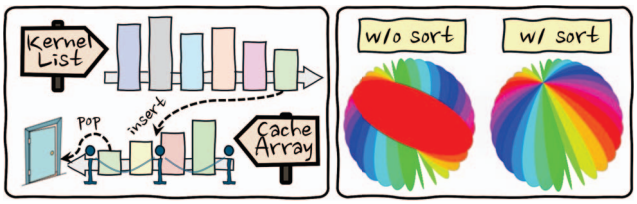

Figure 7. Our cache-sorting uses r_t as keys in cache-sorting to rectify the sorting order while preventing popping artifacts.

Cache Sorting. 3D-GS sorts the Gaussian using the depth of the center μ, which can lead to inconsistencies across multiple views. While Radl et al. [44] address this by estimating the depth of the Gaussian at the point of maximum density and applying a hierarchical sorting solution to mitigate popping artifacts caused by inconsistent sorting, we propose a simpler yet effective solution. In our approach, the distance r_t from the camera to the intersection between the ray and *DRK* is calculated as:

$$r_t = \frac{(\mu - r_o)^T R_z}{r_d^T R_z}. \tag{13}$$

During rendering, each thread processes a pixel using an sorted array to cache the r_t values and kernel indices, as illustrated in Fig. 7. The array is kept sorted by performing insert sorting for each new-coming kernel. If the array is full, the kernel with the smallest r_t is popped out for processing. A dynamic cursor is used to record the scanning. We maintain an array length of 8, which is sufficient for approximating a more accurate sort. Fig. 7 demonstrates our method's effectiveness in handling multiple overlapping kernels with identical center depths, comparing rasterization results with and without cache-sorting, preventing popping artifacts while maintaining rendering efficiency.

5. Experiment

5.1. Datasets and Evaluation Metrics

To evaluate our method's performance across diverse scenarios, we focused on four key aspects: texture complexity, geometric detail, view-dependent effects, and scene scale. We collected 10 representative 3D scenes from Sketchfab[1], with two scenes per category: simple objects, richly textured surfaces, intricate geometric details, specular materials, and large-scale environments. The dataset, namely *DiverseScenes*, serves as a benchmark to assess our representation method's capabilities. Detailed information about DiverseScenes can be found in the supplementary materials. For the evaluation on unbounded real-world scenes, we utilized the Mip-NeRF360 dataset [2], which contains multi-view images captured in open environments. Camera poses were estimated using COLMAP [47]. We assessed performance using three metrics: Peak Signal-to-Noise Ratio (PSNR) for overall reconstruction accuracy, Structural Similarity Index (SSIM) for perceptual quality and Learned Perceptual Image Patch Similarity (LPIPS) [68] for human-perceived visual fidelity. We observed that the number of primitives significantly impacts rendering quality. To evaluate DRK's performance across different densities, we present results using various hyper-parameters (details provided in the Supp), where the density decreases progressively from *DRK* to *DRK*(S2) and then to *DRK*(S1).

5.2. Quantitative Comparisons

DiverseScenes. We compare our method, *DRK*, against state-of-the-art Gaussian representations: 3D-GS [27], 2D-GS [22], 3D-HGS [30], and GES [20], using their official implementations. To ensure a fair comparison focused on kernel representation, we adjust the gradient threshold for densification in these methods, aligning their primitive count with that of 3D-GS (300K+). The results for each dataset category are shown in Table 1, demonstrating that our method achieves the best average performance. This

[1]https://sketchfab.com/

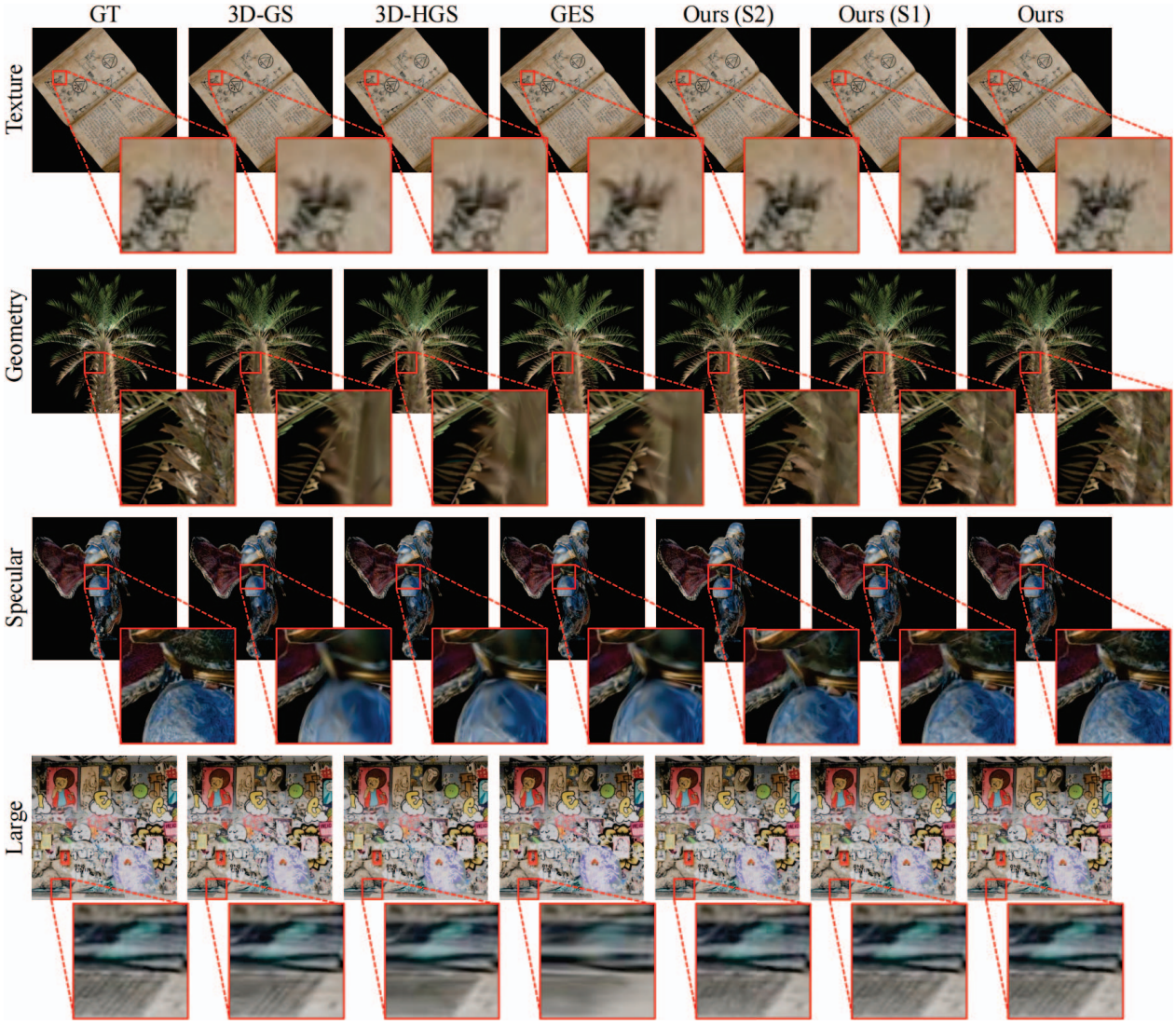

Figure 8. Qualitative comparisons of *DRK* with state-of-the-art methods across various scenarios show that *DRK* effectively captures sharp texture and geometry boundaries.

is followed by our sparser version, *DRK* (S1). *DRK* (S2) also performs comparably to other state-of-the-art methods, with significantly fewer primitives and lower space occupancy, while maintaining comparable FPS, as detailed in Table 2. Although our method requires more computation, the rendering framerates remain efficient (FPS $>$ 50).

Mip-NeRF360 Datasets. We also tested *DRK* on Mip-NeRF360 datasets, known for accurate camera poses and unbounded scenes. Results are shown in Tab. 3. On unbounded scenes, *DRK* still excels in perceptual quality (LPIPS, SSIM), it shows less advantage in PSNR, especially with sparser kernels. We hypothesize that DRK may tend to overfit distant regions with limited supervision. To address this, we used MiVOS[6] to mask image sequences, training on original images but evaluating only the foreground. The results (M-PSNR) indicate our method's strength in well-supervised central regions.

Methods	PSNR(↑)	M-PSNR(↑)	LPIPS(↓)	SSIM(↑)	Num(↓)	Size(↓)
2D-GS	26.32	32.01	.2987	.7617	928 K	216.0 M
3D-GS	26.48	32.23	.3045	.7537	811 K	191.3 M
3D-GS (L)	26.94	32.75	.2688	.7799	1191 K	281.6 M
3D-HGS	26.74	32.51	.2999	.7561	842 K	202.4 M
GES	26.62	32.11	.3047	.7542	824 K	195.0 M
DRK (S2)	26.20	32.31	.2781	.7601	388 K	125.9 M
DRK (S1)	26.40	32.56	.2601	.7722	551 K	161.9 M
DRK	26.76	32.81	.2364	.7871	952 K	279.1 M

Table 3. Quantitative evaluation on Mip-NeRF360 scenes [2].

5.3. Qualitative Comparison

We conducted qualitative comparisons to highlight the advantages of DRK over other state-of-the-art kernel representations. Fig.8 shows results across different scene categories, where our method achieves better texture and geometry fitting with a comparable or smaller number of kernels. Fig.9 demonstrates that DRK effectively handles fine ge-

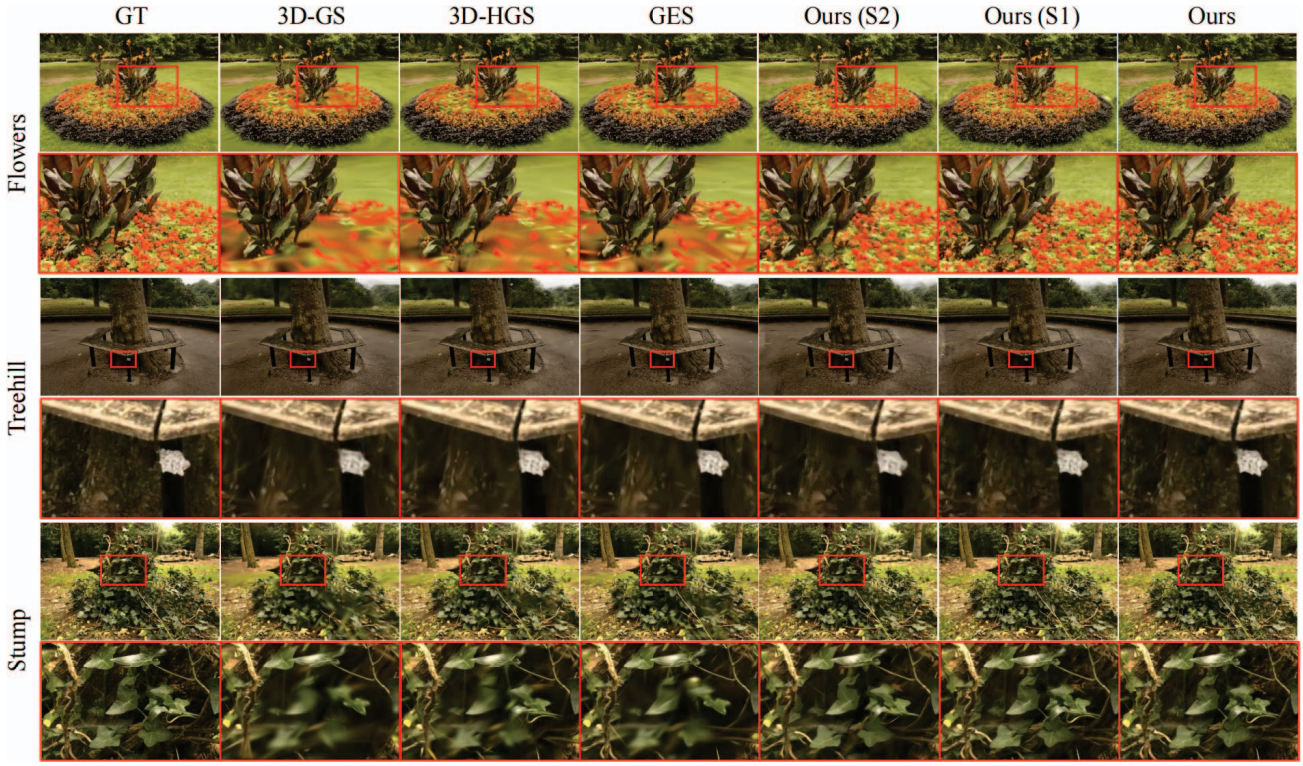

Figure 9. Comparisons on the MipNeRF 360 dataset show our method achieves clearer details and high-fidelity rendering.

ometries, such as flowers and leaves, and detailed textures, like tags and bark, outperforming other representations that may have used more primitives (see Tab. 3).

5.4. Converting Mesh to DRK

All existing "Splatting" methods [20, 22, 27, 30] fail to seamlessly integrate with traditional assets. In contrast, our *DRK* framework provides a versatile representation that allows for effortless conversion of triangle and polygon faces into *DRK*. This is accomplished by placing the *DRK* endpoints at the polygon vertices and adjusting the parameters τ and η to control sharpness, with both $L1$ and $L2$ curvature set to 1. This process eliminates the need for training data rendering or model optimization. Consequently, rich 3D assets can be seamlessly transferred into *DRK*, bridging the gap between **millions of traditional 3D assets** and our fast, high-quality reconstruction representation. Figure 10 illustrates the potential applications of Mesh2*DRK*.

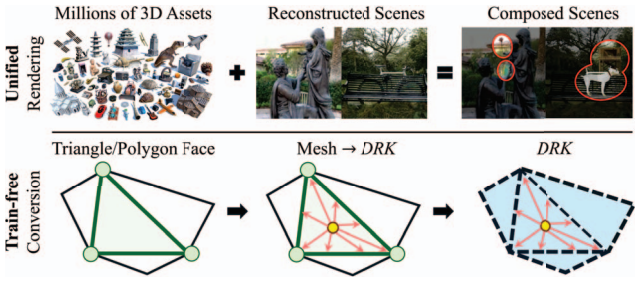

Figure 10. The conversion from Mesh to *DRK* demonstrates the efficiency of using *DRK* to incorporate traditional 3D assets into any *DRK* scene at low cost.

6. Conclusion

We present *DRK*, a novel primitive representation that generalizes and enhances Gaussian splatting. By incorporating learnable radial bases, adaptive shape control, and efficient rendering strategies, *DRK* effectively addresses the fundamental limitations of Gaussian kernels in representing diverse geometric features. Our experiments on both synthetic and real-world scenes demonstrate that *DRK* achieves superior rendering quality with significantly fewer primitives, while maintaining computational efficiency through carefully designed rasterization optimizations. Leveraging the flexibility of *DRK*, traditional triangle or polygon mesh representations can be seamlessly converted into *DRK* without requiring any training. This capability bridges rich 3D assets with reconstructed scenes, paving the way for future applications such as real-scene editing, VR digital world creation, and AR scene enhancement.

Acknowledgement

This work has been supported in part by Hong Kong Research Grant Council - Early Career Scheme (Grant No. 27209621), General Research Fund Scheme (Grant No. 17202422, 17212923), Theme-based Research (Grant No. T45-701/22-R) and Shenzhen Science and Technology Innovation Commission (SGDX20220530111405040). Part of the described research work is conducted in the JC STEM Lab of Robotics for Soft Materials funded by The Hong Kong Jockey Club Charities Trust.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 696–712. Springer, 2020. 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. 2, 6, 7

[3] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. NeRD: Neural reflectance decomposition from image collections. In *ICCV*, pages 12684–12694, 2021. 2

[4] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's gpus. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 17–141. IEEE, 2005. 5

[5] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001. 2

[6] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion. In *CVPR*, 2021. 7

[7] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7830–7839, 2020. 2

[8] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 1, 3

[9] Peng Dai, Feitong Tan, Xin Yu, Yifan Peng, Yinda Zhang, and Xiaojuan Qi. Go-nerf: Generating objects in neural radiance fields for virtual reality content creation. *IEEE Transactions on Visualization and Computer Graphics*, 2025. 2

[10] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. *Computer Graphics Forum*, 31(2pt1):305–314, 2012. 2

[11] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, 1996. 2

[12] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d gaussian splatting: Towards efficient novel view synthesis for dynamic scenes. *arXiv preprint arXiv:2402.03307*, 2024. 3

[13] Zhiwen Fan, Yifan Jiang, Peihao Wang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. Unified implicit neural stylization. In *ECCV*, pages 636–654, 2022. 2

[14] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. DeepView: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2367–2376, 2019. 2

[15] 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. 2

[16] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023. 1

[17] Yiming Gao, Yan-Pei Cao, and Ying Shan. SurfelNeRF: Neural surfel radiance fields for online photorealistic reconstruction of indoor scenes. *arXiv preprint arXiv:2304.08971*, 2023. 2

[18] 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*, pages 43–54, 1996. 2

[19] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024. 1, 3

[20] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19812–19822, 2024. 1, 6, 8

[21] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deli'ege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D convex splatting: Radiance field rendering with 3D smooth convexes. *arXiv*, abs/2411.14974, 2024. 2, 3

[22] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 1, 2, 3, 6, 8

[23] Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. StylizedNeRF: consistent 3D scene stylization as stylized NeRF via 2D-3D mutual learning. In *CVPR*, pages 18342–18352, 2022. 2

[24] Yi-Hua Huang, Yan-Pei Cao, Yu-Kun Lai, Ying Shan, and Lin Gao. Nerf-texture: Texture synthesis with neural radiance fields. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023. 2

[25] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4220–4230, 2024. 2

[26] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 2

[27] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time

radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 2, 3, 6, 8

[28] Zhengfei Kuang, Kyle Olszewski, Menglei Chai, Zeng Huang, Panos Achlioptas, and Sergey Tulyakov. NeROIC: Neural rendering of objects from online image collections. *ACM Trans. Graph.*, 41(4):1–12, 2022. 2

[29] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996. 2

[30] Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 3d-hgs: 3d half-gaussian splatting. *arXiv preprint arXiv:2406.02720*, 2024. 1, 3, 6, 8

[31] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. DeVRF: Fast deformable voxel radiance fields for dynamic scenes. In *NeurIPS*, pages 36762–36775, 2022. 2

[32] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 2019. 2

[33] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 2

[34] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 2

[35] Xiaoyang Lyu, Chirui Chang, Peng Dai, Yang-Tian Sun, and Xiaojuan Qi. Total-decom: decomposed 3d scene reconstruction with minimal interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20860–20869, 2024. 2

[36] Xiaoyang Lyu, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi. 3dgsr: Implicit surface reconstruction with 3d gaussian splatting. *arXiv preprint arXiv:2404.00409*, 2024. 2

[37] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6878–6887, 2019. 2

[38] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 2

[39] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2

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

[41] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3D Ken Burns effect from a single image. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019. 2

[42] Yi-Ling Qiao, Alexander Gao, and Ming Lin. NeuPhysics: Editable neural geometry and physics from monocular videos. In *NeurIPS*, pages 12841–12854, 2022. 2

[43] Haoxuan Qu, Zhuoling Li, Hossein Rahmani, Yujun Cai, and Jun Liu. Disc-gs: Discontinuity-aware gaussian splatting. *arXiv preprint arXiv:2405.15196*, 2024. 1, 3

[44] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering. *ACM Transactions on Graphics (TOG)*, 43(4):1–17, 2024. 6

[45] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 623–640. Springer, 2020. 2

[46] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12216–12225, 2021. 2

[47] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 6

[48] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999. 2

[49] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. DeepVoxels: Learning persistent 3D feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2437–2446, 2019. 2

[50] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. NeRFPlayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE TVCG*, 29(5):2732–2742, 2023. 2

[51] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 175–184, 2019. 2

[52] Yang-Tian Sun, Yi-Hua Huang, Lin Ma, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Splatter a video: Video gaussian representation for versatile processing. *arXiv preprint arXiv:2406.13870*, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 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. *arXiv preprint arXiv:2309.16653*, 2023. 2

[55] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 551–560, 2020. 2

[56] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3D reconstructions. In *Proceedings of the European conference on computer vision (ECCV)*, pages 836–850. Springer, 2014. 2

[57] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7467–7477, 2020. 2

[58] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. NeX: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8534–8543, 2021. 2

[59] Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. Surface light fields for 3D photography. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, 2000. 2

[60] Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. Recent advances in 3d gaussian splatting. *Computational Visual Media*, 10(4):613–642, 2024. 2

[61] Xiuzhe Wu, Peng Dai, Weipeng Deng, Handi Chen, Yang Wu, Yan-Pei Cao, Ying Shan, and Xiaojuan Qi. Cl-nerf: continual learning of neural radiance fields for evolving scene representation. *Advances in Neural Information Processing Systems*, 36:34426–34438, 2023. 2

[62] Ziyi Yang, Yanzhen Chen, Xinyu Gao, Yazhen Yuan, Yu Wu, Xiaowei Zhou, and Xiaogang Jin. Sire-ir: Inverse rendering for brdf reconstruction with shadow and illumination removal in high-illuminance scenes. *arXiv preprint arXiv:2310.13030*, 2023. 2

[63] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 3

[64] Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. *arXiv preprint arXiv:2402.15870*, 2024. 2

[65] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024. 2

[66] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[67] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. ARF: Artistic radiance fields. In *ECCV*, pages 717–733, 2022. 2

[68] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

[69] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 2018. 2

[70] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10324–10335, 2024. 2