3DGUT: Enabling Distorted Cameras and Secondary Rays in Gaussian Splatting

Qi Wu[1]*, Janick Martinez Esturo[1]*, Ashkan Mirzaei[1,2], Nicolas Moenne-Loccoz[1], Zan Gojcic[1]
[1]NVIDIA, [2]University of Toronto

https://research.nvidia.com/labs/toronto-ai/3DGUT/

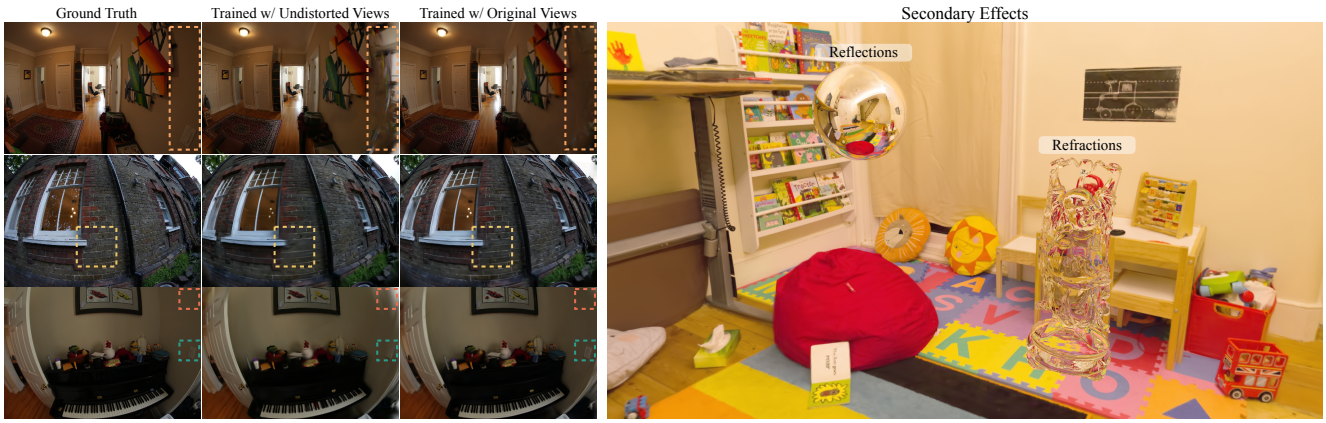

Figure 1. We extend 3D Gaussian Splatting (3DGS) to support nonlinear camera projections and secondary rays for simulating phenomena such as reflections and refractions. By replacing EWA splatting rasterization with the Unscented Transform, our approach retains real-time efficiency while accommodating complex camera effects like rolling shutter. (Left) A comparison of our model trained on undistorted views vs. the original distorted fisheye views, showing that training on the full set of pixels improves visual quality. (Right) Two synthetic objects, a reflective sphere and a refractive statue, inserted into a scene reconstructed with our model.

Abstract

3D Gaussian Splatting (3DGS) has shown great potential for efficient reconstruction and high-fidelity real-time rendering of complex scenes on consumer hardware. However, due to its rasterization-based formulation, 3DGS is constrained to ideal pinhole cameras and lacks support for secondary lighting effects. Recent methods address these limitations by tracing volumetric particles instead, however, this comes at the cost of significantly slower rendering speeds. In this work, we propose 3D Gaussian Unscented Transform (3DGUT), replacing the EWA splatting formulation in 3DGS with the Unscented Transform that approximates the particles through sigma points, which can be projected exactly under any nonlinear projection function. This modification enables trivial support of distorted cameras with time dependent effects such as rolling shutter, while retaining the efficiency of rasterization. Additionally, we align our rendering formulation with that of tracing-based methods, enabling secondary ray tracing required to represent phenomena such as reflections and refraction within the same 3D representation.

* denotes equal contribution.

1. Introduction

Multiview 3D reconstruction and novel view synthesis is a classical problem in computer vision, for which several scene representations have been proposed in recent years, including points [21, 39], surfaces [5, 38, 52], and volumetric fields [32, 34, 49, 51]. Most recently, driven by 3D Gaussian Splatting [17] (3DGS), volumetric particle-based representations have gained significant popularity due to their high visual fidelity and fast rendering speeds. The core idea of 3DGS is to model scenes as an unstructured collection of fuzzy 3D Gaussian particles, each defined by its location, scale, rotation, opacity, and appearance. These particles can be rendered differentiably in real time via rasterization, allowing their parameters to be optimized through a re-rendering loss function.

High frame-rates of 3DGS, especially compared to volumetric ray marching methods, can be largely accredited to the efficient rasterization of particles. However, this reliance on rasterization also imposes some inherent limitations. The EWA splatting formulation [56] does not support highly-distorted cameras with complex time dependent effects such as rolling shutter. Additionally, rasterization cannot simulate secondary rays required for representing phenomena like reflection, refraction, and shadows.

Instead of rasterization, recent works have proposed to

render the volumetric particles using ray tracing [6, 29, 33]. While this mitigates the shortcomings of rasterization, it does so at the expense of significantly reduced rendering speed, even when the tracing formulation is heavily optimized for semi-transparent particles [33]. In this work, we instead aim to overcome the above limitations of 3DGS while remaining in the realm of rasterization, thereby maintaining the high-rendering rates. To this end, we seek answer to the following two questions:

What makes 3DGS ill-suited to represent distorted cameras and rolling shutter? To project 3D Gaussian particles onto the camera image plane, 3DGS relies on an EWA splatting formulation that requires computing the Jacobian of the non-linear projection function. This leads to approximation errors, even for perfect pinhole cameras, and the errors become progressively worse with increasing distortion [13]. Moreover, it is unclear how to even represent time-dependent effect such as rolling-shutter within the EWA splatting formulation.

Instead of approximating the non-linear projection function, we draw inspiration from the classical literature of Unscented Kalman Filter [15] and approximate the 3D Gaussian particles using a set of carefully selected sigma points. These sigma points can be projected exactly onto the camera image plane by applying an arbitrarily complex projection function to each point, after which a 2D Gaussian can be re-estimated from them in form of a Unscented Transform (UT) [11]. Apart from a better approximation quality, UT is derivative-free and completely avoids the need to derive the Jacobians for different camera models (Fig. 1 left). Moreover, complex effects such as rolling shutter distortions can directly be represented by transforming each sigma point with a different extrinsic matrix.

Can we align the rasterization rendering formulation with the one of ray-tracing? The rendering formulations mainly differ in terms of: (i) determining which particles contribute to which pixels, (ii) the order in which the particles are intersected, (iii) how the particles are evaluated. To align the representations we therefore follow 3DGRT [33] and evaluate the Gaussian particle response in 3D, while sorting them in order similar to Radl et al. [36]. While small differences persist, this provides us with a representation that can be both rasterized and ray-traced, enabling secondary-rays required to simulate phenomena like refraction and reflection (Fig. 1 right).

In summary, we propose 3D Gaussian Unscented Transform (3DGUT), where our main contributions are:

- We derive a rasterization formulation that approximates the 3D Gaussian particles instead of approximating the non-linear projection function. This simple change enables us to extend 3DGS to arbitrary camera models and to support complex time dependent effects such as rolling shutter.
- We align the rendering formulation with 3DGRT, which allows us to render the same representation with rasterization and ray-tracing, supporting phenomena such as refraction and reflections.

On multiple datasets, we demonstrate that our formulation leads to comparable rendering rates and image fidelity to 3DGS, while offering greater flexibility and outperforming dedicated methods on datasets with distorted cameras.

2. Related Work

Neural Radiance Fields Neural Radiance Fields (NeRFs) [32] have transformed the field of novel view synthesis, by modeling scenes as emissive volume encoded within coordinate-based neural network. These networks can be queried at any spatial location to return the volume density and view-dependent radiance. Novel views are synthesized by sampling the network along camera rays and accumulating radiance through volumetric rendering. While the original formulation [32] utilized a large, global multi-layer perceptron (MLP), subsequent work has explored more efficient scene representations, including voxel grids [26, 41, 44], triplanes [3], low-rank tensors [4], and hash tables [34]. Despite these advances, even highly optimized NeRF implementations [34] still struggle to achieve real-time inference rates due to the computational cost of ray marching.

To accelerate inference, several efforts have focused on converting the radiance fields into more efficient representations, such as meshes [5, 52], hybrid surface-volume representations [43, 46, 48, 50], and sparse volumes [7, 8, 37]. However, these approaches generally require a cumbersome two-step pipeline: first training a conventional NeRF model and then baking it into a more performant representation, which further increases the training time and complexity.

Volumetric Particle Representations Differentiable rendering via alpha compositing has also been explored in combination with volumetric particles, such as spheres [22]. More recently, 3D Gaussian Splatting [17] replaced spheres with fuzzy anisotropic 3D Gaussians. Instead of ray marching, these explicit volumetric particles can be rendered through highly efficient rasterization, achieving competitive results in terms of quality and efficiency. Due to its simplicity and flexibility, 3DGS has inspired numerous follow-up works focusing on improving memory efficiency [23, 28, 30], developing better densification and pruning heuristics [19, 53], enhancing surface representation [9, 10], and scaling up to large scenes [18, 25, 27]. However, while rasterization is very efficient, it also introduces trade-offs, such as being limited to perfect pinhole cameras. Prior work has attempted to work around these limitations and support complex camera models such as fisheye cameras [24] or rolling shutter [42]. But these works

still require dedicated formulation for each camera type and exhibit quality degradation with increased complexity and distortion of the camera models [13].

In response, recent works have explored replacing rasterization entirely and instead rendering the 3D Gaussians using ray tracing [6, 29, 33]. Ray tracing inherently supports complex camera models and enables secondary effects like shadows, refraction, and reflections through secondary rays. However, this comes with a substantial decrease in rendering efficiency: even the most optimized ray-tracing methods are still 3-4 times slower than rasterization [33].

In this work, we instead propose a generalized approach for efficiently handling complex camera models within the rasterization framework, thereby preserving the computational efficiency. Additionally we unify our rendering formulation with the one of ray-tracing, enabling a hybrid rendering technique within the same representation.

Unscented Transform Computing the statistics of a random variable that has undergone a transformation is one of the fundamental tasks in the fields of estimation and optimization. When the transformation is non-linear, however, no closed form solution exists, so several approximations have been proposed . The simplest and perhaps most widely used approach is to linearize the non-linear transformation using the first order Taylor approximation. However, the local linearity assumption is often violated, and derivation of the Jacobian matrix is non-trivial and error prone. The Unscented Transform (UT) [15, 16] was proposed to address these limitations. The key idea of UT is to approximate the distribution of the random variable using a set of Sigma points that can be transformed exactly, after which they can be used to re-estimate the statistics of the random variable in the target domain. Originally, UT was devised for filtering-based state estimation [15, 47], but it has since found applications in computer vision [2, 14]. Notably, UT has even been explored in the context of novel-view synthesis [2], where it was used to estimate the ray frustum from samples that match its first and second moments.

3. Preliminaries

We provide a short review of 3D Gaussian parametrization, volumetric particle rendering, and EWA splatting.

3D Gaussian Splatting Representation: Kerbl et al. [17] represent scenes using an unordered set of 3D Gaussian particles whose response function $\rho : \mathbb{R}^3 \to \mathbb{R}$ is defined as

$$\rho(\boldsymbol{x}) = \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})), \tag{1}$$

where $\boldsymbol{\mu} \in \mathbb{R}^3$ denotes the particle's position and $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ its covariance matrix. To ensure that $\boldsymbol{\Sigma}$ remains positive semi-definite during gradient-based optimization, it is

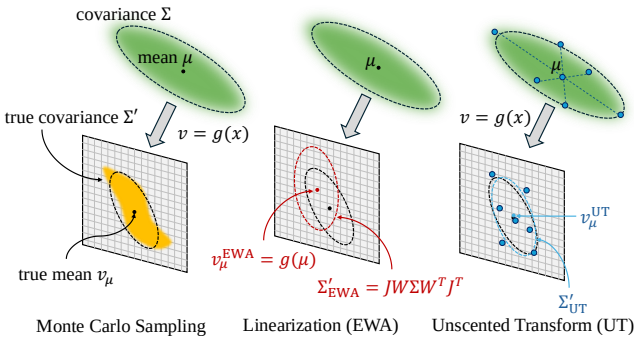

Figure 2. When projecting a Gaussian particle from 3D space onto the camera image plane, Monte Carlo sampling (*left*) provides the most accurate estimate but is costly to compute. EWA Splatting formulation used in [17] approximates the projection function via linearization, which requires a dedicated Jacobian J for each camera model and leads to approximation errors with increasing distortion. Unscented Transform instead approximates the particle with Sigma points than can be projected exactly and from which the 2D conic can then be estimated.

decomposed into a rotation matrix $\boldsymbol{R} \in \mathrm{SO}(3)$ and a scaling matrix $\boldsymbol{S} \in \mathbb{R}^{3\times3}$, such that

$$\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{R}^T \tag{2}$$

In practice, both $\boldsymbol{R}$ and $\boldsymbol{S}$ are stored as vectors—a quaternion $\boldsymbol{q} \in \mathbb{R}^4$ for the rotation and a vector $\boldsymbol{s} \in \mathbb{R}^3$ for the scaling. Each particle is also associated with an opacity coefficient, $\sigma \in \mathbb{R}$, and a view-dependent parametric radiance function $\phi_{\boldsymbol{\beta}}(\boldsymbol{d}) : \mathbb{R}^3 \to \mathbb{R}^3$, with $\boldsymbol{d}$ the incident ray direction, which is in practice represented using spherical harmonics functions of order $m = 3$.

Determining the Particle Response: Within the 3DGS rasterization framework, the 3D particles first need to be projected to the camera image plane in order to determine their contributions to the individual pixels. To this end, 3DGS follows [56] and computes a covariance matrix $\boldsymbol{\Sigma'} \in \mathbb{R}^{2\times2}$ for a projected Gaussian in image coordinates via first-order approximation as

$$\boldsymbol{\Sigma'} = \boldsymbol{J}_{[:2,:3]}\boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^T\boldsymbol{J}^T_{[:2,:3]} \tag{3}$$

where $\boldsymbol{W} \in \mathrm{SE}(3)$ transforms the particle from the world to the camera coordinate system, and $\boldsymbol{J} \in \mathbb{R}^{3\times3}$ denotes the Jacobian matrix of the affine approximation of the projective transformation, which is obtained by considering the linear terms of its Taylor expansion. The Gaussian response of a particle i for a position $\boldsymbol{x} \in \mathbb{R}^3$ can then be computed in 2D from its projection on the image plane $\boldsymbol{v_x} \in \mathbb{R}^2$ as

$$\rho_i(\boldsymbol{x}) = \exp(-\frac{1}{2}(\boldsymbol{v_x}-\boldsymbol{v_{\mu_i}})^T\boldsymbol{\Sigma'}_{\boldsymbol{i}}^{-1}(\boldsymbol{v_x}-\boldsymbol{v_{\mu_i}})) \tag{4}$$

where $\boldsymbol{v_{\mu_i}} \in \mathbb{R}^2$ denotes the projected mean of the particle.

Volumetric Particle Rendering: The color $\boldsymbol{c} \in \mathbb{R}^3$ of a camera ray $\boldsymbol{r}(\tau) = \boldsymbol{o} + \tau\boldsymbol{d}$ with origin $\boldsymbol{o} \in \mathbb{R}^3$ and direction $\boldsymbol{d} \in \mathbb{R}^3$ can be rendered from the above volumetric particle representation using numerical integration

$$\boldsymbol{c}(\boldsymbol{o}, \boldsymbol{d}) = \sum_{i=1}^{N} \boldsymbol{c}_i(\boldsymbol{d}) \alpha_i \prod_{j=1}^{i-1} 1 - \alpha_j, \tag{5}$$

where N denotes the number of particles that contribute to the given ray and opacity $\alpha_i \in \mathbb{R}$ is defined as $\alpha_i = \sigma_i \rho_i(\boldsymbol{o} + \tau\boldsymbol{d})$ for any $\tau \in \mathbb{R}^+$.

4. Method

Our aim is to extend 3DGS [17] and 3DGRT [33] methods by developing a formulation that:

- accommodates highly distorted cameras and time-dependent camera effects, such as rolling shutter,
- unifies the rendering formulation to allow the same reconstructions to be rendered using either splatting or tracing, enabling hybrid rendering with traced secondary rays

all while preserving the efficiency of rasterization. We begin by detailing our approach to bypass the linearization steps of 3DGS [17] in Sec. 4.1, followed by an approach to evaluate the particles in order and directly in 3D (Sec. 4.2). The former enables support for complex camera models, while the latter aligns the rendering formulation with 3DGRT [33].

4.1. Unscented Transform

As illustrated in Fig. 2, the EWA splatting formulation used in 3DGS for projecting 3D Gaussian particles onto the camera image plane relies on the linearization of the affine approximation of the projective transform (Eq. (3)). This approach, however, has several notable limitations: (i) it neglects higher-order terms in the Taylor expansion, leading to projection errors even with perfect pinhole cameras [13], and these errors increase with camera distortion; (ii) it requires deriving a new Jacobian for each specific camera model (e.g., the equidistant fisheye model in [24]), which is cumbersome and error prone; (iii) it necessitates representing the projection as a single function, which is particularly challenging when accounting for time-dependent effects such as rolling shutter.

To overcome these limitations, we build on the ideas of the Unscented Transform (UT) and propose to instead approximate the volumetric particle using a set of carefully selected Sigma points. Specifically, consider the 3D Gaussian scene representation described in Sec. 3, where particles are characterized by their position $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The Sigma points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=0}^{6}$ are then defined as

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{\mu} & \text{for } i = 0 \\ \boldsymbol{\mu} + \sqrt{(3+\lambda)\boldsymbol{\Sigma}}_{[i]} & \text{for } i = 1, 2, 3 \\ \boldsymbol{\mu} - \sqrt{(3+\lambda)\boldsymbol{\Sigma}}_{[i-3]} & \text{for } i = 4, 5, 6 \end{cases} \tag{6}$$

using the available factorization Eq. (2) of the covariance to read of the matrix square-root.

Their corresponding weights $\mathcal{W} = \{w_i\}_{i=0}^{6}$ are given as

$$w_i^{\mu} = \begin{cases} \frac{\lambda}{3+\lambda} & \text{for } i = 0 \\ \frac{1}{2(3+\lambda)} & \text{for } i = 1, \ldots, 6 \end{cases} \tag{7}$$

$$w_i^{\Sigma} = \begin{cases} \frac{\lambda}{3+\lambda} + (1 - \alpha^2 + \beta) & \text{for } i = 0 \\ \frac{1}{2(3+\lambda)} & \text{for } i = 1, \ldots, 6 \end{cases} \tag{8}$$

where $\lambda = \alpha^2(3 + \kappa) - 3$, α is a hyperparameter that controls the spread of the points around the mean, κ is a scaling parameter typically set to 0, and β is used to incorporate prior knowledge about the distribution [47].

Each Sigma point can then be independently projected onto the camera image plane using the non-linear projection function $\boldsymbol{v}_{x_i} = g(\boldsymbol{x}_i)$. The 2D conic can subsequently be approximated as the weighted posterior sample mean and covariance matrix of the Gaussian:

$$\boldsymbol{v}_{\boldsymbol{\mu}} = \sum_{i=0}^{6} w_i^{\mu} \boldsymbol{v}_{x_i} \tag{9}$$

$$\boldsymbol{\Sigma}' = \sum_{i=0}^{6} w_i^{\Sigma} (\boldsymbol{v}_{x_i} - \boldsymbol{v}_{\boldsymbol{\mu}})(\boldsymbol{v}_{x_i} - \boldsymbol{v}_{\boldsymbol{\mu}})^{\mathrm{T}} \tag{10}$$

With the 2D conic computed, we can apply the same tiling and culling procedures as proposed by [17, 36] to determine which particles influence which pixels. As described in the following section, our particle response evaluation does not depend on the 2D conic. Instead, UT only acts as an acceleration structure to efficiently determine the particles that contribute to each pixel thus avoiding the need for computing the backward pass through the non-linear projection function.

4.2. Evaluating Particle Response

Once the Gaussian particles contributing to each pixel have been identified, we need to determine how to evaluate their response. Following 3DGRT [33], we evaluate particles directly in 3D by using a single sample located at the point of maximum particle response along a given ray.

A comparison between 3DGS's 2D conic response evaluation method and our 3D response evaluation method is provided in Fig. 3. Specifically, we compute the distance

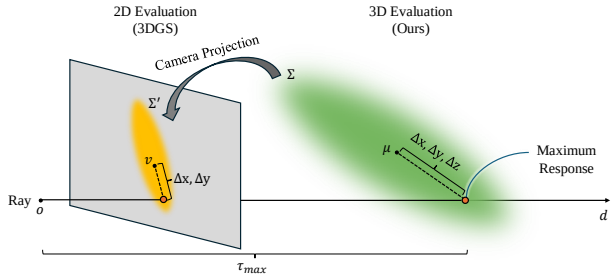

Figure 3. For a given ray, 3DGS [17] evaluates the response of the Gaussian particle in 2D after the projection onto the camera image plane. This requires backpropagation through the (approximated) projection function. Instead, we follow [33] and evaluate particles in 3D at the point of the maximum response along the ray.

$\tau_{\text{max}} = \text{argmax}_\tau \rho(\boldsymbol{o} + \tau \boldsymbol{d})$, which maximizes the particle response along the ray $\boldsymbol{r}(\tau)$, as

$$\tau_{\text{max}} = \frac{(\boldsymbol{\mu} - \boldsymbol{o})^T \boldsymbol{\Sigma}^{-1} \boldsymbol{d}}{\boldsymbol{d}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{d}} = \frac{-\boldsymbol{o}_g^T \boldsymbol{d}_g}{\boldsymbol{d}_g^T \boldsymbol{d}_g} \tag{11}$$

where $\boldsymbol{o}_g = \boldsymbol{S}^{-1} \boldsymbol{R}^T (\boldsymbol{o} - \boldsymbol{\mu})$ and $\boldsymbol{d}_g = \boldsymbol{S}^{-1} \boldsymbol{R}^T \boldsymbol{d}$.

Unlike 3DGS, which performs particle evaluations in 2D, our approach avoids propagating gradients through the projection function, thereby avoiding the approximations and mitigating potential numerical instabilities. The derivation of the numerically stable backward pass is detailed in the Supplementary Material.

4.3. Sorting Particles

The proposed volumetric rendering formulation, i.e. both the rendering equation Eq. (5) and the particle evaluation Eq. (11), is equivalent to the one used in 3DGRT. However, while 3DGRT is able to collect the hit particles in their exact τ_{max} order along the ray thanks to a dedicated acceleration structure [35], 3DGS sorts them globally for each tile. In order to get a better approximation of the τ_{max} order we propose to use the multi-layers alpha blending approximation (MLAB) [40] following [36].[1] It consists in storing the per-ray k-farthest hit particles (typically using k = 16) in a buffer. The closest hits which cannot be stored in the buffer are incrementally alpha-blended until the transmittance of the blended part vanishes.

As an alternative, the hybrid transparency (HT) blending strategy [31] has been recently used for splatting Gaussian particles [12]. Instead of storing the k-farthest hit particles and incrementally blending the closest hits, HT stores the k-closest and incrementally blends the farthest hits. This permits to recover the exact k-closest hit particles, but requires to go through all particles, which may be prohibitively slow without dedicated optimizations and heuristics.

[1] StopThePop [36] denotes the MLAB formulation as k-buffer approach.

4.4. Implementation and Training

We build on the work of [17, 33] and implemented our method in PyTorch, using custom CUDA kernels for the compute-intensive parts. Additionally, we employ advanced culling strategies proposed by Radl et al. [36]. Unless otherwise specified, we adopt all parameters from 3DGS [17] to ensure a fair comparison and keep them consistent across all evaluations.

Similar to [33] we don't have access to 2D screen space gradients, so we follow 3DGRT [33] and replace them with the 3D positional gradients divided by half of the distance to the camera and perform densification and pruning every 300 iterations. For the UT, we set $\alpha = 1.0$, $\beta = 2.0$ and $\kappa = 0.0$ in all evaluations. We train our model for 30k iterations using the weighted sum of the L2-loss $\mathcal{L}_2$ and the perceptual loss $\mathcal{L}_{\text{SSIM}}$ sucht that $\mathcal{L} = \mathcal{L}_2 + 0.2\mathcal{L}_{\text{SSIM}}$.

5. Experiments and Ablations

In this section, we first evaluate the proposed approach on standard novel-view synthesis benchmark datasets [1, 20], analyzing both quality and speed. We additionally evaluate our method on an indoor dataset captured with fisheye cameras [54], as well as an autonomous driving dataset captured using distorted cameras with rolling shutter effect [45]. Ablation studies on key design choices and additional details on experiments and implementation are provided in the Supplementary Material.

Model Variants. In the following evaluation, we will refer to two variants of our method. We use *Ours* to denote the version that extends 3DGS [17] with the UT formulation (Sec. 4.1) and particle evaluation in 3D (Sec. 4.2). The second variant *Ours (sorted)* additionally uses the per-ray sorting strategy as detailed in Sec. 4.3 that leads to unification with 3DGRT [33] .

Metrics. We evaluate the perceptual quality of the novel views using peak signal-to-noise ratio (PSNR), learned perceptual image patch similarity (LPIPS), and structural similarity (SSIM) metrics. To assess performance, we measure the time required for rendering a single image, excluding any overhead from data storage or visualization. For all evaluations, we use the datasets' default resolutions and report frames per second (FPS) measured on a single NVIDIA RTX 6000 Ada GPU.

Baselines. There have been many follow up works that improve or extend 3DGS in different aspects [6, 12, 19, 28, 55]. Many of these improvements are compatible with our approach, so we limit our comparison to the original 3DGS [17] and StopThePop [36] as the representative splatting methods, along with 3DGRT [33] and EVER [29] as volumetric particle tracing methods that natively support distorted cameras and secondary lighting effects. On

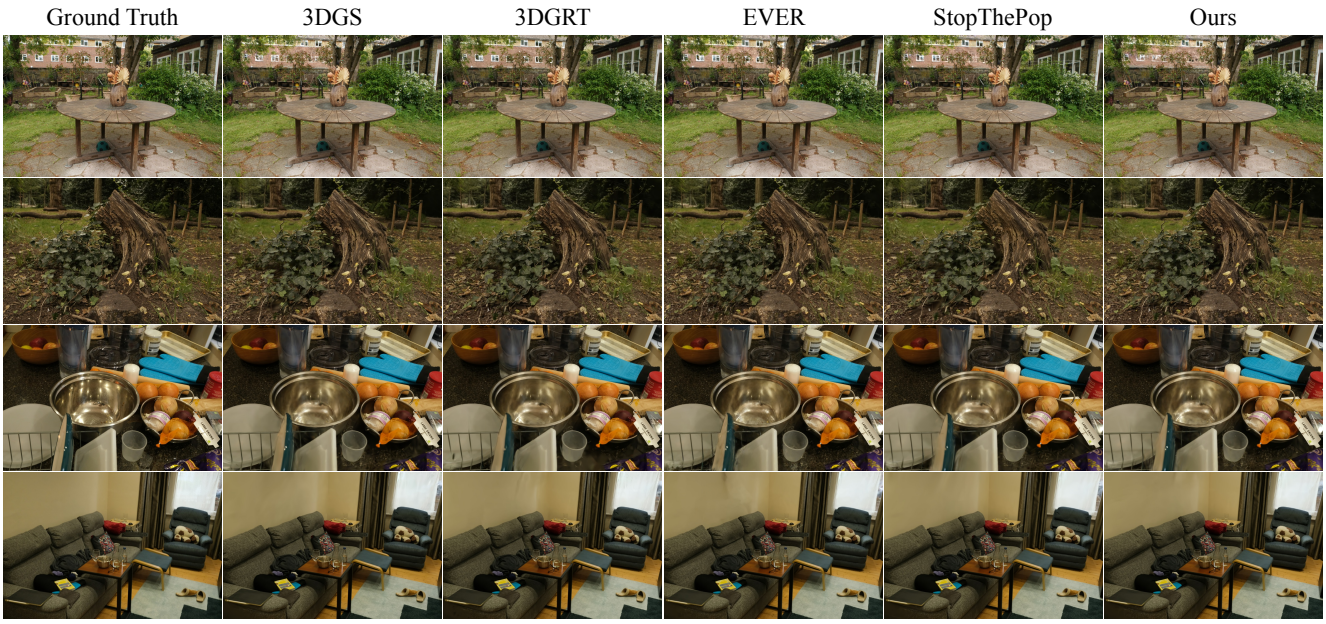

Figure 4. Qualitative comparison of our novel-view synthesis results against the baselines on the MipNERF360 dataset [1].

Table 1. Quantitative results of our approach and baselines on a MipNERF360 [1] and Tanks & Temples [20] datasets.

Method\Metric	Complex Cameras	Without Popping	MipNeRF360 PSNR↑	MipNeRF360 SSIM↑	MipNeRF360 LPIPS↓	MipNeRF360 FPS ↑	Tanks & Temples PSNR↑	Tanks & Temples SSIM↑	Tanks & Temples LPIPS↓	Tanks & Temples FPS ↑
ZipNeRF [2]	✓	/	28.54	0.828	0.219	0.2	/	/	/	/
3DGS [17]	✗	✗	27.26	0.803	0.240	347	23.64	0.837	0.196	476
Ours	✓	✗	27.26	0.810	0.218	265	23.21	0.841	0.178	277
StopThePop [36]	✗	✓	27.14	0.804	0.235	340	23.15	0.837	0.189	482
3DGRT [33]	✓	✓	27.20	0.818	0.248	52	23.20	0.830	0.222	190
EVER [29]	✓	✓	27.51	0.825	0.233	36	/	/	/	/
Ours (sorted)	✓	✓	27.26	0.812	0.215	200	22.90	0.844	0.172	272

Table 2. Detailed timings on the MipNeRF360 [1] dataset

Timings in ms	Preprocess	Duplicate	Sort	Render	Total
3DGS [17]	0.59	0.34	0.55	1.27	2.88
Ours	1.34	0.31	0.33	1.61	3.77
StopThePop [36]	0.57	0.27	0.14	1.83	2.94
3DGRT [33]	/	/	/	19.24	19.24
Ours (sorted)	1.24	0.47	0.24	2.85	4.98

the dataset captured with fisheye cameras, we compare our method to FisheyeGS [24] which extended 3DGS to fisheye cameras by deriving the Jacobian of the equidistant fisheye camera model. In addition to volumetric particle-based methods, we also compare our approach to state-of-the-art NeRF method ZipNeRF [2].

5.1. Novel View Synthesis Benchmarks

MipNeRF360 [1]. is the most popular novel-view synthesis benchmark consisting of nine large scale outdoor and indoor scenes. Following prior work, we used the images downsampled by a factor of four for the outdoor scenes, and by a factor of two for the indoor scenes. To enable comparison with other splatting method, we use rectified images provided by Kerbl et al. [17].

Tab. 1 depicts the quantitative comparison, while the qualitative comparison on selected scenes is provided in Fig. 4. As anticipated, on this dataset with perfect pinhole inputs, both *Ours* and *Ours (sorted)* achieve comparable perceptual quality to other splatting and tracing methods. In terms of inference runtime, our method achieves comparable frame rates to 3DGS [17], while greatly outperforming all other methods that support complex cameras at more than 265FPS while the closest competitor, 3DGRT [33], achieves 52FPS.

Tanks & Temples [20]. contains two large-scale outdoor scenes where the camera circulates around a prominent object (*Truck* and *Train*). Both scenes include lighting variations, and the *Truck* scene also contains transient objects that should ideally be ignored by reconstruction methods. Tab. 1 depicts the quantitative comparison while the quali-

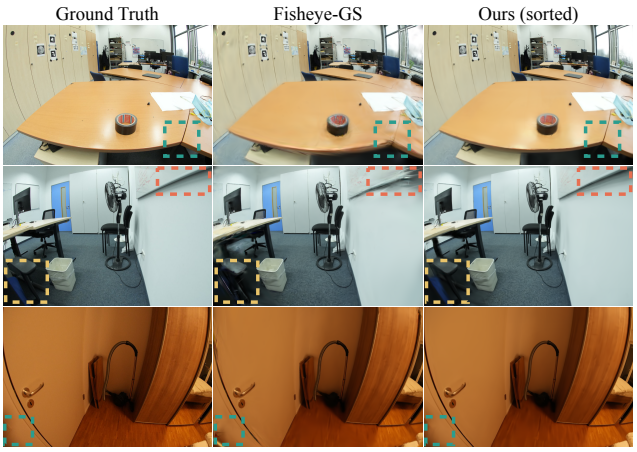

Figure 5. Comparison of our renderings against Fisheye-GS [24], on scenes from the Scannet++ dataset [54].

tative results are provided in the Supplementary Material.

Scannet++ [54]. is a large-scale indoor dataset captured with a fisheye camera at a resolution of 1752 × 1168 pixels. For our evaluation, we use the same six scenes as FisheyeGS [24] and follow the same pre-processing steps. Specifically, we convert the images to an equidistant fisheye camera model to match the requirements of [24]. [2]

On this dataset, we compare *Ours* to FisheyeGS [24] and 3DGS [24]. The results for the latter are taken from [24] where they were obtained by: (i) undistorting the training images and training with the official 3DGS [17] implementation, and (ii) rendering equidistant fisheye test views from that representation using the FisheyeGS [24] formulation. This setting is unfavorable for 3DGS [24] as significant portions of the images are lost during undistortion, but it highlight the problem of being limited to perfect pinhole cameras. The quantitative comparison is shown in Tab. 3 and qualitative results are provided in Fig. 5. *Ours* significantly outperforms FisheyeGS [24] across all perceptual metrics, while using less than half the particles (1.07M vs. 0.38M). This result underscores the flexibility and potential of our approach. Despite FisheyeGS [24] deriving a Jacobian for this particular camera model—limiting its applicability even to similar models (e.g., fisheye with distortions)—it still underperforms our simple formulation that can be trivially applied to any camera model.

Waymo [45]. is a large scale autonomous driving dataset captured using distorted cameras with rolling-shutter. We follow 3DGRT [33] and select 9 scenes with no dynamic objects to ensure accurate reconstructions. Fig. 6 show qualitative results. *Ours (sorted)* can faithfully represent complex camera mounted on a moving platform and reaches comparable performance to 3DGRT [33]. More quantitative and qualitative results are provided in the Supplementary Material.

[2]Note that our method seamlessly supports the full fisheye camera model without any code modifications.

Table 3. When evaluated on a dataset acquired with equidistant fisheye cameras, our general method outperforms [24] which derived the linerization for this specific camera model. Undistortion removes large parts of the original images and results in underobserved regions [17]. Results marked with † are taken from [24].

Method\Metric	Scannet++ PSNR↑	SSIM↑	LPIPS↓	N. Gaussians↓
3DGS†	22.76	0.798	/	1.31M
FisheyeGS† [24]	27.86	0.897	/	1.25M
FisheyeGS [24]	28.15	0.901	0.261	1.07M
Ours (sorted)	29.11	0.910	0.252	0.38M

Figure 6. Qualitative comparison of our novel-view synthesis results against 3DGRT on the Waymo dataset [45].

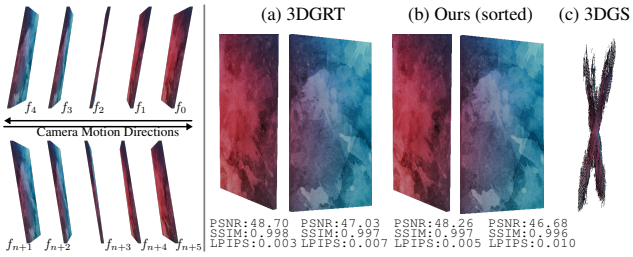

Figure 7. Multiple frame tiles f_i of a *single* solid box rendered by a left- and right-panning rolling shutter camera with a top-to-bottom shutter direction illustrate this *time-dependent* sensor effect (data from [33]). While ray-tracing-based methods like 3DGRT naturally support compensating for these time-dependent effects (a), traditional splatting methods struggle to model these (c), whereas our UT-based splatting formulation faithfully incorporates the sensor's motion into the projection formulation and recuperates the true undistorted geometry (b).

6. Applications

3DGUT also enables novel applications and techniques that were previously unattainable with particle scene representation within a rasterization framework.

6.1. Complex cameras

Distorted Camera Models. Projection of particles using UT enables 3DGUT not only to train with distorted cam-

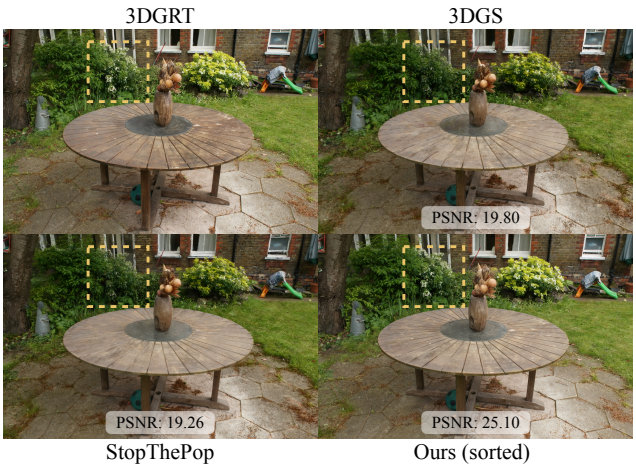

Figure 8. Scenes trained with different methods and rendered using 3DGRT [33]. Our method is the most consistent with the tracing approach, allowing for seamless hybrid rendering with splatting for primary and tracing for secondary rays.

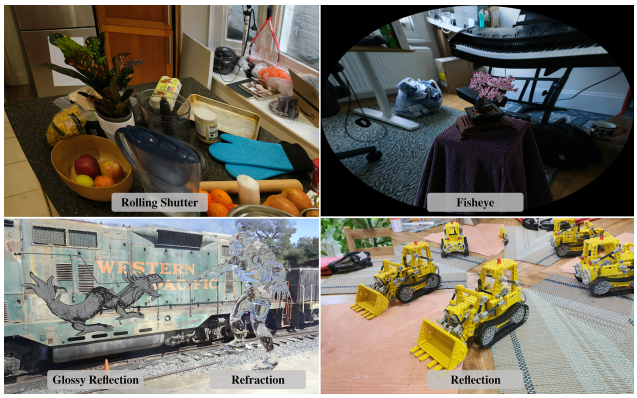

Figure 9. Illustration of the effects unlocked by our method. Top-left : rendering an image with rolling-shutter. Top-right : applying a strong lens distortion. Bottom : hybrid splatting / tracing rendering. Primary rays are splatted using our method while secondary rays are traced using 3DGRT [33]. This is possible by our 3D consistent representation and permits to simulate refraction (left) and reflections (right).

eras, but also to render different camera models with varying distortion from scenes that were trained using perfect pinhole camera inputs (Fig. 9 top row).

Rolling Shutter. Apart from the modeling of distorted cameras, 3DGUT can also faithfully incorporate the camera motion into the projection formulation, hence offering support for time-dependent camera effects such as rolling-shutter, which are commonly encountered in the fields of autonomous driving and robotics. Although optical distortion can be addressed with image rectification [3], incorporating time-dependency of the projection function in the linearization framework is highly non-trivial.

To illustrate the impact of rolling shutter on various reconstruction methods, in Fig. 7 we use the synthetic dataset provided by Moenne-Loccoz et al. [33] where the motion of the camera and the shutter time are provided.

6.2. Secondary rays and lighting effects

Aligning the representation with 3DGRT [33]. The rendering formulations of 3DGS and 3DGRT mainly differ in terms of (i) determining which particles contribute to which pixels, (ii) the order of particles evaluation, (iii) the computation of the particles response. In Secs. 4.2 and 4.3 our goal was to reduce these differences to arrive to a common 3D representation that can be both rasterized and traced. Fig. 8 shows the comparison of 3D representations trained with different methods and evaluated with 3DGRT [33]. While some discrepancies naturally remain, *Ours (sorted)* achieves much better alignment to 3DGRT than StopThePop or 3DGS.

[3]Image rectification is generally effective only for low-FoV cameras and results in information loss, as shown in Tab. 3

Secondary rays. Aligning our rendering formulation to 3DGRT [33] enables us to perform hybrid rendering by rasterizing the primary and tracing the secondary rays within the same representations. Specifically, we first compute all the primary rays intersections with the scene, then we can have these primary rays rendered using our splatting method by simply discarding Gaussian hits falling behind a ray's closest intersection. Next, we compute secondary rays and trace them using 3DGRT. This hybrid rendering method allows us to achieve most of the complex visual effects (such as reflections and refractions) that would otherwise only be possible with ray tracing.

7. Discussion

We proposed a simple idea to replace the linearization of the non-linear projection function in 3DGS [17] with the Unscented Transform. This modification enables us to seamlessly generalize 3DGS to distorted cameras, support time-dependent effects such as rolling shutter, and align our rendering formulation with 3DGRT [33]. The latter enables us to perform hybrid rendering and unlock secondary rays for lighting effects.

Limitations and Future Work. Our method is significantly more efficient than ray-tracing-based methods [6, 29, 33], but it is still marginally slower than [17] (see details in Tab. 2). While UT is faster than linearization, it cannot fully offset the added complexity of 3D particle evaluation. Additionally, although UT permits exact projection of sigma points under arbitrary distortions, the resulting projected shape deviates from a 2D Gaussian in case of large distortions. This degrades the approximation of which particles contribute to which pixels. Finally, as our method

still uses a single point to evaluate each primitive, it is currently unable to render overlapping Gaussians accurately. Approaches such as EVER [29] may offer promising directions for addressing this limitation. Looking ahead, we hope that this work could inspire new research, particularly in fields like autonomous driving and robotics, where training and rendering with distorted cameras is essential. Our alignment with 3DGRT [33] also opens interesting opportunities for future research in inverse rendering and relighting.

8. Acknowledgements

We thank our colleagues Riccardo De Lutio, Or Perel, and Nicholas Sharp for their help in setting up experiments and for their valuable insights that helped us improve this work.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 5, 6

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. 3, 6

[3] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, 2022. 2

[4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[5] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *The Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2

[6] Jorge Condor, Sebastien Speierer, Lukas Bode, Aljaz Bozic, Simon Green, Piotr Didyk, and Adrian Jarabo. Don't Splat your Gaussians: Volumetric Ray-Traced Primitives for Modeling and Rendering Scattering and Emissive Media, 2024. 2, 3, 5, 8

[7] Daniel Duckworth, Peter Hedman, Christian Reiser, Peter Zhizhin, Jean-François Thibert, Mario Lučić, Richard Szeliski, and Jonathan T. Barron. Smerf: Streamable memory efficient radiance fields for real-time large-scene exploration, 2023. 2

[8] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021. 2

[9] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*, 2024. 2

[10] Antoine Guédon and Vincent Lepetit. Gaussian frosting: Editable complex radiance fields with real-time rendering. *ECCV*, 2024. 2

[11] Fredrik Gustafsson and Gustaf Hendeby. Some relations between extended and unscented kalman filters. *IEEE Transactions on Signal Processing*, 60(2):545–555, 2012. 2

[12] Florian Hahlbohm, Fabian Friederichs, Tim Weyrich, Linus Franke, Moritz Kappel, Susana Castillo, Marc Stamminger, Martin Eisemann, and Marcus Magnor. Efficient perspective-correct 3d gaussian splatting using hybrid transparency, 2024. 5

[13] Letian Huang, Jiayang Bai, Jie Guo, Yuanqi Li, and Yanwen Guo. On the error analysis of 3d gaussian splatting and an optimal projection strategy. *arXiv preprint arXiv:2402.00752*, 2024. 2, 3, 4

[14] Faris Janjoš, Lars Rosenbaum, Maxim Dolgov, and J. Marius Zöllner. Unscented autoencoder, 2023. 3

[15] Simon J. Julier and Jeffrey K. Uhlmann. New extension of the kalman filter to nonlinear systems. In *Defense, Security, and Sensing*, 1997. 2, 3

[16] Simon J Julier, Jeffrey K Uhlmann, and Hugh F Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of 1995 American Control Conference-ACC'95*, pages 1628–1632. IEEE, 1995. 3

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

[18] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics*, 43(4), 2024. 2

[19] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Jeff Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *arXiv preprint arXiv:2404.09591*, 2024. 2, 5

[20] 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. 5, 6

[21] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 40(4), 2021. 1

[22] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1440–1449, 2021. 2

[23] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21719–21728, 2024. 2

[24] Zimu Liao, Siyan Chen, Rong Fu, Yi Wang, Zhongling Su, Hao Luo, Linning Xu, Bo Dai, Hengjie Li, Zhilin Pei, et al. Fisheye-gs: Lightweight and extensible gaussian splatting module for fisheye cameras. *arXiv preprint arXiv:2409.04751*, 2024. 2, 4, 6, 7

[25] Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, and Wenming Yang. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *CVPR*, 2024. 2

[26] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 2

[27] Yang Liu, He Guan, Chuanchen Luo, Lue Fan, Junran Peng, and Zhaoxiang Zhang. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians, 2024. 2

[28] 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, 5

[29] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T. Barron, and Yinda Zhang. Ever: Exact volumetric ellipsoid rendering for real-time view synthesis, 2024. 2, 3, 5, 6, 8, 9

[30] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Francisco Vicente Carrasco, Markus Steinberger, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources, 2024. 2

[31] Marilena Maule, João Comba, Rafael Torchelsen, and Rui Bastos. Hybrid transparency. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, page 103–118, New York, NY, USA, 2013. Association for Computing Machinery. 5

[32] 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. 1, 2

[33] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics and SIGGRAPH Asia*, 2024. 2, 3, 4, 5, 6, 7, 8, 9

[34] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 1, 2

[35] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: A general purpose ray tracing engine. *ACM Trans. Graph.*, 29(4), 2010. 5

[36] 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*, 4 (43), 2024. 2, 4, 5, 6

[37] Christian Reiser, Richard Szeliski, Dor Verbin, Pratul P. Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T. Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *SIGGRAPH*, 2023. 2

[38] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *European Conference on Computer Vision*, 2020. 1

[39] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)*, 41(4):1–14, 2022. 1

[40] Marco Salvi and Karthikeyan Vaidyanathan. Multi-layer alpha blending. *Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2014. 5

[41] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2

[42] Otto Seiskari, Jerry Ylilammi, Valtteri Kaatrasalo, Pekka Rantalankila, Matias Turkulainen, Juho Kannala, and Arno Solin. Gaussian splatting on the move: Blur and rolling shutter compensation for natural camera motion, 2024. 2

[43] Gopal Sharma, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Volumetric rendering with baked quadrature fields. *arXiv preprint arXiv:2312.02202*, 2023. 2

[44] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2

[45] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5, 7

[46] Haithem Turki, Vasu Agrawal, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Deva Ramanan, Michael Zollhöfer, and Christian Richardt. Hybridnerf: Efficient neural rendering via adaptive volumetric surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19647–19656, 2024. 2

[47] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000. 3, 4

[48] Ziyu Wan, Christian Richardt, Aljaž Božič, Chao Li, Vijay Rengarajan, Seonghyeon Nam, Xiaoyu Xiang, Tuotuo Li, Bo Zhu, Rakesh Ranjan, et al. Learning neural duplex radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8307–8316, 2023. 2

[49] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 1

[50] Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic. Adaptive shells for efficient neural radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(6):1–15, 2023. 2

[51] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 1

[52] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv*, 2023. 1, 2

[53] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details for 3d gaussian splatting, 2024. 2

[54] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 5, 7

[55] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19447–19456, 2024. 5

[56] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 1, 3