Quadratic Gaussian Splatting: High Quality Surface Reconstruction with Second-order Geometric Primitives

Ziyu Zhang[1,2,†], Binbin Huang[3,†], Hanqing Jiang[4], Liyang Zhou[4], Xiaojun Xiang[4], Shuhan Shen[1,2,*]

[1]CASIA [2]UCAS [3]The University of Hong Kong [4]SenseTime Research

https://quadraticgs.github.io/QGS

Abstract

We propose Quadratic Gaussian Splatting (QGS), a novel representation that replaces static primitives with deformable quadric surfaces (e.g., ellipse, paraboloids) to capture intricate geometry. Unlike prior works that rely on Euclidean distance for primitive density modeling—a metric misaligned with surface geometry under deformation—QGS introduces geodesic distance-based density distributions. This innovation ensures that density weights adapt intrinsically to the primitive curvature, preserving consistency during shape changes (e.g., from planar disks to curved paraboloids). By solving geodesic distances in closed form on quadric surfaces, QGS enables surface-aware splatting, where a single primitive can represent complex curvature that previously required dozens of planar surfels, potentially reducing memory usage while maintaining efficient rendering via fast ray-quadric intersection. Experiments on DTU, Tanks and Temples, and Mip-NeRF360 datasets demonstrate state-of-the-art surface reconstruction, with QGS reducing geometric error (chamfer distance) by 33% over 2DGS and 27% over GOF on the DTU dataset. Crucially, QGS retains competitive appearance quality, bridging the gap between geometric precision and visual fidelity for applications like robotics and immersive reality.

1. Introduction

The pursuit of photorealistic view synthesis and geometry reconstruction is a central focus in academia and industry, with applications spanning virtual reality, film-making, and autonomous driving. Recent point-based methods, such as 3D Gaussians [21, 49], have proven highly effective, delivering view synthesis quality comparable to Neural Radiance Fields (NeRF)-based approaches while preserving superior geometry [16, 45] and offering significantly faster performance. Since then, these methods have been extended to dynamic reconstruction [10, 17, 27], scene editing [7, 39, 40, 53], and large-scale scene reconstruction [26, 28, 29, 43].

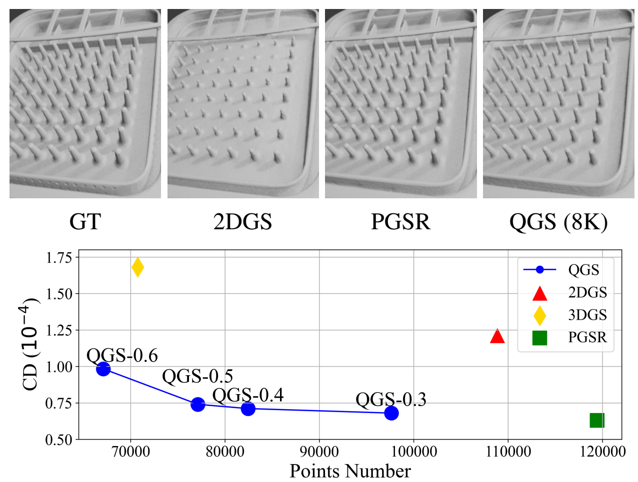


Figure 1. Our QGS model employs quadric surfaces to recover high-precision geometry from multi-view RGB images, reducing point count while enhancing accuracy. To isolate geometry from texture, we evaluate QGS on a weakly textured, geometry-rich subset of the Google Scanned Object dataset [9]. The results suggest that QGS, under different gradient threshold settings (QGS-x), consistently outperforms 2DGS [16] and achieves comparable quality to PGSR [5] with fewer points.

However, due to their approximation-based nature, 3D Gaussian methods often face challenges like high memory consumption, blurriness, and geometric inaccuracies. To address these limitations, recent efforts have focused on "primitive-level modifications." Methods like GES [14] generalize density distributions to reduce point counts, while others employ sharp kernels (e.g., clipped Gaussian [23], radial [18], or convex kernels [15]) to capture high-frequency appearance details. Assigning texture maps to planar primitives has also enabled more efficient scene representations [4, 34, 36, 38, 42]. Despite these advances, such modifications mainly target *high-quality appearance reconstruction*, leaving *detailed geometry reconstruction* largely unaddressed. Several approaches adopt alternative kernel shapes, such as flat ellipsoids or planar disks (*e.g.*, surfels) [8, 16, 45] to improve surface alignment. Yet, as a first-order geometric approximation, planar disks lack sufficient expressiveness; For example, highly curved surfaces requires dense primitive sampling, creating memory bottlenecks and rendering latency.

In this paper, we extend the representations of ellipsoid [21] and elliptical disks to a more general quadratic framework. Specifically, we define primitive boundaries using an implicit function $f(x, y, z) = 0$ where $f(x, y, z) < 0$ for points inside the primitive and $f(x, y, z) > 0$ otherwise. By adopting a quadratic form $f(x, y, z) = (x, y, z, 1)^\top \mathbf{Q}(x, y, z, 1) = 0$, where $\mathbf{Q} \in \mathbb{R}^{4\times4}$ is a conic matrix, we allow the representation of various shapes, including cylinder , ellipsoids, paraboloids, and hyperboloids. As a result, it generalizes 3D Gaussian primitives [21] and includes 2D Gaussian splatting [16] as a special case, allowing higher-order surface adaptation while incorporating kernel-level innovations such as shape kernels [18], texture billboards [34, 38, 42], and weight distributions [14]. In particular, there is an efficient ray-quadratic intersection algorithm [35] that makes this representation practical and versatile in the context of splatting. Since the primitive is essentially a second-order primitive, we term our method Quadratic Gaussian Splatting (QGS).

However, for deformable quadratic primitives, defining a continuous density distribution is a key challenge. Unlike methods using rigid primitives (e.g., triangles, spheres) with Euclidean-based density, our deformable quadratic surfaces require a geometry-aware density metric. For instance, as Fig 4 shows, when a primitive deforms from a disk to a paraboloid, Euclidean distance would measure straight-line distance, which ignores the curved surface. While Geodesic distance follows the surface, yields more accurate density modeling, it is generally difficult to compute.

Luckily, we target at surface modeling so we constrain our quadratic representation to paraboloids—including planar ellipses as degenerate cases—where efficient closed-form geodesic solvers exist. This formulation maintains point-based rendering efficiency while enabling adaptive curvature modeling, where a single paraboloid approximates complex geometry that otherwise requires multiple planar surfels, achieving higher fidelity with similar primitive counts. With such surfel-based representation and inspired by 2DGS [16], we adopt a ray-splat intersection [35] for splatting, maintaining multi-view consistency and perspective correctness. The surfel nature also allow QGS to leverage advanced splatting techniques that address perspective distortion [16] and popping effects [33], or advanced surface regularization [2, 11, 16] that improves overall geometry accuracy, albeit with added computational cost.

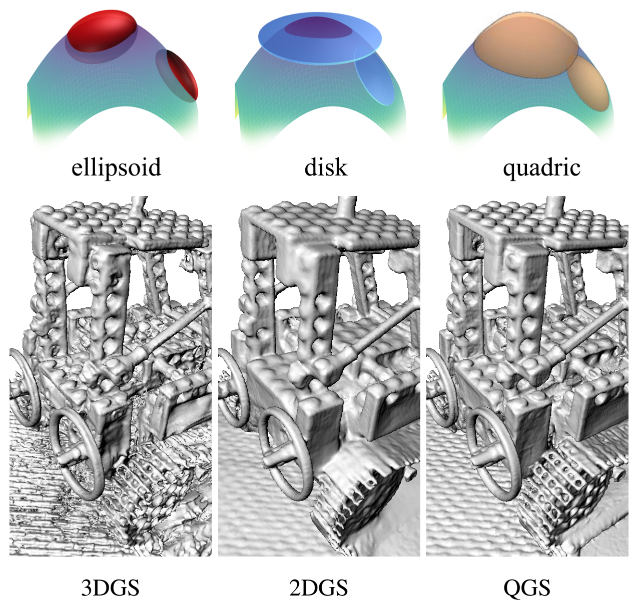


Figure 2. Demo and mesh reconstruction comparison on the Mip-NeRF 360 dataset. ellipsoids struggle to align accurately with surfaces, producing coarse and incomplete meshes. Planes fail to capture high-curvature regions, resulting in oversmoothed reconstructions. In contrast, quadric surfaces effectively capture geometric edge regions, achieving detailed reconstruction results.

To validate the effectiveness, we evaluate QGS on three benchmark datasets—MipNeRF360 [2], DTU [19], and Tanks and Temples [22]—and demonstrate state-of-the-art surface reconstruction accuracy. Our ablation studies reveal that deformable primitives improve geometric fidelity by 33% over 2DGS [16], establishing QGS as a robust alternative for applications demanding higher geometric precision. We summarize our key contributions as follows:

- A quadric surfel representation that generalizes 2D Gaussians to higher-order surfaces (e.g., paraboloids), enabling adaptive curvature modeling and capturing high frequency details.
- Geodesic-aware density distributions, the first method to unify deformable primitives with surface-aligned density modeling via closed-form geodesic solvers.
- State-of-the-art reconstruction: QGS reduces geometric error by 33% over 2DGS and 27% over GOF on DTU [19] while maintaining competitive training and rendering speeds.

2. Related work

2.1. Novel View Synthesis

Novel view synthesis has advanced rapidly, particularly since NeRF [31]. NeRF represents scene geometry and view-dependent appearance using a multi-layer perceptron (MLP) and optimizes it end-to-end through volumetric rendering to generate realistic images. Subsequent methods have introduced various enhancements. Mip-NeRF [1], Mip-NeRF 360 [2], and Zip-NeRF [3] addressed aliasing issues by refining sampling strategies, while I-NGP [32] and DVGO [37] significantly improved training and rendering speeds by replacing MLPs with feature grids.

More recently, 3DGS [21] has gained attention for its superior rendering quality and real-time performance, often surpassing NeRF [30, 33, 48, 53]. Mip-Splatting [48] introduced a smoothing filter to reduce high-frequency artifacts, while StopThePop [33] improved multi-view consistency through per-tile and per-pixel sorting. Scaffold-GS [30] optimized Gaussian distributions by reducing redundancy and improving rendering quality through anchor-based 3D Gaussian placement.

2.2. Neural Surface Reconstruction

Neural volumetric rendering methods often produce smoother and more complete reconstructions than traditional approaches. NeuS [41] and VolSDF [44] introduced signed distance fields (SDF) to model scene geometry within the NeRF framework [31]. Geo-NeuS [11] and NeuralWarp [12] further improved reconstruction by enforcing multi-view consistency. Neuralangelo [24] reduced reliance on MLPs by incorporating feature grids and numerical gradients, enhancing geometric fidelity.

However, neural rendering-based surface reconstruction often takes hours to converge using only image supervision. Additionally, the structured implicit representations make direct access and editing of scene geometry challenging.

2.3. Gaussian Splatting Surface Reconstruction

With the rapid advancement of 3DGS across various domains, Gaussian splatting methods have also significantly improved surface reconstruction. SuGaR [13] introduced regularization terms to encourage Gaussians to align with scene surfaces, constructing volumetric density fields from Gaussian ellipsoids and applying Poisson reconstruction [20] to extract meshes. GSDF [47] and NeuSG [6] combined 3D Gaussian ellipsoids with signed distance fields (SDFs), achieving high-quality geometric reconstructions and rendering results. Other approaches have leveraged ray-Gaussian intersections for improved accuracy. RadeGS [50], GOF [49], and PGSR [5] computed ray-ellipsoid intersections to obtain unbiased depth estimates and applied normal consistency supervision from 2DGS [16] to achieve state-of-the-art reconstruction. 2DGS further refined this approach by flattening Gaussian ellipsoids into disks, better aligning primitives with surfaces while introducing additional regularization losses for enhanced geometric constraints. MVG-splatting [25] extended 2DGS by incorporating multi-view consistency constraints, enabling more complex surface reconstructions. In this work, we propose Quadratic Gaussian Splatting (QGS), a novel surface representation based on quadric surfaces designed to improve the geometric fitting of primitives. We establish Gaussian distributions on quadrics, enabling end-to-end optimization. Additionally, by integrating multi-view consistency supervision, QGS achieves state-of-the-art geometric reconstruction and high-quality rendering.

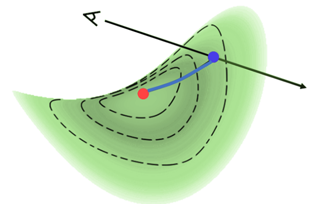


Figure 3. Illustration of a Gaussian on a quadric surface. The blue dot marks the Gaussian centroid, the red dot indicates the ray-splat intersection, and the red line represents the geodesic.

3. Method

3.1. Preliminary

Kerbl et al. [21] proposed representing a scene using 3D Gaussian ellipsoids as primitives and render images using differentiable volume splatting.

$$C(\mathbf{p}) = \sum_{i=0}^{N-1} G_i^{2D}(\mathbf{p})\alpha_i c_i \prod_{j=0}^{i-1}(1 - G_j^{2D}(\mathbf{p})\alpha_j) \quad (1)$$

Here, α_i represents the opacity, c_i denotes the color of each Gaussian primitive modeled by spherical harmonics, which is also used in our method, and $\mathbf{p}$ is the pixel coordinate. Finally, each Gaussian primitive is optimized by minimizing the photometric loss.

GS mesh reconstruction. Vanilla 3DGS [21] can render high-quality images, but it yields suboptimal results for scene geometry reconstruction. In subsequent surface reconstruction methods, volumetric approaches like GOF [49] and RadeGS [50] employ ray-splat intersection techniques, achieving SOTA reconstruction quality but limiting the consistency of normal and depth. In contrast, 2DGS [16] defines the 2D Gaussian distribution within a planar disk, which inherently provides consistent normals and depths across multiple views. But the disk is only a first-order approximation of the surface, which often leads to overly smooth reconstruction results in 2DGS, as shown in Fig 2.

3.2. Quadratic Gaussian Splatting

To enhance the geometric fitting capability of the surface representation, we present differentiable Quadratic Gaussian Splatting, as shown in Fig 1. We will first introduce the Quadric Gaussian Model, then discuss the splatting design for quadrics, and finally explain the optimization process.

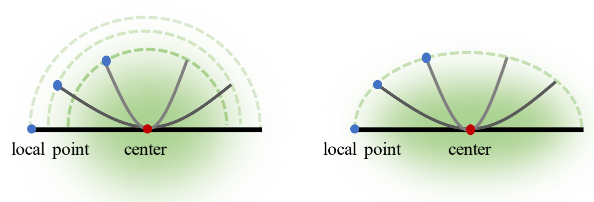


Figure 4. Density distribution among deformable primitives (disk→paraboloid). (a) Euclidean distance fails to adapt to curvature during primitive deformation, distorting density distributions (uneven weight in the blue point). (b) Our geodesic distance follows intrinsic curvature, maintaining spatial coherence across deformation and preventing uneven splatting artifacts.

Quadratic Gaussian Model. Given a homogeneous coordinate $\mathbf{x} = [x, y, z, 1]^T \in \mathbb{R}^4$, a quadric surface can be defined as the solution set to the following equation:

$$\begin{aligned} f(x,y,z) &= Ax^2 + 2Bxy + 2Cxz + 2Dx + Ey^2 \\ &+ 2Fyz + 2Gy + Hz^2 + 2Iz + J \\ &= \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \\ D & G & I & J \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\ &= \mathbf{x}^T \mathbf{Q} \mathbf{x} = 0 \end{aligned} \tag{2}$$

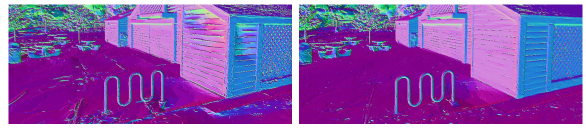
We apply a congruent transformation to obtain the transformation from object space to parameter space.

$$\mathbf{Q} = \mathbf{T}^{-T} \mathbf{D} \mathbf{T}^{-1}, \text{with } \mathbf{D} \text{ symmetrical} \qquad \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{c} \\ 0 & 1 \end{bmatrix} \tag{3}$$

Here, $\mathbf{c}$ denotes the position of the quadric, and $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ denotes the orientation of the quadric in the object space. The matrix $\mathbf{D}$ defines the surface scale and shape: $\mathbf{D} = \text{diag}(s_1^2, s_2^2, s_3^2, 1)$ yields an ellipsoid, while $\mathbf{D} = \text{diag}(s_1^2, s_2^2, 0, 0)$ produces a plane.

To compute the Gaussian weight at any surface point, we first define a measure on the surface to establish the Gaussian distribution. A straightforward approach is to construct the Gaussian distribution using Euclidean distance. However, this leads to inconsistencies when the surface undergoes deformation, resulting in uneven splatting reconstruction artifacts, as shown in Fig. 5. To address this, we use geodesic distance to ensure that weights adapt consistently to surface deformations. Fig. 4 shows a 1D illustration.

However, not all geodesics have closed-form solutions. When $\mathbf{Q}$ is an ellipsoid or hyperboloid, computing the geodesic length typically requires numerical methods. Luckily, as we aim at surface reconstruction, we focus on paraboloid with the following form:



(a) w/ Euclid (b) w/ Geodesic

Figure 5. Comparison between Euclidean and geodesic distances. The first row presents schematic illustrations, while the second row shows normal comparisons. The left column uses Euclidean distance, while the right column employs geodesic distance.

$$\begin{aligned} f(x,y,z) &= \mathbf{x}^T \begin{bmatrix} \mathbf{R} & \mathbf{c} \\ 0 & 1 \end{bmatrix}^{-T} \mathbf{D} \begin{bmatrix} \mathbf{R} & \mathbf{c} \\ 0 & 1 \end{bmatrix}^{-1} \mathbf{x} \\ &= \hat{\mathbf{x}}^T \begin{bmatrix} \frac{d_{11}}{s_1^2} & 0 & 0 & 0 \\ 0 & \frac{d_{22}}{s_2^2} & 0 & 0 \\ 0 & 0 & 0 & -\frac{d_{33}}{2s_3} \\ 0 & 0 & -\frac{d_{33}}{2s_3} & 0 \end{bmatrix} \hat{\mathbf{x}} \\ &= \frac{d_{11}}{s_1^2}\hat{x}^2 + \frac{d_{22}}{s_2^2}\hat{y}^2 - \frac{d_{33}}{s_3}\hat{z} = 0 \end{aligned} \tag{4}$$

Here and henceforth, we use $\hat{}$ to denote Gaussian local coordinate. $d_{ii} \in \{0, \pm 1\}$ determines whether the paraboloid is elliptic, hyperbolic, or planar. However, since d_{ii} is discrete, the primitive cannot transition smoothly between elliptic and hyperbolic paraboloids. To resolve this, we introduce a signed scale for a differentiable transition between paraboloid types.

$$f(\hat{x}, \hat{y}, \hat{z}) = \frac{\text{sign}(s_1)}{s_1^2}\hat{x}^2 + \frac{\text{sign}(s_2)}{s_2^2}\hat{y}^2 - \frac{1}{s_3}\hat{z} = 0 \tag{5}$$

In vanilla 3DGS [21], the scale is obtained through the exp activation function, i.e., $s(x) = \exp(x)$. To introduce a sign, we add another variable t to control the sign, i.e., $s(x,t) = \tanh(t)\exp(x)$.

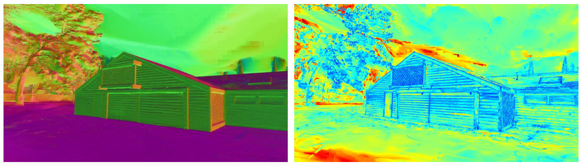
Gaussian Distribution on Quadric. We will now describe how to define a Gaussian distribution on a paraboloid using geodesic lines. First, paraboloid (Equation 5) could be expressed in explicit form:

$$\hat{z}(\hat{x}, \hat{y}) = s_3(\frac{\text{sign}(s_1)}{s_1^2}\hat{x}^2 + \frac{\text{sign}(s_2)}{s_2^2}\hat{y}^2) \tag{6}$$

We convert it isometrically into cylindrical coordinates, i.e. $\hat{x} = \rho\cos\theta, \hat{y} = \rho\sin\theta$. And we rewrite Equation 6 as:

$$\begin{aligned} \hat{z}(\theta, \rho) &= s_3(\frac{\text{sign}(s_1)\cos^2\theta}{s_1^2} + \frac{\text{sign}(s_2)\sin^2\theta}{s_2^2})\rho^2 \\ &= a(\theta)\rho^2 \end{aligned} \tag{7}$$

Due to the paraboloid's symmetry, for any point $\hat{\mathbf{p}}_0 = (\rho_0, \theta_0, \hat{z}(\theta_0, \rho_0))$, the intersection line of the plane $\theta = \theta_0$



(a) normal (b) curvature

Figure 6. The (a) normal map and (b) curvature map of a scene. In the curvature map, blue indicates higher curvature and curved areas, while red indicates lower curvature and flatter areas.

with the paraboloid: $\hat{z}(\theta_0, \rho), \rho \in (0, \rho_0)$, is the geodesic to the origin. Then the geodesic distance is the arc length l of this curve, as shown in Fig 3.

$$\begin{aligned} l(a, \rho_0) &= \int_0^{\rho_0} \sqrt{1 + (2at)^2} \mathrm{d}t \\ &= \frac{\ln(\sqrt{u^2+1} + u) + u\sqrt{u^2+1}}{4a} \\ &\text{where } u = 2a\rho_0 \end{aligned} \tag{8}$$

For the derivation of the integral, please refer to the supplementary materials. We then define the mean of the 2D Gaussian distribution on the surface at the origin of the quadric surface, with (s_1, s_2) representing the principal axis variances of the Gaussian. Since the contours of the 2D Gaussian distribution form an ellipse:

$$\frac{\rho^2 \cos^2 \theta}{s_1^2} + \frac{\rho^2 \sin^2 \theta}{s_2^2} = 1 \tag{9}$$

Given a point (θ_0, ρ_0) on the ellipse, ρ_0 represents the standard deviation of the 2D Gaussian distribution in the θ_0 direction.

$$\sigma(\theta_0) = \rho_0 = \frac{s_1 s_2}{\sqrt{(s_2 \cos\theta_0)^2 + (s_1 \sin\theta_0)^2}} \tag{10}$$

Thus, for any point $\hat{\mathbf{p}}_0$ on the surface, we can define the corresponding Gaussian function value as:

$$G(\hat{\mathbf{p}}_0(\theta_0, \rho_0)) = \exp\left(-\frac{(l(a(\theta_0), \rho_0))^2}{2(\sigma(\theta_0))^2}\right) \tag{11}$$

Notably, when $|s_3| \to 0$, the paraboloid becomes equivalent to a disk. Furthermore, as $x \to 0$, we have $\sqrt{1+x} \to 1$ and $\ln(1+x) \sim x$. Thus, as $|s_3| \to 0$, we get $a \to 0$ and $l \to \rho_0$ by Equation 8, meaning the geodesic distance becomes equivalent to the Euclidean distance. This indicates that 2DGS can be regarded as a specific degenerate case of QGS, whose more generalized nature allows it to fit high-curvature regions effectively.

3.2.1 Splatting

Although Sigg et al. [35] derived the Ray-Quadric Intersection, QGS integrates Gaussian distribution, necessitating a redefinition of the intersection for multi-view consistency.

Ray-splat Intersection. Let the camera center in the Gaussian local space be denoted as $\hat{\mathbf{o}} \in \mathbb{R}^{3\times 1}$ and the ray direction as $\hat{\mathbf{d}} \in \mathbb{R}^{3\times 1}$. A point on the ray can be defined as $\hat{\mathbf{p}} = \hat{\mathbf{o}} + t\hat{\mathbf{d}}$. By substituting $\hat{\mathbf{p}}$ into the Equation 6, we solve a quadratic equation to obtain two intersection points: the nearer point $\hat{\mathbf{p}}_n = (\hat{x}_n, \hat{y}_n, \hat{z}_n)$ and the farther point $\hat{\mathbf{p}}_f = (\hat{x}_f, \hat{y}_f, \hat{z}_f)$, with $t_n \leq t_f$. To ensure multi-view consistency in QGS, only one valid point of the two is selected. First, if the geodesic distance of $\hat{\mathbf{p}}_n$ is within $3\sigma(\theta_n)$, we choose $\hat{\mathbf{p}}_n$. If not, we check if $\hat{\mathbf{p}}_f$ is within $3\sigma(\theta_f)$. If so, we select $\hat{\mathbf{p}}_f$. If neither condition is met, we assume no intersection between the ray $\hat{\mathbf{p}}(t)$ and the primitive. This assumption is generally valid because the significant weight typically causes the first point to occlude the second point. The derivation is provided in the supplementary material.

Normal and Curvature. Similar to 2DGS [16], QGS is a surface-based representation that naturally possesses multi-view consistent geometric properties, making it straightforward to compute surface normals. Given any point $\hat{\mathbf{p}}_0 = (\hat{x}_0, \hat{y}_0, \hat{z}(\hat{x}_0, \hat{y}_0))$ on the surface, we can take the partial derivatives of the Equation 5, yielding:

$$\hat{\mathbf{n}}_0(\hat{\mathbf{p}}_0) = \left(\frac{2\mathrm{sign}(s_1)}{s_1^2}\hat{x}_0, \frac{2\mathrm{sign}(s_2)}{s_2^2}\hat{y}_0, -\frac{1}{s_3}\right) \tag{12}$$

As QGS provides a second-order fit, it naturally outputs second-order geometric information like curvature, which describes surface bending. For each QGS primitive, the Gaussian curvature at the ray-splat intersection can be computed analytically, with detailed derivation available in the supplementary material. Let $\lambda_1 = \mathrm{sign}(s_1) \cdot s_3/s_1^2$ and $\lambda_2 = \mathrm{sign}(s_2) \cdot s_3/s_2^2$. The curvature at point $\hat{\mathbf{p}}_0$ is:

$$\hat{K}_0(\hat{\mathbf{p}}_0) = \frac{4\lambda_1\lambda_2}{(1 + 4\lambda_1^2\hat{x}_0^2 + 4\lambda_2^2\hat{y}_0^2)^2} \tag{13}$$

We render the normal map $\mathbf{N}$ and curvature map $\mathbf{K}$ by Equation 14 for a given viewpoint using alpha-blending, as shown in the Fig 6.

$$\{\mathbf{N}, \mathbf{K}\} = \sum_{i=0}^{N-1} G_i \alpha_i \prod_{j=0}^{i-1} (1 - G_j \alpha_j) \{\mathbf{n}_i, K_i\} \tag{14}$$

Per-pixel Resorting. We adopt the per-pixel resorting method from Stopthepop to address the depth pop-out issue, as illustrated in Fig 7. Implementation and derivation details are provided in the supplementary material.

3.3. Optimization

We now introduce the our training objectives during optimization.

(a) w/o resorting (b) w/ resorting

Figure 7. Comparison of depth normals with and without resorting. Centroid-only sorting (a) introduces stripe artifacts, while resorting (b) produces smoother and more consistent normals.

Depth Distortion. We adopt the depth distortion loss and normal consistency loss proposed by 2DGS [16].

$$\mathcal{L}_d = \sum_{i=0}^{N-1} \sum_{j=0}^{i-1} \omega_i \omega_j (t_i - t_j)^2 \tag{15}$$

Here, $\omega_i = \bar{\alpha}_i T_i$ denotes the alpha-blending weight of the i-th Gaussian, and t_i represents the depth at the ray-splat intersection. i, j index the Gaussians along the ray.

Curvature Guided Normal Consistency. 2DGS [16] introduces the normal consistency loss to ensure that all primitives on a ray are locally aligned with the actual surface.

$$\mathcal{L}_n = \sum_i \omega_i (1 - \mathbf{n}_i^T \mathbf{N}) \tag{16}$$

Here, $\mathbf{n}_i$ represents the splat normal facing the camera, while $\mathbf{N}$ is computed by differentiating the depth point $\mathbf{p}$ from neighboring pixels. However, neighboring pixels may violate the local planar assumption, particularly in edge regions. Using normal consistency loss in edge areas can introduce errors, contributing to the over-smoothing observed in 2DGS. A straightforward approach is to approximate geometric edges using image edges, which can then guide normal consistency supervision [5, 25]. However, we found that, in most scenes, image edges do not fully correspond to geometric edges, especially in uniformly lit areas. Thus, we use the curvature map, which more accurately corresponds to geometric edges and is efficiently and uniquely generated by QGS, to guide normal supervision.

$$\begin{aligned} \lambda_K(K(u,v)) &= 1 - \text{sigmoid}(\ln(|K(u,v)| + \varepsilon) \\ \mathcal{L}_{Kn}(u,v) &= \lambda_K(K(u,v))\mathcal{L}_n(u,v) \end{aligned} \tag{17}$$

Multi-view regularization Loss. To ensure a fairer comparison with plane-based PGSR [5], we incorporate PGSR's multi-view regularization, computing warp and photometric consistency losses via homography transformations.

$$\begin{aligned} \mathbf{H}_{rn} =& \mathbf{K}_n(\mathbf{R}_{rn} + \frac{\mathbf{T}_{rn}\mathbf{n}_r^T}{\mathbf{n}_r^T \mathbf{X}_r})\mathbf{K}_r^{-1} \\ \mathcal{L}_{Mv} =& \frac{1}{V} \sum_{\mathbf{p}_r \in V} ||\mathbf{p}_r - \mathbf{H}_{nr}\mathbf{H}_{rn}\mathbf{p}_r|| + \\ & (1 - NCC(\mathbf{I}_r(\mathbf{p}_r), \mathbf{I}_n(\mathbf{H}_{rn}\mathbf{p}_r))) \end{aligned} \tag{18}$$

Here, $\mathbf{n}_r$ represents the normal map, and $\mathbf{X}_r$ denotes the 3D points projected from the depth map. $\mathbf{R}_{rn}$ represents the relative rotation, $\mathbf{K}$ is the camera intrinsic, $\mathbf{p}_r$ denotes the pixel coordinates, and $\mathbf{I}$ refers to the image intensity. This loss enforces multi-view constraints by computing the reprojection error from the depth and normal maps, along with normalized cross-correlation (NCC) [46].

Final Loss. Finally, we input a sparse point cloud and posed images to optimize QGS with the following loss function:

$$\mathcal{L} = \mathcal{L}_c + \lambda_d \mathcal{L}_d + \lambda_n \mathcal{L}_{Kn} + \lambda_{Mv} \mathcal{L}_{Mv} \tag{19}$$

4. Experiment

We compared our method with several SOTA approaches across multiple datasets, including DTU [19], TNT [22], and Mip-NeRF 360 [2]. We evaluated geometry with F1-score and Chamfer Distance, and appearance with PSNR, SSIM, and LPIPS, followed by analysis and conclusions.

4.1. Implementation Details.

Rasterizer. We implemented Quadratic Splatting with custom CUDA kernels on the 3DGS framework [21], extending the renderer to output depth distortion maps, depth maps, normal maps, and curvature maps. Since quadrics can be non-convex, we used rectangular truncation and approximations to compute image bounding boxes, as detailed in the supplementary materials.

Settings. We adopted the adaptive control strategy from 3DGS [21]. Similar to 2DGS [16], QGS projects the 3D center gradient onto screen space instead of using the 2D projected gradient. A gradient threshold of 0.3 and a percent dense value of 0.001 ensure consistent point cloud number with other methods. All experiments were conducted on a single A6000 GPU.

Mesh Extraction. We used median depth (i.e., $t_{\text{median}} = \max t_i | T_i > 0.5$) as the final output depth. Then we fused the depth maps using Truncated Signed Distance Fusion (TSDF) with Open3D [52]. Following the 2DGS setup, we set the voxel size to 0.004 and truncation threshold to 0.02.

4.2. Comparison.

Geometry Evaluation. In all experiments, evaluations were performed at half the image resolution. To clearly demonstrate the improvement introduced by our quadric surfels and to maintain consistency with other methods, we denote the variant of QGS without multi-view regularization loss as QGS w/o MV. In Table 1, we compared QGS with implicit [24, 41, 44] and explicit [5, 13, 16, 47, 49, 51] SOTA reconstruction methods on the DTU dataset using Chamfer Distance as the metric. As shown in Table 1, our method outperforms both implicit and explicit approaches on the DTU dataset. Furthermore, even without multi-view regularization, our method surpasses others in accu-

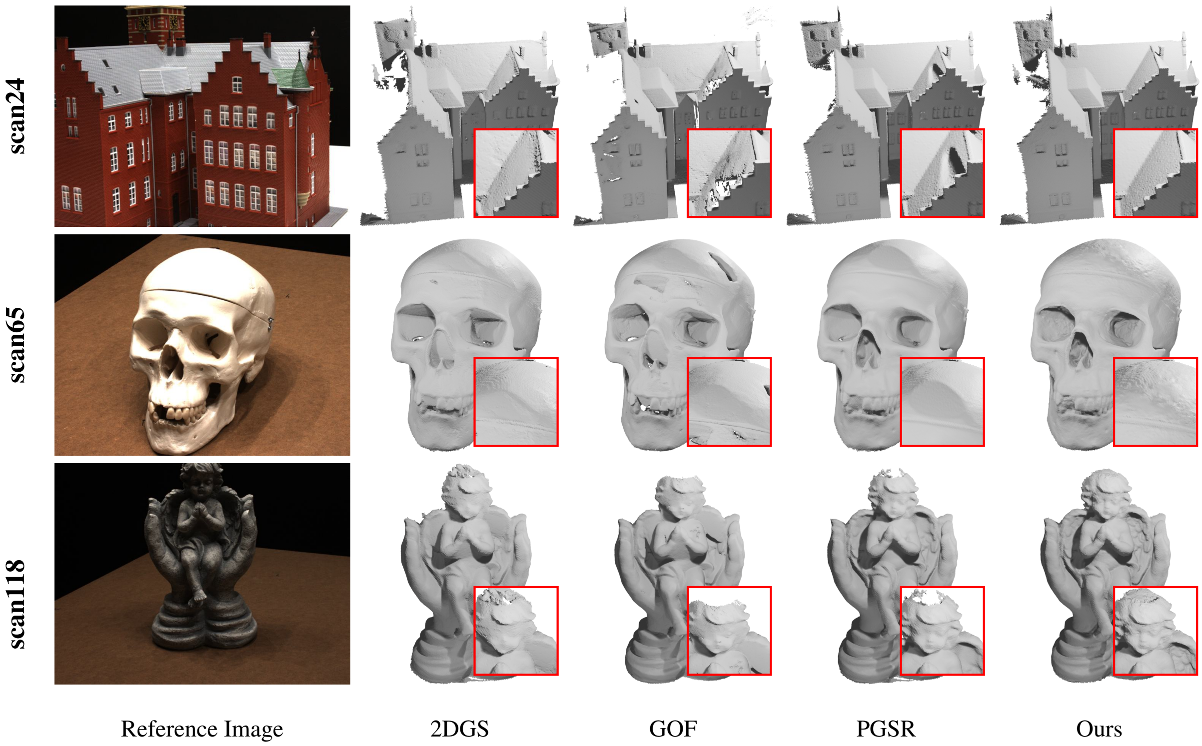


Figure 8. Qualitative geometric reconstruction comparisons on the DTU dataset. Our method achieves reconstructions of higher quality and greater detail.

CD (mm)↓	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean	Time
NeuS [41]	1.00	1.37	0.93	0.43	1.10	0.65	0.57	1.48	1.09	0.83	0.52	1.20	0.35	0.49	0.54	0.84	>12h
VolSDF [44]	1.14	1.26	0.81	0.49	1.25	0.70	0.72	1.29	1.18	0.70	0.66	1.08	0.42	0.61	0.55	0.86	>12h
Neuralangelo [24]	0.37	0.72	0.35	0.35	0.87	0.54	0.53	1.29	0.97	0.73	0.47	0.74	0.32	0.41	0.43	0.61	>128h
3DGS [21]	2.14	1.53	2.08	1.68	3.49	2.21	1.43	2.07	2.22	1.75	1.79	2.55	1.53	1.52	1.50	1.96	11.2min
Gaussian surfels [8]	0.66	0.93	0.54	0.41	1.06	1.14	0.85	1.29	1.53	0.79	0.82	1.58	0.45	0.66	0.53	0.88	6.7min
SuGaR[13]	1.47	1.33	1.13	0.61	2.25	1.71	1.15	1.63	1.62	1.07	0.79	2.45	0.98	0.88	0.79	1.33	1h
2DGS [16]	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	1.27	0.76	0.70	1.40	0.40	0.76	0.52	0.80	19.2min
GOF [49]	0.50	0.82	0.37	0.37	1.12	0.74	0.73	1.18	1.29	0.68	0.77	0.90	0.42	0.66	0.49	0.74	1h
GSDF [47]	0.59	0.94	0.46	0.38	1.30	0.77	0.73	1.59	1.29	0.76	0.59	1.22	0.38	0.52	0.51	0.80	32min
GS-pull [51]	0.51	0.56	0.46	0.39	0.82	0.67	0.85	1.37	1.25	0.73	0.54	1.39	0.35	0.88	0.42	0.75	22min
Ours w/o MV	0.46	0.76	0.40	0.38	0.92	0.80	0.76	1.25	0.95	0.67	0.62	1.20	0.38	0.60	0.47	0.71	25min
PGSR [5]	0.40	0.60	0.39	0.37	0.78	0.59	0.53	1.18	0.67	0.63	0.48	0.62	0.34	0.42	0.39	0.56	40min
Ours	0.38	0.62	0.37	0.38	0.75	0.55	0.51	1.12	0.68	0.61	0.46	0.58	0.35	0.41	0.40	0.54	48min

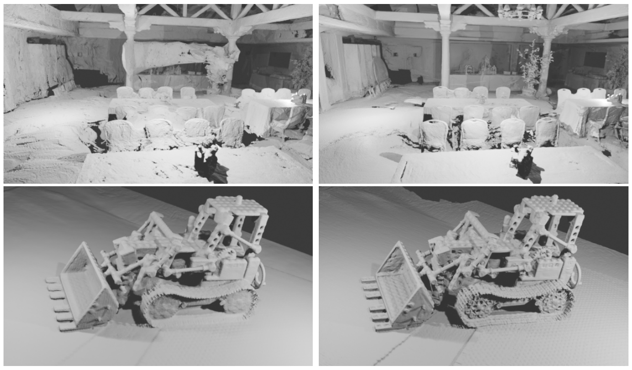
Table 1. Quantitative Chamfer Distance comparison on the DTU dataset. Our method achieves the best performance both with and without multi-view regularization. ■, ■, ■ denote the best, second best, and third best results, respectively.

F1-Score ↑	Geo-NeuS	N-angelo	2DGS	GOF	Ours w/o MV	PGSR	Ours
Barn	0.33	0.70	0.41	0.51	0.46	0.52	0.55
Caterpillar	0.26	0.36	0.24	0.41	0.32	0.38	0.40
Courthouse	0.12	0.28	0.16	0.28	0.26	0.26	0.28
Ignatius	0.72	0.89	0.52	0.68	0.79	0.77	0.81
Meetingroom	0.20	0.32	0.17	0.28	0.25	0.29	0.31
Truck	0.45	0.48	0.45	0.58	0.60	0.62	0.64
Mean	0.35	0.50	0.33	0.46	0.45	0.47	0.50
Time	>24h	>127h	34min	114min	43min	66min	75min

Table 2. Quantitative F1-Score comparison of QGS with GS-like and NeRF-like methods on the TNT dataset. QGS outperforms all methods, achieving state-of-the-art reconstruction results.

racy while maintaining competitive speed. This improvement stems from the superior geometric fitting capability of quadrics, which enables finer detail preservation, as illustrated in Fig 8.

In Table 2, we also evaluated QGS and other SOTA methods [5, 11, 16, 24, 49] on the TNT dataset using F1-score. Without multi-view regularization, our method significantly outperforms the disk-based 2DGS and achieves comparable results to GOF while requiring only half the computation time. With multi-view regularization, it surpasses all methods in reconstruction accuracy. Additionally, we provide a qualitative comparison between QGS without multi-view regularization and 2DGS to further demonstrate the enhanced geometric fitting capability of quadric surfels, as shown in Fig 9.



(a) 2DGS (b) QGS w/o MV

Figure 9. Mesh reconstruction comparison between 2DGS and QGS w/o MV. The first row shows the Meetingroom in TNT dataset, and the second row shows the kitchen scene in Mip-NeRF 360 dataset.

	Indoor scenes			Outdoor scenes		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF	26.84	0.790	0.370	21.46	0.458	0.515
Deep Blending	26.40	0.844	0.261	21.54	0.524	0.364
i-NGP	29.15	0.880	0.216	22.90	0.566	0.371
Mip-NeRF360	31.72	0.917	0.180	24.47	0.691	0.283
3DGS	30.52	0.921	0.199	24.45	0.728	0.240
SuGar	29.44	0.911	0.216	22.76	0.631	0.349
2DGS	30.39	0.924	0.182	24.33	0.709	0.284
GOF	30.80	0.928	0.167	24.76	0.742	0.225
Ours w/o MV	30.48	0.926	0.166	24.56	0.724	0.239
PGSR	30.35	0.924	0.176	24.29	0.718	0.236
Ours	30.45	0.919	0.184	24.32	0.706	0.242

Table 3. Quantitative comparison of appearance between QGS, GS-like, and NeRF-like methods on the Mip-NeRF 360 dataset.

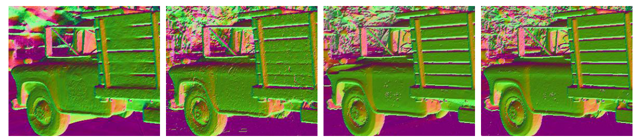
Rendering Evaluation. We compared rendering quality on the Mip-NeRF360 dataset with baseline approaches. As a surface-based method, QGS achieves competitive rendering results in indoor scenes, as shown in Table 3, but performs less effectively in outdoor environments. We attribute this to QGS's higher geometric fitting capability, which may lead to overfitting in regions with sparse views or low texture. Future work could explore additional regularization constraints, particularly on the curvature of quadric surfaces, to address this issue.

4.3. Ablation

In this section, we assess the impact of individual QGS components on reconstruction quality, including per-pixel resorting, curvature-guided normal consistency, and multi-view regularization, as presented in Table 4 and Fig 10. We denote w/o sort as the absence of per-pixel resorting, w/o λ_K as the absence of curvature-guided normal consistency, and w/o MV as the absence of multi-view regularization. We observe that: (a) Disabling curvature guidance fills fine gaps in the normal map, reducing reconstruction quality.

F1-Score ↑	B	C	CH	I	M	T	mean
Full model	0.55	0.40	0.28	0.81	0.31	0.64	0.50
w/o Sort	0.53	0.39	0.26	0.77	0.29	0.62	0.48
w/o λ_K	0.56	0.37	0.27	0.80	0.29	0.62	0.49
w/o MV	0.46	0.32	0.26	0.79	0.25	0.60	0.45

Table 4. Quantitative ablation study on the TNT dataset. We present ablation results for Barn (B), Caterpillar (C), Courthouse (CH), Ignatius (I), Meeting Room (M), and Truck (T). Our full setting achieves the best performance.



(a) w/o λ_K (b) w/o Sort (c) w/o MV (d) Full

Figure 10. Ablation study. The full setting achieves the highest reconstruction quality.

Setting	w/o MV	s_3 fixed	w/ Euclid
Mean F1↑	**0.45**	0.37	0.38

Table 5. Ablation study without multi-view regularization on the TNT dataset. Fixing the quadric as a disk or using Euclidean distance optimization both result in reduced reconstruction quality.

(b) The absence of per-pixel sorting similarly fills gaps and leads to uneven surfaces. (c) Disabling multi-view regularization causes overfitting in low-texture regions, resulting in surface artifacts, such as dents on the front of the truck.

Additionally, we conducted ablation studies without multi-view regularization to isolate and evaluate the effectiveness of quadrics. Specifically, we fixed the quadric's z-axis scale to 0.001, approximating it as a disk (denoted as s_3 fixed). We also evaluated optimization using Euclidean distance (denoted as w/ Euclid). Both modifications significantly reduced reconstruction quality, as shown in Table 5.

5. Conclusion

In this work, we introduce Quadratic Gaussian Splatting, a variant of GS-like methods, designed to reconstruct accurate scene geometry and recover finer details. QGS is the first to introduce quadric surfaces to Gaussian Splatting, defining Gaussian distributions in non-Euclidean space to improve fitting and capture second-order curvature. We achieve SOTA geometric reconstruction and competitive rendering results on various indoor and outdoor datasets. Additional results are in the supplementary materials.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (Grants U22B2055 and 62273345), the Beijing Natural Science Foundation (Grant L223003), and the Key R&D Project in Henan Province (Grant 231111210300).

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021. 3

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 2, 3, 6

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023. 3

[4] Brian Chao, Hung-Yu Tseng, Lorenzo Porzi, Chen Gao, Tuotuo Li, Qinbo Li, Ayush Saraf, Jia-Bin Huang, Johannes Kopf, Gordon Wetzstein, and Changil Kim. Textured gaussians for enhanced 3d scene appearance modeling. In *CVPR*, 2025. 1

[5] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *arXiv preprint arXiv:2406.06521*, 2024. 1, 3, 6, 7

[6] Hanlin Chen, Chen Li, and Gim Hee Lee. Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance. *arXiv preprint arXiv:2312.00846*, 2023. 3

[7] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21476–21485, 2024. 1

[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. 2, 7

[9] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022. 1

[10] Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, et al. Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. *arXiv preprint arXiv:2401.15318*, 2024. 1

[11] Qiancheng Fu, Qingshan Xu, Yew Soon Ong, and Wenbing Tao. Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 35:3403–3416, 2022. 2, 3, 7

[12] Josif Grabocka and Lars Schmidt-Thieme. Neuralwarp: Time-series similarity with warping networks. *arXiv preprint arXiv:1812.08306*, 2018. 3

[13] 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. 3, 6, 7

[14] 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, 2

[15] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3d convex splatting: Radiance field rendering with 3d smooth convexes. *arXiv preprint arXiv:2411.14974*, 2024. 1

[16] 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, 5, 6, 7

[17] Yiming Huang, Beilei Cui, Long Bai, Ziqi Guo, Mengya Xu, and Hongliang Ren. Endo-4dgs: Distilling depth ranking for endoscopic monocular scene reconstruction with 4d gaussian splatting. *arXiv preprint arXiv:2401.16416*, 2024. 1

[18] Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Deformable radial kernel splatting. *arXiv preprint arXiv:2412.11752*, 2024. 1, 2

[19] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014. 2, 6

[20] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006. 3

[21] 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, 6, 7

[22] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 2, 6

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

[24] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2023. 3, 6, 7

[25] Zhuoxiao Li, Shanliang Yao, Yijie Chu, Angel F Garcia-Fernandez, Yong Yue, Eng Gee Lim, and Xiaohui Zhu.

Mvg-splatting: Multi-view guided gaussian splatting with adaptive quantile-based geometric consistency densification. *arXiv preprint arXiv:2407.11840*, 2024. 3, 6

[26] Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, et al. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5166–5175, 2024. 1

[27] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21136–21145, 2024. 1

[28] Yang Liu, He Guan, Chuanchen Luo, Lue Fan, Junran Peng, and Zhaoxiang Zhang. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. *arXiv preprint arXiv:2404.01133*, 2024. 1

[29] Yang Liu, Chuanchen Luo, Zhongkai Mao, Junran Peng, and Zhaoxiang Zhang. Citygaussianv2: Efficient and geometrically accurate reconstruction for large-scale scenes. *arXiv preprint arXiv:2411.00771*, 2024. 1

[30] 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. 3

[31] 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, 3

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 3

[33] 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. 2, 3

[34] Victor Rong, Jingxiang Chen, Sherwin Bahmani, Kiriakos N Kutulakos, and David B Lindell. Gstex: Per-primitive texturing of 2d gaussian splatting for decoupled appearance and geometry modeling. *arXiv preprint arXiv:2409.12954*, 2024. 1, 2

[35] Christian Sigg, Tim Weyrich, Mario Botsch, and Markus H Gross. Gpu-based ray-casting of quadratic surfaces. In *PBG@ SIGGRAPH*, pages 59–65, 2006. 2, 5

[36] Yunzhou Song, Heguang Lin, Jiahui Lei, Lingjie Liu, and Kostas Daniilidis. Hdgs: Textured 2d gaussian splatting for enhanced scene rendering, 2024. 1

[37] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5459–5469, 2022. 3

[38] David Svitov, Pietro Morerio, Lourdes Agapito, and Alessio Del Bue. Billboard splatting (bbsplat): Learnable textured primitives for novel view synthesis. *arXiv preprint arXiv:2411.08508*, 2024. 1, 2

[39] Rafał Tobiasz, Grzegorz Wilczyński, Marcin Mazur, Sławomir Tadeja, and Przemysław Spurek. Meshsplats: Mesh-based rendering with gaussian splatting initialization. *arXiv preprint arXiv:2502.07754*, 2025. 1

[40] Joanna Waczyńska, Piotr Borycki, Sławomir Tadeja, Jacek Tabor, and Przemysław Spurek. Games: Mesh-based adapting and modification of gaussian splatting. *arXiv preprint arXiv:2402.01459*, 2024. 1

[41] 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. *arXiv preprint arXiv:2106.10689*, 2021. 3, 6, 7

[42] Sebastian Weiss and Derek Bradley. Gaussian billboards: Expressive 2d gaussian splatting with textures. *arXiv preprint arXiv:2412.12734*, 2024. 1, 2

[43] Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. Gaussiancity: Generative gaussian splatting for unbounded 3d city generation. *arXiv preprint arXiv:2406.06526*, 2024. 1

[44] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021. 3, 6, 7

[45] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Trans. Graph.*, 38 (6), 2019. 1, 2

[46] Jae-Chern Yoo and Tae Hee Han. Fast normalized cross-correlation. *Circuits, systems and signal processing*, 28:819–843, 2009. 6

[47] Mulin Yu, Tao Lu, Linning Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai. Gsdf: 3dgs meets sdf for improved rendering and reconstruction. *arXiv preprint arXiv:2403.16964*, 2024. 3, 6, 7

[48] 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*, pages 19447–19456, 2024. 3

[49] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes. *arXiv preprint arXiv:2404.10772*, 2024. 1, 3, 6, 7

[50] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024. 3

[51] Wenyuan Zhang, Yu-Shen Liu, and Zhizhong Han. Neural signed distance function inference through splatting 3d gaussians pulled on zero-level set. *Advances in Neural Information Processing Systems*, 37:101856–101879, 2025. 6, 7

[52] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 6

[53] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21634–21643, 2024. 1, 3