

Quadratic Gaussian Splatting for Efficient and Detailed Surface Reconstruction

Ziyu Zhang¹, Binbin Huang², Hanqing Jiang³, Liyang Zhou³, Xiaojun Xiang³, Shuhan Shen¹

¹CASIA ²The University of Hong Kong ³SenseTime Research

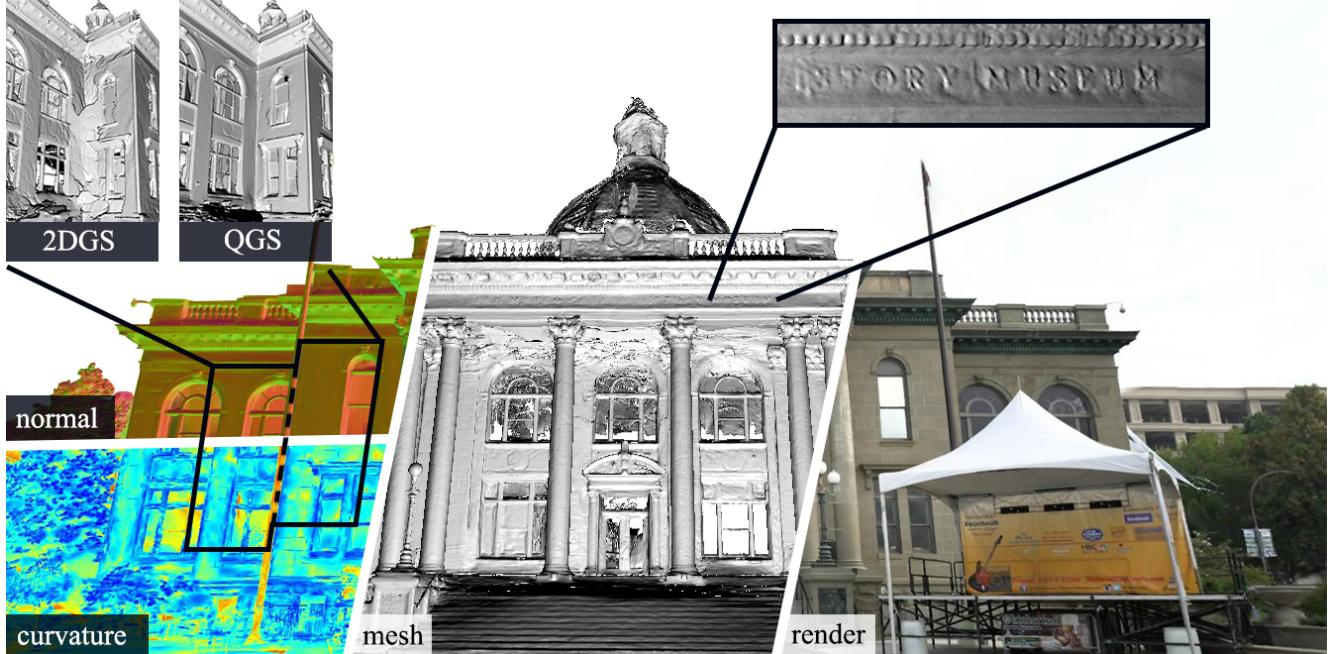


Figure 1. Quadratic Gaussian Splatting, a novel quadric-based approach for recovering high-precision real-world geometry from multi-view RGB images. In the Courthouse scene, our method achieves multi-view consistent normals, depth, and curvature, producing high-quality meshes and realistic images. Compared to 2DGS, QGS more accurately models the building’s contours.

Abstract

Recently, 3D Gaussian Splatting (3DGS) has attracted attention for its superior rendering quality and speed over Neural Radiance Fields (NeRF). To address 3DGS’s limitations in surface representation, 2D Gaussian Splatting (2DGS) introduced disks as scene primitives to model and reconstruct geometries from multi-view images, offering view-consistent geometry. However, the disk’s first-order linear approximation often leads to over-smoothed results. We propose Quadratic Gaussian Splatting (QGS), a novel method that replaces disks with quadric surfaces, enhancing geometric fitting. QGS defines Gaussian distributions in non-Euclidean space, allowing primitives to capture more complex textures. As a second-order surface approximation, QGS also renders spatial curvature to guide

the normal consistency term, to effectively reduce over-smoothing. Moreover, QGS is a generalized version of 2DGS that achieves more accurate and detailed reconstructions, as verified by experiments on DTU and TNT, demonstrating its effectiveness in surpassing most state-of-the-art methods in geometry reconstruction. Our code will be released as open source. The project is available at <https://quadraticgs.github.io/QGS>

1. Introduction

Surface reconstruction and realistic novel view synthesis (NVS) are critical tasks in computer graphics and vision. Their goal is to recover dense geometric structures from images captured from different viewpoints and render photo-realistic images. Recently, 3D Gaussian Splatting (3DGS)

has surpassed Neural Radiance Fileds (NeRF) based methods [2, 23, 24] in both rendering quality and speed by combining traditional splatting techniques with end-to-end optimization. Subsequently, GS-based methods have seen rapid advancements in dynamic reconstruction [8, 13], editing [6, 29], and large-scale scene reconstruction [20, 21].

However, vanilla splatting ignores the Gaussian contribution along the z-axis and uses approximation [16, 39]. As a result, it can render high-quality, multi-view consistent textures even at offsets from the scene surface, making it challenging to capture accurate geometry. In subsequent work [4, 12, 19, 35, 36], ray-splat intersections are computed in 3D to retrieve Gaussian weights, a key factor for accurate scene geometry recovery. In methods with Gaussian ellipsoids as primitives [35, 36], ray-splat intersections depend on the view direction, limiting normal consistency across views. In contrast, methods using Gaussian disks as primitives [12, 19] can ensure multi-view consistent geometry due to their surface-based nature, facilitating the incorporation of multi-view geometric consistency [4, 19]. However, disks, as a first-order fit of geometry, often result in overly smooth reconstructions, as shown in the Fig 2.

Earlier works [26, 31, 40] demonstrated that surfel elements can effectively represent complex scene geometry, such as atomic models. Building on this, 2DGS [12] combined Gaussian distributions with disk elements to achieve state-of-the-art (SOTA) reconstruction, inspiring our approach. In this paper, we propose a novel surface-based rendering method, which we call Quadratic Gaussian Splatting (QGS), for 3D scene reconstruction and novel view synthesis. This method enhances the geometric fitting of scene primitives through higher-order representation, enabling the extraction of accurate surface mesh from the quadric model.

Unlike previous splatting techniques operating in Euclidean space, such as 3DGS or 2DGS, our QGS defines the Gaussian distribution on a quadratic paraboloid, which can continuously transition between convex and concave forms, offering greater geometric fitting flexibility. The key technique in using paraboloids for splatting lies in establishing Gaussian distributions based on geodesic distances in non-Euclidean space, which allows the distribution's energy to be concentrated on the surface, to effectively capture complex geometric textures. Compared to the first-order approximation in 2DGS, QGS provides a second-order fit to the scene geometry. As a result, QGS not only produces multi-view consistent normals and depth but also provides curvature information, which represents surface bending and helps guide single-view normal consistency supervision. Furthermore, we observe that the volumetric rendering order used by most GS methods causes popping artifacts, which impact both novel view synthesis and geometric reconstruction. To address this, we adopted the sorting criteria from StopThePop [25], restructuring it to accom-

modate quadric surface structures, thereby improving the centroid sorting used in 2DGS to better handle complex intersections of quadratic surfaces. Finally, QGS is able to extract high-quality and detail-rich mesh models, as shown in the Fig 1.

In summary, the main contributions of our work are as follows:

- We propose QGS, a novel and efficient differentiable representation that uses quadric surfaces as scene primitives, offering stronger geometric fitting capabilities.
- We propose the first Gaussian Splatting work to introduce geodesic distance to establish Gaussian distributions on surfaces, enabling primitives to fit more complex textures.
- With stricter depth sorting in the higher-degree-of-freedom QGS, our method achieves SOTA geometric reconstruction and enhanced rendering quality.

2. Related work

2.1. Novel View Synthesis

Novel view synthesis has seen rapid development in recent years, especially since the introduction of NeRF [23]. NeRF encodes scene geometry and view-dependent appearance using a multi-layer perceptron (MLP) and optimizes it end-to-end through volumetric rendering to produce realistic images. Subsequent methods introduced various improvements. Mip-NeRF [1], Mip-NeRF 360 [2], and Zip-NeRF [3] addressed NeRF's aliasing issues by introducing new sampling strategies. I-NGP [24] and DVGO [28] significantly accelerated NeRF's training and rendering speeds by encoding scenes into feature grids instead of MLPs.

Recently, 3DGS [16] has gained attention for its more realistic rendering capabilities and real-time rendering speed, surpassing NeRF in many cases [22, 25, 34, 38]. Mip-Splatting [34] introduced a 3D smoothing filter to eliminate high-frequency artifacts during rendering. StopThePop [25] proposed per-tile and per-pixel resorting to resolve multi-view geometric inconsistencies. Scaffold-GS [22] effectively reduced redundant Gaussians and improved rendering quality by constructing anchor points to distribute local 3D Gaussians.

2.2. Neural Surface Reconstruction

Compared to traditional multi-view geometric reconstruction, neural volumetric rendering methods often produce smoother and more complete results. NeuS [30] and VolSDF [32] introduced signed distance fields (SDF) to describe scene geometry based on NeRF [23]. Geo-NeuS [9] and NeuralWarp [10] improved reconstruction by incorporating multi-view consistency. Neuralangelo [18] reduced MLP reliance by using feature grids and numerical gradients to improve geometric reconstruction quality.

However, neural rendering-based surface reconstruction

generally requires several hours to converge using only images. Additionally, due to the structured implicit fields, accessing and editing scene geometry is not straightforward.

2.3. Gaussian Splatting Surface Reconstruction

With the rapid development of 3DGS across various fields, Gaussian splatting methods have also made significant progress in surface reconstruction. SuGaR [11] introduced regularization terms to encourage Gaussians to fit the scene surface by building volumetric density fields from Gaussian ellipsoids and using Poisson reconstruction [15] to obtain a mesh. GSDF [33] and NeuSG [5] combined 3D Gaussian ellipsoids with SDF, achieving high-quality geometric reconstructions and rendering results. Rade-GS [36], GOF [35], and PGSR [4] computed ray-Gaussian ellipsoid intersections to obtain unbiased depth and used the normal consistency supervision from 2DGS [12] to achieve state-of-the-art reconstruction results. 2DGS flattened Gaussian ellipsoids into Gaussian disks to better align Gaussian primitives with surfaces and introduced two regularization losses to provide additional geometric constraints. MVG-splatting [19] extended 2DGS by introducing multi-view consistency constraints, enabling more complex surface reconstructions. In this paper, we propose Quadratic Gaussian Splatting (QGS), a new surface representation based on quadric surfaces designed to improve the local geometric fitting capability of primitives. We further establish Gaussian distributions on quadric, enabling end-to-end optimization. Lastly, we address the volume rendering order issue in 2DGS by incorporating the sorting method from StopThePop [25], achieving precise geometric reconstructions and high-quality rendering results.

3. Method

3.1. Preliminary

Kerbl et al. [16] proposed representing a scene using 3D Gaussian ellipsoids as primitives and render images using differentiable volume splatting. The shape and orientation of the ellipsoids are controlled by the scale \mathbf{S} and pose \mathbf{R} at their location \mathbf{p}_k . Together, \mathbf{R} , \mathbf{S} , and \mathbf{p}_k define a metric space, within which a Gaussian distribution is formulated:

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{p}_k)^T \Sigma^{-1} (\mathbf{x} - \mathbf{p}_k)\right) \quad (1)$$

Where the covariance matrix $\Sigma = \mathbf{R} \mathbf{S} \mathbf{S}^T \mathbf{R}^T$. To render the 3D Gaussian onto an image, each Gaussian distribution is first transformed into the camera space using the world-to-camera transformation \mathbf{W} . It is then mapped into ray space, where the viewing direction is aligned with the coordinate axis, through a local affine transformation \mathbf{J} [39].

$$\Sigma' = \mathbf{J} \mathbf{W} \Sigma \mathbf{W}^T \mathbf{J}^T \quad (2)$$

Under orthogonal projection, we skip the third row and column of Σ' , yielding a 2×2 covariance matrix Σ^{2D} to represent the 2D Gaussian G^{2D} . Subsequently, 3DGS employs volume rendering to integrate the 2D Gaussian.

$$C(\mathbf{p}) = \sum_{i=0}^{N-1} G_i^{2D}(\mathbf{p}) \alpha_i c_i \prod_{j=0}^{i-1} (1 - G_j^{2D}(\mathbf{p}) \alpha_j) \quad (3)$$

Here, α_i represents the opacity, c_i denotes the color of each Gaussian primitive, and \mathbf{p} is the pixel coordinate. Finally, each Gaussian primitive is optimized by minimizing the photometric loss.

GS mesh reconstruction. Vanilla 3DGS [16] can render high-quality images, but it yields suboptimal results for scene geometry reconstruction due to a lack of multi-view consistency in its splatting process. In subsequent geometric reconstruction methods, volumetric approaches like GOF [35] and RadeGS [36] employ ray-splat intersection techniques, achieving state-of-the-art reconstruction quality but limiting the consistency of normal and depth. In contrast, 2DGS [12] 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.

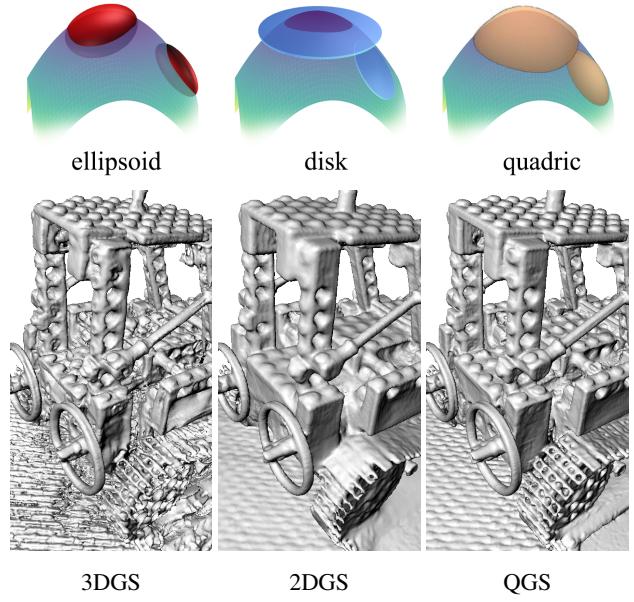


Figure 2. Demo and mesh reconstruction comparison on MipNeRF 360 dataset. 3D Gaussian ellipsoids struggle to fit precisely onto surfaces, resulting in coarse and incomplete meshes. Planes cannot fully capture high-curvature regions, leading to over-smoothed reconstructions. In contrast, quadric surfaces effectively capture geometric edge regions, achieving detailed reconstruction results.

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.

3.2.1 Quadratic Gaussian Model

Quadratic 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:

$$\begin{aligned} f(x, y, z) &= Ax^2 + 2Bxy + 2Cxz + 2Dx + Ey^2 \\ &\quad + 2Fyz + 2Gy + Hz^2 + 2Iz + J \\ &= [\mathbf{x} \quad \mathbf{y} \quad \mathbf{z} \quad 1] \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \\ D & G & I & J \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (4) \\ &= \mathbf{x}^T \mathbf{Q} \mathbf{x} = 0 \end{aligned}$$

Similar to [26], we apply congruent diagonalization to transform the above equation into its canonical form:

$$\begin{aligned} \mathbf{Q} &= \mathbf{T}^{-T} \mathbf{D} \mathbf{T}^{-1}, \text{ with } \mathbf{D} \text{ diagonal, } d_{ii} \in \{0, \pm 1\} \\ \mathbf{T} &= \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5) \end{aligned}$$

Here, \mathbf{c} denotes the position of the quadric, and $[\mathbf{u}, \mathbf{v}, \mathbf{w}]$ can be decomposed into \mathbf{RS} , where $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ and $\mathbf{S} = \text{diag}(s_1, s_2, s_3)$, which denotes the orientation and scale of the quadric in the object space. The matrix \mathbf{D} defines the surface shape: $\mathbf{D} = \text{diag}(1, 1, 1, 1)$ yields an ellipsoid, while $\mathbf{D} = \text{diag}(1, 0, 0, 0)$ 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. To concentrate the Gaussian energy on the surface, we use geodesic length [27] as the metric: the geodesic distance between two surface points is the shortest path along the surface, as illustrated by the red line in Fig 3.

However, not all geodesics have closed-form solutions. When \mathbf{Q} is an ellipsoid or hyperboloid, computing the geodesic length typically requires numerical methods. Therefore, we focus only on the case of a paraboloid:

$$\begin{aligned} f(x, y, z) &= \mathbf{x}^T \begin{bmatrix} \mathbf{R} & \mathbf{c} \\ 0 & 1 \end{bmatrix}^{-T} \bar{\mathbf{D}} \begin{bmatrix} \mathbf{R} & \mathbf{c} \\ 0 & 1 \end{bmatrix}^{-1} \mathbf{x} \\ &= \hat{\mathbf{x}}^T \begin{bmatrix} \frac{d_{11}}{s_1^2} & 0 & 0 & 0 \\ 0 & \frac{d_{22}}{s_2^2} & 0 & 0 \\ 0 & 0 & 0 & -\frac{d_{33}}{2s_3} \\ 0 & 0 & -\frac{d_{33}}{2s_3} & 0 \end{bmatrix} \hat{\mathbf{x}} \quad (6) \\ &= \frac{d_{11}}{s_1^2} \hat{x}^2 + \frac{d_{22}}{s_2^2} \hat{y}^2 - \frac{d_{33}}{s_3} \hat{z} = 0 \end{aligned}$$

Here and henceforth, we use $\hat{\cdot}$ 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.

$$f(\hat{x}, \hat{y}, \hat{z}) = \frac{\text{sign}(s_1)}{s_1^2} \hat{x}^2 + \frac{\text{sign}(s_2)}{s_2^2} \hat{y}^2 - \frac{1}{s_3} \hat{z} = 0 \quad (7)$$

In vanilla 3DGS [16], 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. First, we need to define a measure on the quadric. Paraboloid (Equation 7) could be expressed in explicit form:

$$\hat{z}(\hat{x}, \hat{y}) = s_3 \left(\frac{\text{sign}(s_1)}{s_1^2} \hat{x}^2 + \frac{\text{sign}(s_2)}{s_2^2} \hat{y}^2 \right) \quad (8)$$

We convert it isometrically into cylindrical coordinates, i.e. $\hat{x} = \rho \cos \theta, \hat{y} = \rho \sin \theta$. And we rewrite Equation 8 as:

$$\begin{aligned} \hat{z}(\theta, \rho) &= s_3 \left(\frac{\text{sign}(s_1) \cos^2 \theta}{s_1^2} + \frac{\text{sign}(s_2) \sin^2 \theta}{s_2^2} \right) \rho^2 \\ &= a(\theta) \rho^2 \quad (9) \end{aligned}$$

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 $\theta = \theta_0$ 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.

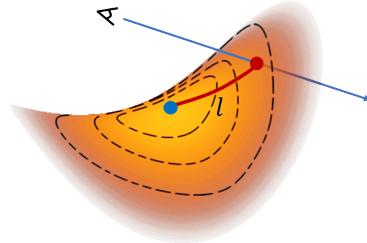


Figure 3. Illustration of a Gaussian on a quadric surface. The blue dot marks the Gaussian centroid, the red dot indicates the ray-splat intersection, and the red line represents the geodesic.

$$\begin{aligned} l(a, \rho_0) &= \int_0^{\rho_0} \sqrt{1 + (2at)^2} dt \\ &= \frac{\ln(\sqrt{u^2 + 1} + u) + u\sqrt{u^2 + 1}}{4a} \quad (10) \end{aligned}$$

where $u = 2a\rho_0$

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:

$$\frac{\rho^2 \cos^2 \theta}{s_1^2} + \frac{\rho^2 \sin^2 \theta}{s_2^2} = 1 \quad (11)$$

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.

$$\sigma(\theta_0) = \rho_0 = \frac{s_1 s_2}{\sqrt{(s_2 \cos \theta_0)^2 + (s_1 \sin \theta_0)^2}} \quad (12)$$

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

$$G(\hat{\mathbf{p}}_0(\theta_0, \rho_0)) = \exp\left(-\frac{(l(a(\theta_0), \rho_0))^2}{2(\sigma(\theta_0))^2}\right) \quad (13)$$

Notably, when $|s_3| \rightarrow 0$, the paraboloid becomes equivalent to a disk. Furthermore, as $x \rightarrow 0$, we have $\sqrt{1+x} \rightarrow 1$ and $\ln(1+x) \sim x$. Thus, as $|s_3| \rightarrow 0$, we get $a \rightarrow 0$ and $l \rightarrow \rho_0$ by Equation 10, 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.

3.2.2 Splatting

Although Sigg et al. [26] 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 8, 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, we select one of these two points: 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. The derivation is provided in the supplementary material.

Normal and Curvature. Similar to 2DGS [12], 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 7, yielding:

$$\hat{\mathbf{n}}_0(\hat{\mathbf{p}}_0) = \left(\frac{2\text{sign}(s_1)}{s_1^2} \hat{x}_0, \frac{2\text{sign}(s_2)}{s_2^2} \hat{y}_0, -\frac{1}{s_3} \right) \quad (14)$$

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 $\lambda_1 = \text{sign}(s_1) \cdot s_3/s_1^2$ and $\lambda_2 = \text{sign}(s_2) \cdot s_3/s_2^2$. The curvature at point $\hat{\mathbf{p}}_0$ is:

$$\hat{K}_0(\hat{\mathbf{p}}_0) = \frac{4\lambda_1\lambda_2}{(1 + 4\lambda_1^2\hat{x}_0^2 + 4\lambda_2^2\hat{y}_0^2)^2} \quad (15)$$

We render the normal map $N(\mathbf{p})$ and curvature map $K(\mathbf{p})$ by Equation 16 for a given viewpoint using alpha-blending, as shown in the Fig 4.

$$N(u, v) = \sum_{i=0}^{N-1} G_i \alpha_i \mathbf{n}_i \prod_{j=0}^{i-1} (1 - G_j \alpha_j) \quad (16)$$

$$K(u, v) = \sum_{i=0}^{N-1} G_i \alpha_i K_i \prod_{j=0}^{i-1} (1 - G_j \alpha_j)$$

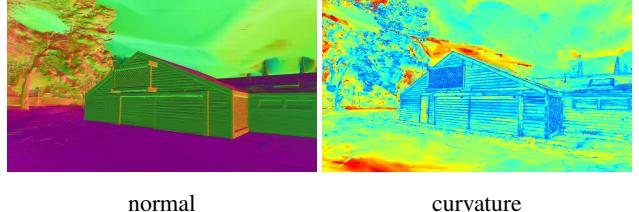


Figure 4. The normal map (left) and curvature map (right) of a scene. In the curvature map, blue indicates higher curvature and curved areas, while red indicates lower curvature and flatter areas.

Per-tile Sorting and Per-pixel Resorting. In Equation 16, the index i is determined by sorting primitives from near to far. Most GS-like methods sort by splat centroid depth rather than ray-splat intersection depth, which, as noted by StopThePop [25], causes popping artifacts. Additionally, we observed that centroid sorting introduces streak artifacts in geometry reconstruction, as shown in Fig 8. To address this, we apply StopThePop’s Per-tile Sorting and Per-pixel Resorting [25] to our QGS. In short, we first compute the ray closest to the quadric center in each tile to estimate intersection depth for rough sorting. During volume rendering, we use a buffer array to locally resort quadrics based on the intersection depth of each pixel. Further details are provided in the supplementary material.

3.3 Optimization

For QGS, which has stronger geometric fitting capability, relying solely on photometric consistency error can introduce geometric noise, such as unsMOOTH surfaces. Therefore, we adopt the depth distortion loss and normal consistency loss proposed by 2DGS [12] to mitigate this.

Depth Distortion. We employ the depth distortion loss to

promote tighter alignment of the quadric surfaces.

$$\mathcal{L}_d = \sum_{i=0}^{N-1} \sum_{j=0}^{i-1} \omega_i \omega_j (t_i - t_j)^2 \quad (17)$$

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 contributing to the ray. Additionally, following GOF's recommendation, we freeze the gradient of the depth distortion loss through ω_i , optimizing only the depth t_i .

Normal Consistency. 2DGS [12] introduces the normal consistency loss to ensure that all primitives on a ray are locally aligned with the actual surface.

$$\mathcal{L}_n = \sum_i \omega_i (1 - \mathbf{n}_i^T \mathbf{N}) \quad (18)$$

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.

$$\mathbf{N}(u, v) = \frac{\nabla_u \mathbf{p} \times \nabla_v \mathbf{p}}{|\nabla_u \mathbf{p} \times \nabla_v \mathbf{p}|} \quad (19)$$

However, neighboring pixels may not satisfy the local planar assumption, especially in regions with significant depth variation. Using differential normal consistency loss in edge areas can introduce errors, contributing to the over-smoothing observed in 2DGS. PGSR [4] and MVG-splatting [19] have both noted this issue and use image edges to approximate geometric edges, applying additional processing at these edges. We have found that, in most scenes, image edges do not fully correspond to geometric edges, especially in uniformly lit areas. Thus, we utilize the curvature map, which more accurately corresponds to geometric edges and is both efficiently and uniquely generated by QGS, to guide normal supervision.

$$\begin{aligned} \lambda_K(K(u, v)) &= 1 - \text{sigmoid}(\ln(|K(u, v)|) + \varepsilon) \\ \mathcal{L}_{Kn}(u, v) &= \lambda_K(K(u, v)) \mathcal{L}_n(u, v) \end{aligned} \quad (20)$$

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

$$\mathcal{L} = \mathcal{L}_c + \lambda_d \mathcal{L}_d + \lambda_n \mathcal{L}_{Kn} \quad (21)$$

4. Experiment

We compared our method with several SOTA approaches across multiple datasets, including DTU [14], TNT [17], 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, extending the renderer to output depth distortion maps, depth maps, normal maps, and curvature maps. Given the non-convex nature of paraboloids and the lack of an inverse for the geodesic distance function, we used rectangular truncation and approximations to compute image bounding boxes, as detailed in the supplementary materials. To handle the more complex surface intersections, we incorporate Per-tile Sorting, Per-pixel Local Resorting from StopThePop [25], with further details also in the supplementary materials.

Settings. We adopted the adaptive control strategy from 3DGS [16]. Similar to 2DGS [12], the QGS splatting process approximates by projecting the 3D center gradient onto screen space instead of relying directly on the 2D projected center gradient, using a gradient threshold of 0.3 and a percent dense value of 0.001. To ensure a similar number of Gaussian primitives across methods, we also set the gradient threshold for other methods to 2e-4. All experiments were conducted on a single A6000 GPU.

Mesh Extraction. We used a weighted combination of 0.5 median depth (i.e., $t_{\text{median}} = \max t_i | T_i > 0.5$) and 0.5 expected depth (i.e., $t_{\text{expected}} = \sum_{i=0}^{N-1} G_i \alpha_i T_i t_i$) as the final output depth. Then we fused the depth maps using Truncated Signed Distance Fusion (TSDF) with Open3D [37]. Following the 2DGS setup, we set the voxel size to 0.004 and truncation threshold to 0.002 for fair comparison.

4.2. Comparison

Geometry Evaluation In all experiments, we performed evaluations at half image resolution. In Table 1, we compared QGS with implicit [18, 30, 32] and explicit [11, 12, 35] SOTA reconstruction methods on the DTU dataset using Chamfer Distance as the benchmark. QGS outperformed all explicit methods, and while a gap remains with Neuralangelo [18], our reconstruction time is two orders of magnitude faster. In certain scenes, such as Scan 97 and Scan 105, QGS even outperformed Neuralangelo. In Table 2, we evaluated QGS and other SOTA methods on the TNT dataset using F1-score. In large-scale scenes, QGS consistently outperformed 2DGS [12]. Notably, in the Courthouse and Meetingroom scenes, QGS captured more details due to its higher geometric flexibility, as shown in the Fig 6.

As shown in Fig 5, our QGS achieves more complete geometry and stronger detail capture due to the greater flexibility of individual primitives. Note that our mesh extraction is limited by the lower resolution of the voxel grid in TSDF fusion, resulting in slightly lower accuracy compared to GOF [35], which uses marching tetrahedra, leading to longer meshing times and higher VRAM consumption.

QGS's second-order fit greatly improves on 2DGS [12], capturing finer details and providing second-order geomet-

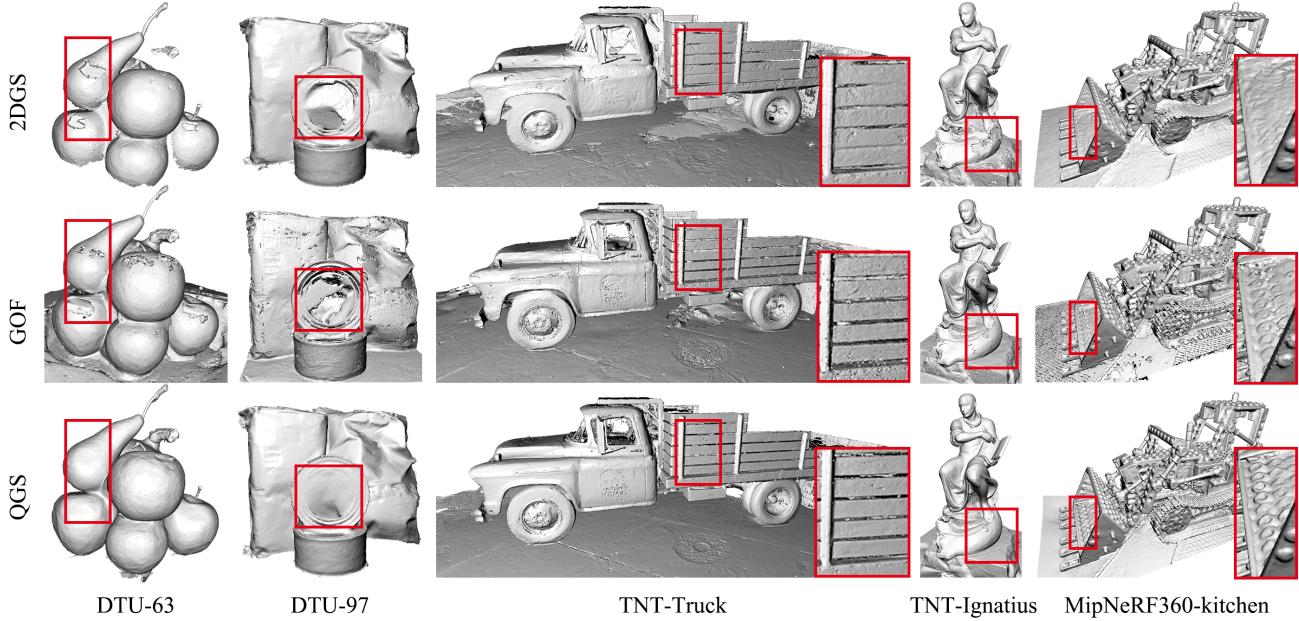


Figure 5. Qualitative geometric reconstruction comparisons on three datasets (DTU, Tanks and Temples, and Mip-NeRF 360) against 2DGS and GOF, covering indoor, outdoor, object-centric, and forward-facing scenes. QGS shows obvious improvements in both detail and smoothness compared to 2DGS and GOF.

CD (mm)↓	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean	Time
NeuS [30]	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 [32]	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 [18]	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
SuGaR [11]	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 [12]	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	0.32h
GOF [35]	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
Ours	0.46	0.81	0.40	0.42	1.07	0.80	0.79	1.25	0.95	0.67	0.62	1.27	0.40	0.60	0.49	0.73	0.83h

Table 1. Quantitative Chamfer Distance comparison of QGS with GS-like and NeRF-like methods on the DTU dataset. Compared to the planar representation in 2DGS, the surface-based QGS shows a clear improvement in geometric reconstruction quality. Additionally, QGS is an order of magnitude faster than NeRF-like methods.

F1-Score ↑	NeuS	Geo-NeuS	Neuralangelo	2DGS	GOF	Ours
Barn	0.29	0.33	0.70	0.41	0.51	0.43
Caterpillar	0.29	0.26	0.36	0.24	0.41	0.31
Courthouse	0.17	0.12	0.28	0.16	0.28	0.26
Ignatius	0.83	0.72	0.89	0.52	0.68	0.79
Meetingroom	0.24	0.20	0.32	0.17	0.28	0.25
Truck	0.45	0.45	0.48	0.45	0.58	0.60
Mean	0.38	0.35	0.50	0.33	0.46	0.44
Time	>24h	>24h	>127h	0.57h	2h	2h

Table 2. Quantitative F1-Score comparison of QGS with GS-like and NeRF-like methods on the TNT dataset. QGS shows notable gains over 2DGS and is competitive with prior SOTA methods.

ric information. The curvature map in Fig 4 illustrates how QGS better conforms to object edges with higher curvature, while adhering to flatter surfaces in other areas. It is important to note that the advantage of surface-based representations lies in their ability to provide multi-view consistent depth and normals (and curvature in our method). Thus, both theoretically and practically, it is straightforward to

transfer multi-view geometric techniques, as done in PGSR [4] and MVG-Splatting [19], to QGS. We leave this exploration for future work to further enhance our approach.

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

Curvature-Guided Normal Consistency Loss. As mentioned in Section 3.2, we use curvature to guide normal consistency supervision. High-curvature regions are treated as

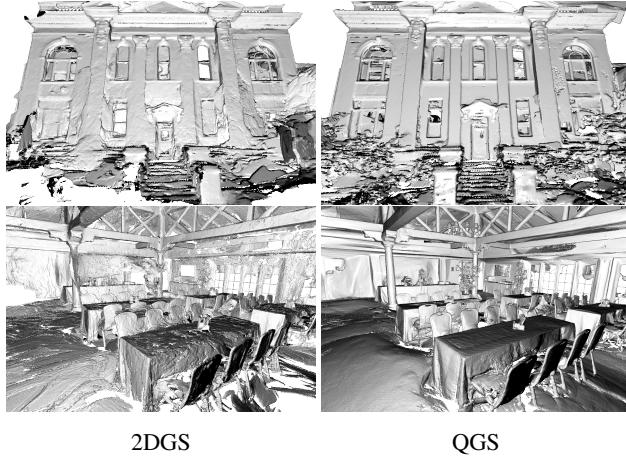


Figure 6. Mesh reconstruction comparison between QGS and 2DGS in large-scale scenes. The first row shows the Courthouse scene, with both using median depth, and the second row shows the Meetingroom scene, using a 0.5 weighted combination of median and expected depth.

	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	30.99	0.926	0.199	24.24	0.705	0.283
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
QGS	30.76	0.926	0.166	24.08	0.699	0.273

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

high-frequency areas that violate the local planar assumption, so we reduce supervision in these regions. As shown in Table 4 and Fig 7, curvature-guided normal supervision helps recover more scene details, leading to improved geometric quality.

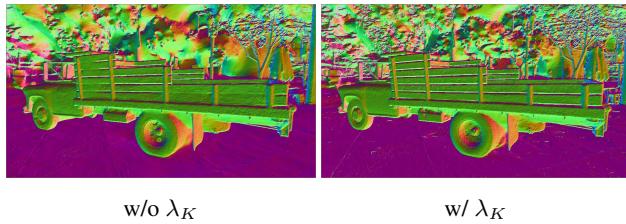


Figure 7. Comparison of curvature-guided normal consistency versus without curvature guidance. Curvature-guided supervision effectively reduces over-smoothing and recovers more details.

F1-Score ↑	Truck			Ignatius		
	Precision↑	Recall↑	F1-score↑	Precision↑	Recall↑	F1-score↑
w/o λ_K	0.63	0.51	0.57	0.74	0.75	0.75
w/ λ_K	0.65	0.55	0.60	0.78	0.80	0.79

Table 4. Quantitative F1-Score comparison on the TNT dataset with and without curvature-guided normal supervision.

Per-tile Sorting. 2DGS suggests the actual surface lies at the median intersection point where opacity reaches 0.5. However, most GS-like methods sort Gaussians by centroid depth, leading to streak-like inconsistencies, as shown in Fig 8. To address this, we implemented per-tile sorting (TS) and per-pixel resorting (PR) based on the methods proposed in StopThePop [25], further refining them to remove inconsistencies and enhance reconstruction quality, as shown in Table 5.

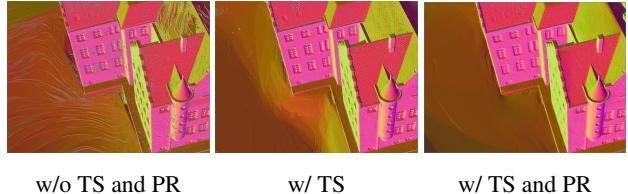


Figure 8. Normal comparison with and without per-tile sorting. Centroid depth sorting causes noticeable streak-like inconsistencies, also seen in 2DGS.

CD (mm)↓	24	37	97	122
w/o TS and PR	0.48	0.85	1.04	0.55
w/ TS	0.46	0.80	0.96	0.52
w/ TS + PR	0.46	0.81	0.95	0.49

Table 5. Quantitative Chamfer Distance comparison on the DTU dataset with and without per-tile sorting.

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.

Discussion. Gaussian Splatting is a rapidly evolving field, with numerous works exploring its use for surface reconstruction, all developed independently from ours, such as Rade-GS [36], PGSR [4], Trim-GS [7], and MVG-Splatting [19]. Our approach differs in focus, introducing a novel representation that provides curvature information. Our contributions are orthogonal to these approaches and, in theory, could enhance their methods as well.

Limitation. Since the bounding box of QGS primitives on the image is approximated, QGS processes more tiles than typical GS methods, particularly for larger quadric surfaces, resulting in a speed disadvantage. As a surface-based representation, QGS is still limited in capturing foggy scenes or transparent objects.

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.
- [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.
- [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.
- [4] 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.
- [5] Hanlin Chen, Chen Li, and Gim Hee Lee. Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance. *arXiv preprint arXiv:2312.00846*, 2023.
- [6] 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.
- [7] Lue Fan, Yuxue Yang, Minxing Li, Hongsheng Li, and Zhaoxiang Zhang. Trim 3d gaussian splatting for accurate geometry representation. *arXiv preprint arXiv:2406.07499*, 2024.
- [8] 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.
- [9] 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.
- [10] Josif Grabocka and Lars Schmidt-Thieme. Neuralwarp: Time-series similarity with warping networks. *arXiv preprint arXiv:1812.08306*, 2018.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.

- [26] Christian Sigg, Tim Weyrich, Mario Botsch, and Markus H Gross. Gpu-based ray-casting of quadratic surfaces. In *PBG@ SIGGRAPH*, pages 59–65, 2006.
- [27] James Johnston Stoker. *Differential geometry*. John Wiley & Sons, 2011.
- [28] 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.
- [29] 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.
- [30] 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.
- [31] Tim Weyrich, Simon Heinze, Timo Aila, Daniel B Fasnacht, Stephan Oetiker, Mario Botsch, Cyril Flraig, Simon Mall, Kaspar Rohrer, Norbert Felber, et al. A hardware architecture for surface splatting. *ACM Transactions on Graphics (TOG)*, 26(3):90–es, 2007.
- [32] 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.
- [33] Mulin Yu, Tao Lu, Lining Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai. Gsdf: 3dgs meets sdf for improved rendering and reconstruction. *arXiv preprint arXiv:2403.16964*, 2024.
- [34] 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.
- [35] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes. *arXiv preprint arXiv:2404.10772*, 2024.
- [36] 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.
- [37] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.
- [38] 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.
- [39] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001.
- [40] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001.

Quadratic Gaussian Splatting for Efficient and Detailed Surface Reconstruction

Supplementary Material

In this supplementary material, we provide detailed explanations of the following: (1) Calculation of the projected bounding box of primitives. (2) Details of per-tile and per-pixel sorting. (3) Solving the ray-primitive intersection equation and numerical handling. (4) Derivation of the geodesic distance formula integration. (5) Derivation of Gaussian curvature. (6) We present more qualitative results. Additionally, we include a video (QGS.mp4) that briefly explains the main workflow of QGS and showcases comparative results.

Details of Bounding Box Calculation.

In 2DGS/3DGS [12, 16], Gaussian primitives are enclosed convex representations, making it straightforward and convenient to compute their bounding box on the image, as shown in the first column of Fig 9.

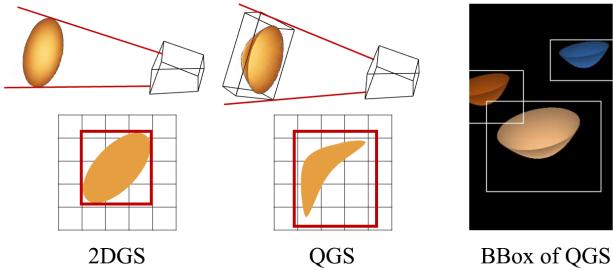


Figure 9. Comparison diagram of the bounding boxes between 2DGS and QGS.

However, QGS includes concave cases, and geodesic distance complicates analytical determination of the Gaussian primitive’s rendered portion during preprocessing. Specifically, the geodesic function Equation 10 in the main text lacks an elementary inverse, so computing the equipotential $\rho(\theta_0) = \{l^{-1}(l_0) : l_0 = \sigma(\theta_0)\}$, requires numerical or approximate solutions. Even with an analytical solution for $\rho(\theta_0) = l^{-1}(\sigma(\theta_0))$, the Gaussian distribution in non-Euclidean space means equipotential lines may not fully enclose the primitive, requiring extra boundary calculations. To simplify, we use the quadric’s circumscribing rectangular box as the 3D bounding box. Its 8 vertices are projected onto the image plane, and their 2D AABB is used as the 2D bounding box, as shown in Fig 9, columns two and three. Specifically, we first scale the geodesic function to a quadratic polynomial function:

$$l(a, \rho) \approx \hat{l}(a, \rho) = \frac{4|a|\rho^2 + 6\rho}{7} \quad (22)$$

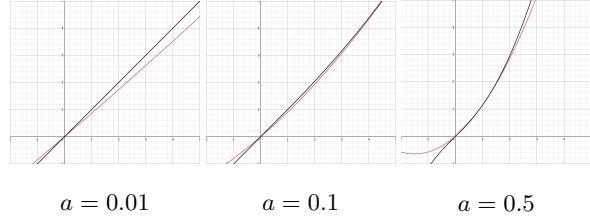


Figure 10. Comparison diagram of the approximate function versus the geodesic function. The red line represents the approximate function, while the black line represents the geodesic function.

Given the direction θ_0 , we obtain the variance of the Gaussian distribution $\sigma_0(\theta_0)$. The root of the equation $\hat{l}(a, \rho) - \sigma_0 = 0$ in this direction can be calculated as:

$$\rho_{0,1} = \frac{-6 \pm \sqrt{36 + 112|a|\sigma_0}}{8|a|} \quad (23)$$

The comparison between the approximate function and the geodesic distance function is shown in Fig 10, demonstrating that the two functions nearly overlap.

Since the quadratic curve passes through the origin, we take its positive root, representing the horizontal distance from the primitive’s vertex at a geodesic distance of σ_0 . We compute this for the major and minor axes, i.e. $\theta = 0$ and $\theta = \pi/2$, yielding ρ_l and ρ_s , with circumscribing box radii $(\rho_l, \rho_s, a \max(\rho_l^2, \rho_s^2))$. We then project the eight vertices of the bounding box onto the image and take the minimum 2D bounding box of the resulting polygon. Due to approximations, gaps may appear between the bounding box and the primitive as shown in the third column of Fig 9, affecting rendering speed but not quality.

Details of Per-tile Sorting and Per-pixel Resorting

2DGS [12] suggests the surface lies at the median intersection point where opacity reaches 0.5. However, for surface-based representations like 2DGS and QGS, the Gaussian distribution is concentrated on the surface, making the alpha-blending order more sensitive. Most reconstruction methods [4, 19, 35, 36], including 2DGS, sort by Gaussian centroid depth, causing geometric inconsistencies, as shown in Fig 8 of the main text.

To address this, we introduce StopThePop’s per-tile sorting and per-pixel resorting [25] for more precise ordering. Specifically, for each 16×16 pixel tile, we compute the intersection depth using the ray from the pixel closest to the projected vertex of the quadric surface and apply this depth to all 256 pixels for tile-based global sorting. As shown in Fig 8 in the main text, this method effectively removes

streak-like inconsistencies but introduces minor blocky artifacts due to approximating with a single ray per tile. Then we adopt StopThePop’s per-pixel local resorting to reorder the Gaussians along each ray, to eliminate blocky inconsistencies. Specifically, after calculating each Gaussian’s depth, normal, and other properties, we do not use them for alpha-blending immediately. Instead, we store them in an 8-length buffer array, and once the buffer is full, we select the closest Gaussian for alpha-blending. In original 3DGS, forward and backward computations use different traversal orders, causing the buffering mechanism to produce inconsistent rendering. Thus, following StopThePop’s strategy, we perform gradient computation in a near-to-far order, which requires modifying the original gradient formulas. For volumetric rendering maps, we can uniformly express them as:

$$\hat{X} = \sum_{i=0}^{N-1} \bar{\alpha}_i T_i X_i, \text{ where } \bar{\alpha}_i = \alpha_i g_i, T_i = \prod_{j=0}^{i-1} (1 - \bar{\alpha}_j)$$

Where X_i represents properties such as the color, depth, normal, or curvature of the primitives. The gradient computation in a back-to-front order is given as follows:

$$\begin{aligned} \frac{\partial \hat{X}}{\partial \bar{\alpha}_i} &= X_i T_i - \frac{\sum_{j=i+1}^N X_j \bar{\alpha}_j T_j}{1 - \bar{\alpha}_i} \\ &= (X_i - \frac{\sum_{j=i+1}^N X_j \bar{\alpha}_j T_j}{T_{i+1}}) T_i \end{aligned}$$

We rewrite it as front-to-back form:

$$\frac{\partial \hat{X}}{\partial \bar{\alpha}_i} = (X_i - \frac{\hat{X} - \sum_{j=0}^i X_j \bar{\alpha}_j T_j}{T_{i+1}}) T_i$$

Ensuring that the backward computation follows the same order as the forward computation. Fortunately, the gradient computation order does not affect the distortion loss, meaning no additional derivation is needed for the distortion loss.

Details of Ray-splat Intersection

Given the camera center $\hat{\mathbf{o}} = [\hat{o}^1, \hat{o}^2, \hat{o}^3]^T$ and ray direction $\hat{\mathbf{r}} = [\hat{r}^1, \hat{r}^2, \hat{r}^3]^T$ in the Gaussian coordinate system, the ray can be expressed as:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{o}^1 + t\hat{r}^1 \\ \hat{o}^2 + t\hat{r}^2 \\ \hat{o}^3 + t\hat{r}^3 \end{bmatrix} \quad (24)$$

Here, t denotes the depth. By solving the ray equation 24 together with the quadric surface equation 7 in the main text, we find two intersection points:

$$\begin{aligned} t^2 \left[\frac{\text{sign}(s_1)}{s_1^2} (\hat{r}^1)^2 + \frac{\text{sign}(s_2)}{s_2^2} (\hat{r}^2)^2 \right] + \\ t \left[\frac{\text{sign}(s_1)}{s_1^2} 2\hat{o}^1 \hat{r}^1 + \frac{\text{sign}(s_2)}{s_2^2} 2\hat{o}^2 \hat{r}^2 - \frac{1}{s_3} \hat{r}^3 \right] + \\ \left[\frac{\text{sign}(s_1)}{s_1^2} (\hat{o}^1)^2 + \frac{\text{sign}(s_2)}{s_2^2} (\hat{o}^2)^2 - \frac{1}{s_3} \hat{o}^3 \right] = 0 \end{aligned} \quad (25)$$

$$At^2 + Bt + C = 0$$

$$t_n = \frac{-B - \text{sign}(A) \cdot \sqrt{B^2 - 4AC}}{2A},$$

$$t_f = \frac{-B + \text{sign}(A) \cdot \sqrt{B^2 - 4AC}}{2A}$$

However, Equation 25 encounters numerical issues when $|A|$ is very small, particularly during backpropagation, as a small denominator can cause floating-point overflow, resulting in invalid values such as NaN or Inf. Upon examination, we observe that when A is especially small, it corresponds to cases where the quadric surface is nearly flat or when the ray is almost perpendicular to the $s_1 \times s_2$ plane. In such cases, we can ignore the quadratic term, reducing Equation 25 to $Bt + C = 0$, or $t = -\frac{C}{B}$. Geometrically, this solution represents the depth of the intersection between the ray and the tangent plane determined by the point where the perpendicular line from the camera center meets the quadric surface, as indicated by the thick blue line n in Fig 11.

Let $\mathbf{M} = s_1 \times s_2$ be the horizontal plane in the Gaussian local coordinate system. Let $\mathbf{p}_o = [\hat{o}_1, \hat{o}_2, s_3(\frac{\hat{o}_1^2}{s_1^2} + \frac{\hat{o}_2^2}{s_2^2})]^T$ represent the intersection of the perpendicular line from the camera center to \mathbf{M} and the quadric surface. The normal of the tangent plane \mathbf{N} at the intersection point can be expressed as:

$$\hat{\mathbf{n}}(\mathbf{p}_o) = [\frac{2\text{sign}(s_1)\hat{o}_1}{s_1^2}, \frac{2\text{sign}(s_2)\hat{o}_2}{s_2^2}, -\frac{1}{s_3}]^T$$

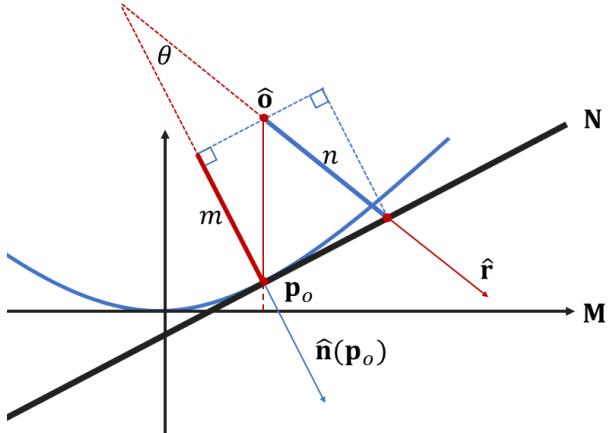


Figure 11. Illustration of approximate intersections. The blue thick line represents the depth of the approximate intersection.

The projection of $(\mathbf{p}_o - \hat{\mathbf{o}})$ onto the normal direction \mathbf{n} is:

$$m = \frac{\hat{\mathbf{n}}(\mathbf{p}_o) \cdot (\mathbf{p}_o - \hat{\mathbf{o}})}{\|\hat{\mathbf{n}}(\mathbf{p}_o)\|}$$

As indicated by the thick red line segment in the Fig 11. On the other hand, the cosine of the angle between the normal vector and the viewing direction can be expressed as:

$$\cos\theta = \frac{\hat{\mathbf{r}} \cdot \hat{\mathbf{n}}(\mathbf{p}_o)}{\|\hat{\mathbf{r}}\| \cdot \|\hat{\mathbf{n}}(\mathbf{p}_o)\|}$$

Noting that $\|\hat{\mathbf{r}}\| = 1$, the length of the line from the camera center along the viewing direction to the intersection with the tangent plane is given by:

$$\begin{aligned} n &= \frac{m}{\cos\theta} = \frac{\hat{\mathbf{n}}(\mathbf{p}_o) \cdot (\mathbf{p}_o - \hat{\mathbf{o}})}{\hat{\mathbf{r}} \cdot \hat{\mathbf{n}}(\mathbf{p}_o)} \\ &= -\frac{C}{B} \end{aligned}$$

It is evident that when the surface is nearly flat or the viewing direction is almost perpendicular to the horizontal plane, the approximate solution nearly coincides with the exact solution. Therefore, for cases where $|A| < 1e-6$, we approximate the solution using the method described above.

Derivation of Geodesic Arc Length

For the geodesic arc length formula:

$$l(a, \rho_0) = \int \sqrt{1 + (2at)^2} dt \quad (26)$$

Let $u = 2at$, $x = \arctan(u)$, then Equation 26 becomes:

$$\begin{aligned} l(a, \rho_0) &= \int \frac{\sqrt{1+u^2}}{2a} du \\ &= \frac{1}{2a} \int \sqrt{1+\tan^2(x)} d\tan(x) \\ &= \frac{1}{2a} \int \sec(x) d\tan(x) \\ &= \frac{1}{2a} \int \sec^3(x) dx \end{aligned} \quad (27)$$

Equation 27 can be solved using integration by parts:

$$\begin{aligned} \int \sec^3(x) dx &= \tan(x) \sec(x) - \int \tan(x) d\cos(x) \\ &= \tan(x) \sec(x) - \int \sec^3(x) dx \\ &\quad + \int \sec(x) dx \end{aligned}$$

Using $\int \sec(x) dx = \ln |\sec(x) + \tan(x)| + C$, we have:

$$\begin{aligned} \int \sec^3(x) dx &= \frac{\tan(x) \sec(x) + \ln |\sec(x) + \tan(x)|}{2} \\ &= (u\sqrt{1+u^2} + \ln(u + \sqrt{1+u^2}))/2 \\ \Rightarrow l(a, \rho_0) &= \frac{u\sqrt{1+u^2} + \ln(u + \sqrt{1+u^2})}{4a} \end{aligned}$$

Details of Curvature

Here, we compute the Gaussian curvature analytically using a standard differential geometry approach [27]. Given the intersection point $\hat{\mathbf{p}}_0 = [\hat{x}_0, \hat{y}_0, \hat{z}_0]^T$, we simplify Equation 7 as $\hat{z} = \lambda_1 \hat{x}^2 + \lambda_2 \hat{y}^2$. The partial derivatives at $\hat{\mathbf{p}}_0$ are:

$$x_u = (1, 0, 2\lambda_1 \hat{x}_0)$$

$$x_v = (0, 1, 2\lambda_2 \hat{y}_0)$$

The first fundamental form is:

$$E = \langle x_u, x_u \rangle = 1 + 4\lambda_1^2 \hat{x}_0^2$$

$$F = \langle x_u, x_v \rangle = 4\lambda_1 \lambda_2 \hat{x}_0 \hat{y}_0$$

$$G = \langle x_v, x_v \rangle = 1 + 4\lambda_2^2 \hat{y}_0^2$$

The second fundamental form is:

$$n = \frac{x_u \times x_v}{\|x_u \times x_v\|} = \frac{(-2\lambda_1 \hat{x}_0, -2\lambda_2 \hat{y}_0, 1)}{\sqrt{1 + 4\lambda_1^2 \hat{x}_0^2 + 4\lambda_2^2 \hat{y}_0^2}}$$

$$x_{uu} = (0, 0, 2\lambda_1)$$

$$x_{uv} = (0, 0, 0)$$

$$x_{vv} = (0, 0, 2\lambda_2)$$

$$L = \langle n, x_{uu} \rangle = \frac{2\lambda_1}{\sqrt{1 + 4\lambda_1^2 \hat{x}_0^2 + 4\lambda_2^2 \hat{y}_0^2}}$$

$$M = \langle n, x_{uv} \rangle = 0$$

$$N = \langle n, x_{vv} \rangle = \frac{2\lambda_2}{\sqrt{1 + 4\lambda_1^2 \hat{x}_0^2 + 4\lambda_2^2 \hat{y}_0^2}}$$

Finally, the Gaussian curvature can be computed as:

$$K = \frac{LN - M^2}{EG - F^2} = \frac{\frac{4\lambda_1 \lambda_2}{1 + 4\lambda_1^2 \hat{x}_0^2 + 4\lambda_2^2 \hat{y}_0^2}}{1 + 4\lambda_1^2 \hat{x}_0^2 + 4\lambda_2^2 \hat{y}_0^2}$$

Additional Results. In this section, we present additional qualitative results of QGS in both indoor and outdoor scenarios. Fig 12 compares QGS and 2DGS on the DTU dataset, where QGS demonstrates sharper contours. QGS also achieves accurate reconstructions in indoor and outdoor scenes, as shown in Fig 13 with results from the TNT and Mip-NeRF 360 datasets. Rendering results for all three datasets are shown in Fig 14. Finally, we qualitatively tested our method on an urban scene captured from aerial view. As shown in Fig 15, QGS captures more building details than 2DGS [12] in aerial views. Additionally, most methods using the default 3DGS configuration failed to reconstruct the Courthouse scene and the urban scene, both containing nearly 1,000 images. For these two scenes, we simply doubled the training parameters, setting the total iterations to 60,000, densifying every 800 iterations from the 4,000th to the 48,000th iteration, and resetting Gaussian opacity every 6,000 iterations. For other scenes, we used the default 3DGS configuration.

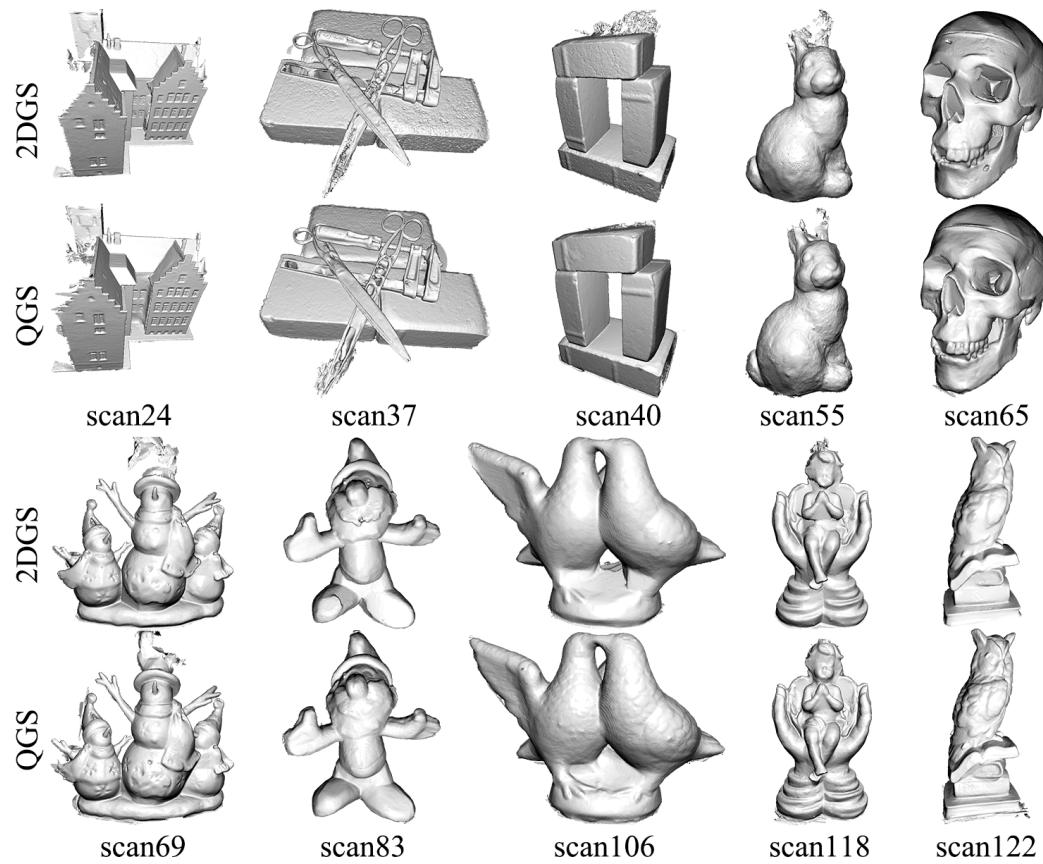


Figure 12. Qualitative geometry comparison for the DTU dataset [14].

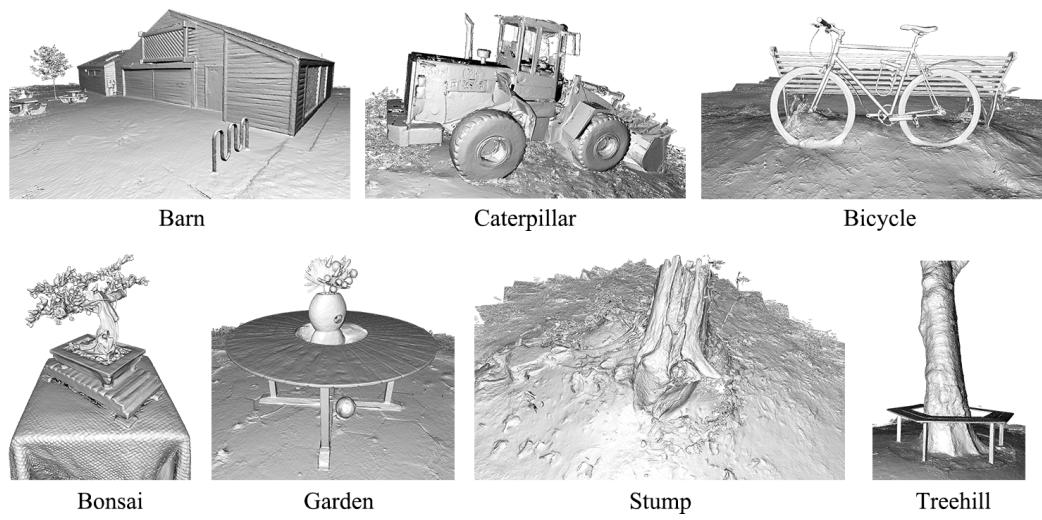


Figure 13. Qualitative geometry results for the Tanks and Temples dataset [17] and Mip-NeRF 360 dataset [2].

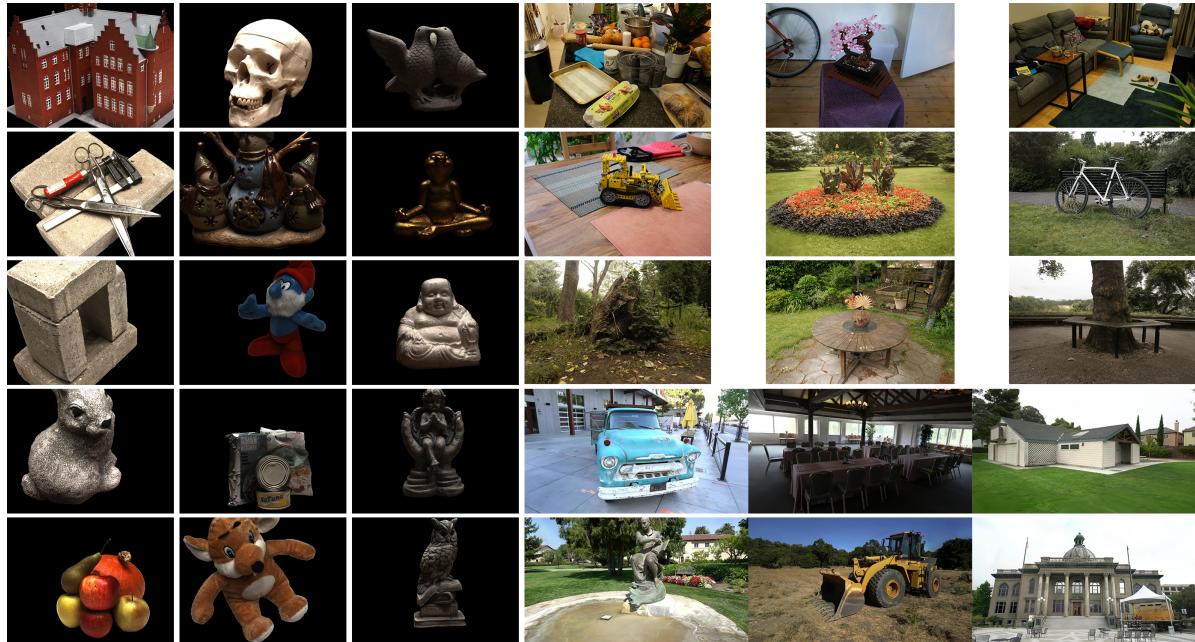


Figure 14. Qualitative appearance results for the DTU, TNT and Mip-NeRF 360 dataset.

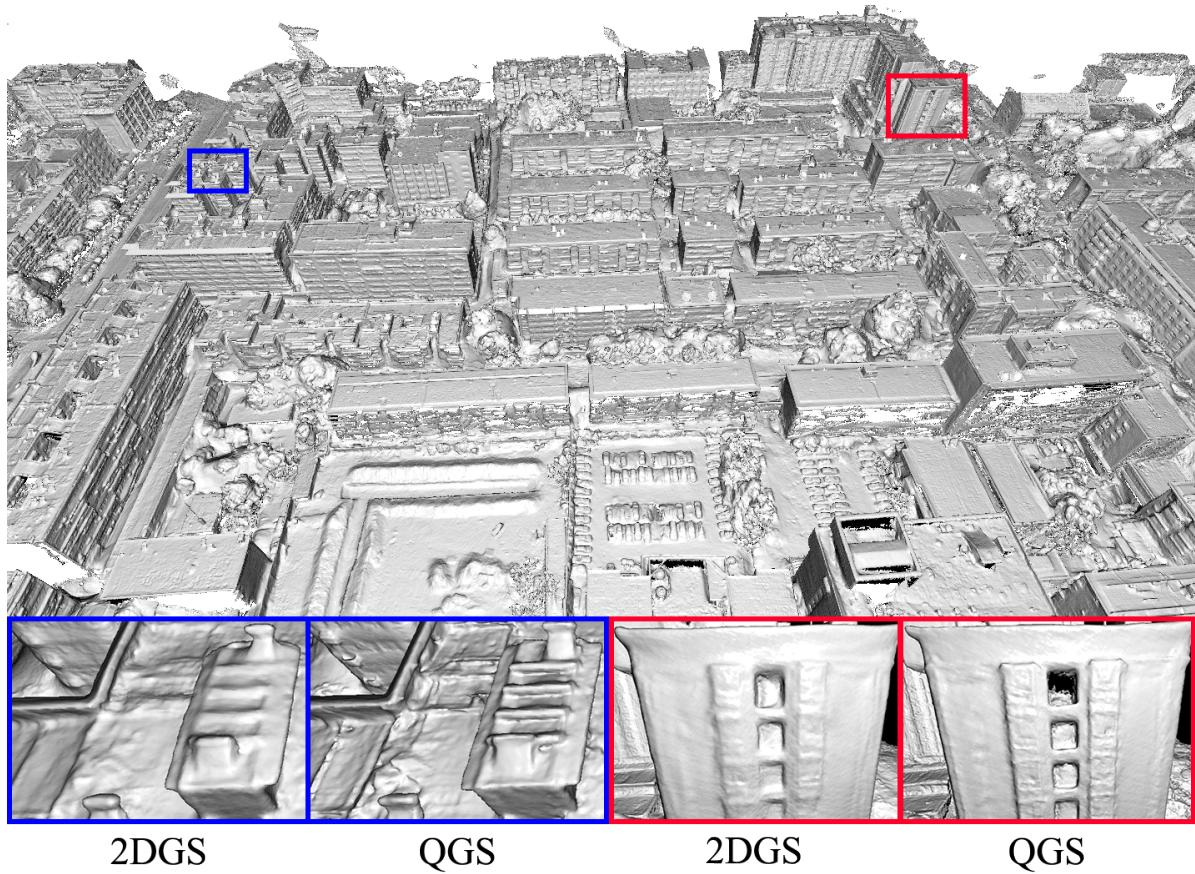


Figure 15. Qualitative geometry result for the urban scene captured from aerial view, involving over 1,000 images. In the subfigure, the left shows the results of 2DGS, while the right shows the results of QGS.