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

Supplementary Material

In this supplementary material, we provide detailed explanations of the following: (1) Comparison of quadric (QGS) and disk (2DGS [4]). (2) Calculation of the projected bounding box of primitives. (3) Details of per-tile and per-pixel sorting. (4) Solving the ray-primitive intersection equation and numerical handling. (5) Derivation of the geodesic distance formula integration. (6) Derivation of Gaussian curvature. (7) We present more qualitative results.

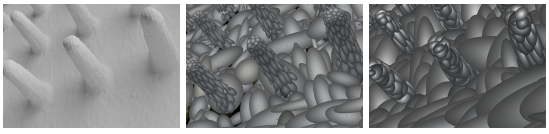
Comparison of Disk and Quadric

We visualize the primitives of QGS (Quadric) and 2DGS (Disk), as shown in Fig 1. It can be observed that quadrics tightly conform to the surface in a curved manner, with a more noticeable fit in regions of high curvature.

Details of Bounding Box Calculation.

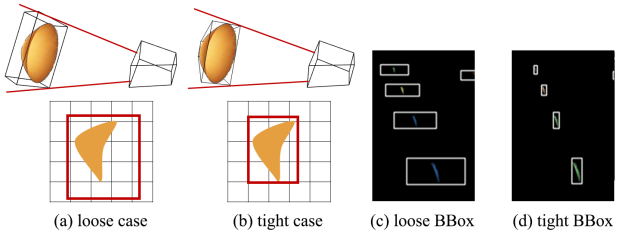
In 2DGS/3DGS [4, 6], Gaussian primitives are enclosed convex representations, making it straightforward and convenient to compute their bounding box on the image, as shown in the first column of Fig 2.

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. Moreover, 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. A straightforward approach is to use the



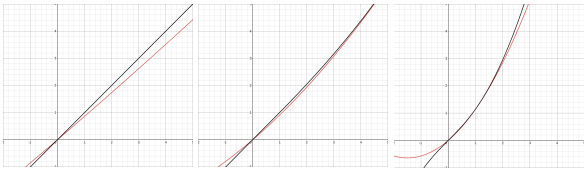
(a) Mesh (b) Disk (c) Quadric

Figure 1. Visualizaion results on GSO dataset [3]. (a) shows the ground truth mesh, (b) displays the disk primitives, and (c) presents the quadric primitives.



(a) loose case (b) tight case (c) loose BBox (d) tight BBox

Figure 2. Comparison diagram of the bounding boxes between loose and tight case.



$a = 0.01$ $a = 0.1$ $a = 0.5$

Figure 3. 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.

projection of the quadric's 3D bounding box as the 2D bounding box, as shown in Fig 2 (a). Its eight vertices are projected onto the image plane, and their 2D axis-aligned bounding box (AABB) is used as the 2D bounding box, as illustrated in Fig 2 (c). However, this loose case significantly enlarges the 2D bounding box, leading to a decrease in efficiency. To address this, we construct a tighter bounding box using the quadric's tangent planes, as shown in Fig 2 (b). Specifically, we first scale the geodesic function to a quadratic polynomial function:

$$l(a, \rho) \approx \hat{l}(a, \rho) = \frac{4|a|\rho^2 + 6\rho}{7} \tag{1}$$

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:

$$\rho_{0,1} = \frac{-6 \pm \sqrt{36 + 112|a|\sigma_0}}{8|a|} \tag{2}$$

The comparison between the approximate function and the geodesic distance function is shown in Fig 3, demonstrating that the two functions nearly overlap.

Since the Equation 1 through the origin, we take its positive root, representing the horizontal distance from the primitive's vertex at a geodesic distance of $\sigma_0(\theta_0)$. We compute this for the major and minor axes, i.e. $\theta = 0$ and $\theta = \pi/2$, yielding ρ_1 and ρ_2, and obtain four points: $(\pm\rho_1, 0, \rho_1^2), (0, \pm\rho_2, \rho_2^2)$. We then compute the intersection of the tangent planes at these four points with the plane $z = 0$.

$$x_1 = \pm\frac{\rho_1 s_1 - \rho_1^2}{s_1}, \quad x_2 = \pm\frac{\rho_2 s_2 - \rho_2^2}{s_2} \tag{3}$$

At this stage, we obtain a 3D truncated pyramid with top dimensions $\rho_1 \times \rho_2$, bottom dimensions $x_1 \times x_2$, and height

	Mip-NeRF360			TNT	
	Storage	FPS	Training Time	Storage	FPS
2DGS	556MB	15.34	1h5min	218MB	31.66
GOF	825MB	7.61	2h12min	367MB	13.45
QGS	688MB	7.92	1h48min	274MB	13.27
QGS w/ TB	641MB	14.15	1h13min	263MB	25.36

Table 1. Speed and storage comparison on the Mip-NeRF 360 [1] and TNT [7] datasets. With a tighter bounding box, QGS achieves a noticeable speed improvement.

$\max(\rho_1^2, \rho_2^2)$. We then project its eight vertices onto the image plane and determine the minimum 2D bounding box enclosing the resulting polygon as shown in Fig 2 (d).

By introducing a tighter bounding box and optimizing global memory access on the GPU, QGS achieves twice the speed. We denote the accelerated QGS as QGS w/ TB and compare the storage and speed of different methods on the Mip-NeRF 360 and TNT datasets as shown in Table 1.

Details of Per-tile Sorting and Per-pixel Resorting

2DGS [4] 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.

To address this, we introduce StopThePop's per-tile sorting and per-pixel resorting [8] 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 7 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:

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

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:

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

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.

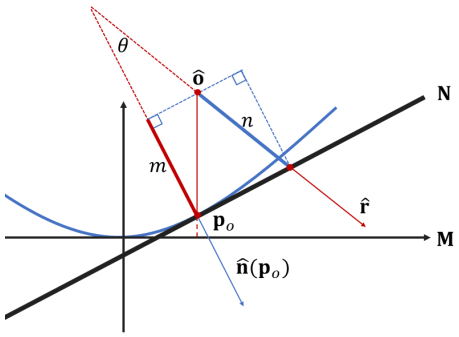


Figure 4. Illustration of approximate intersections. The blue thick line represents the depth of the approximate intersection.

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:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{o}^1 + t\hat{r}^1 \\ \hat{o}^2 + t\hat{r}^2 \\ \hat{o}^3 + t\hat{r}^3 \end{bmatrix} \tag{4}$$

Here, t denotes the depth. By solving the ray equation 4 together with the quadric surface equation 7 in the main text, we find two intersection points:

$$\begin{aligned} &t^2[\frac{\text{sign}(s_1)}{s_1^2}(\hat{r}^1)^2 + \frac{\text{sign}(s_2)}{s_2^2}(\hat{r}^2)^2] + \\ &t[\frac{\text{sign}(s_1)}{s_1^2}2\hat{o}^1\hat{r}^1 + \frac{\text{sign}(s_2)}{s_2^2}2\hat{o}^2\hat{r}^2 - \frac{1}{s_3}\hat{r}^3] + \\ &[\frac{\text{sign}(s_1)}{s_1^2}(\hat{o}^1)^2 + \frac{\text{sign}(s_2)}{s_2^2}(\hat{o}^2)^2 - \frac{1}{s_3}\hat{o}^3] = 0 \\ &At^2 + Bt + C = 0 \\ &t_{n,f} = \frac{-B \pm \text{sign}(A) \cdot \sqrt{B^2 - 4AC}}{2A} \end{aligned} \tag{5}$$

However, Equation 5 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 5 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 4.

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:

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

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 4. 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:

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

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 $|A| < 1\text{e}-6$, we approximate the solution using the method described above.

Derivation of Geodesic Arc Length

For the geodesic arc length formula:

$$l(a, \rho_0) = \int \sqrt{1 + (2at)^2} dt \tag{6}$$

Let $u = 2at, x = \arctan(u)$, then Equation 6 becomes:

$$\begin{aligned} l(a, \rho_0) &= \int \frac{\sqrt{1+u^2}}{2a} du \\ &= \frac{1}{2a} \int \sqrt{1 + \tan^2(x)} d\tan(x) \\ &= \frac{1}{2a} \int \sec^3(x) dx \end{aligned} \tag{7}$$

Equation 7 can be solved using integration by parts:

$$\int \sec^3(x)dx = \tan(x)\sec(x) - \int \sec^3(x)dx + \int \sec(x)dx$$

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 [9]. 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:

$$x_u = (1, 0, 2\lambda_1\hat{x}_0), \quad x_v = (0, 1, 2\lambda_2\hat{y}_0)$$

The first fundamental form is:

$$\begin{aligned} E &= < x_u, x_u > = 1 + 4\lambda_1^2\hat{x}_0^2 \\ F &= < x_u, x_v > = 4\lambda_1\lambda_2\hat{x}_0\hat{y}_0 \\ G &= < x_v, x_v > = 1 + 4\lambda_2^2\hat{y}_0^2 \end{aligned}$$

The second fundamental form is:

$$\begin{aligned} n &= \frac{x_u \times x_v}{||x_u \times x_v||} = \frac{(-2\lambda_1\hat{x}_0, -2\lambda_2\hat{y}_0, 1)}{\sqrt{1 + 4\lambda_1^2\hat{x}_0^2 + 4\lambda_2^2\hat{y}_0^2}} \\ x_{uu} &= (0, 0, 2\lambda_1), \; x_{uv} = (0, 0, 0), \; x_{vv} = (0, 0, 2\lambda_2) \\ L &= < n, x_{uu} > = \frac{2\lambda_1}{\sqrt{1 + 4\lambda_1^2\hat{x}_0^2 + 4\lambda_2^2\hat{y}_0^2}} \\ M &= < n, x_{uv} > = 0 \\ N &= < n, x_{vv} > = \frac{2\lambda_2}{\sqrt{1 + 4\lambda_1^2\hat{x}_0^2 + 4\lambda_2^2\hat{y}_0^2}} \end{aligned}$$

Finally, the Gaussian curvature can be computed as:

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

Addtional Results. In this section, we present additional qualitative results of QGS in both indoor and outdoor scenarios. Figure 5 presents a comparison of QGS with 2DGS [4], GOF [10], and PGSR [2] on the DTU dataset, illustrating that QGS captures more geometric details. QGS also achieves accurate reconstructions in indoor and outdoor scenes, as shown in Fig 6 with results from the TNT and Mip-NeRF 360 datasets. Rendering results for all three datasets are shown in Fig 7. Finally, we qualitatively tested our method on an urban scene captured from aerial view. As shown in Fig 8, QGS captures more building details than 2DGS [4] in aerial views.

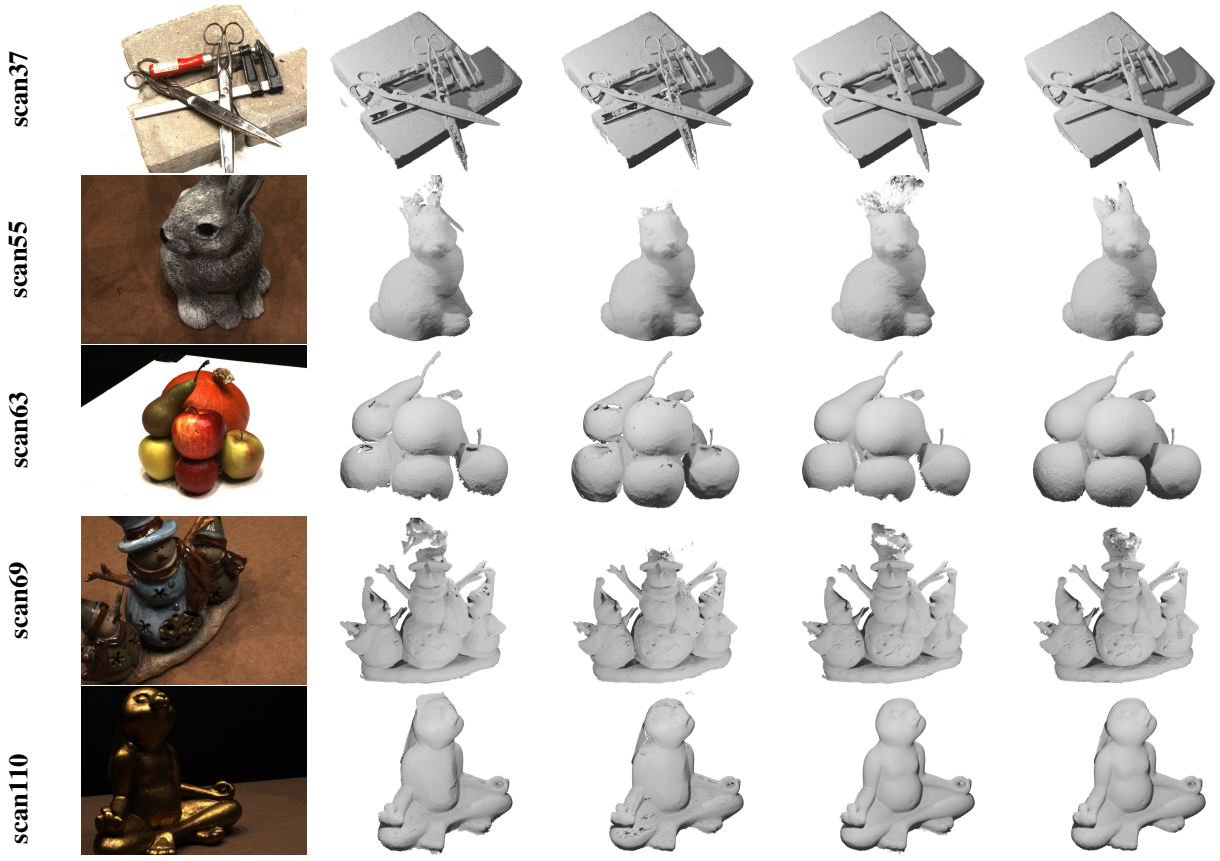


Figure 5. Qualitative geometric reconstruction comparisons on the DTU dataset [5]. Our method achieves reconstructions of higher quality and greater detail.

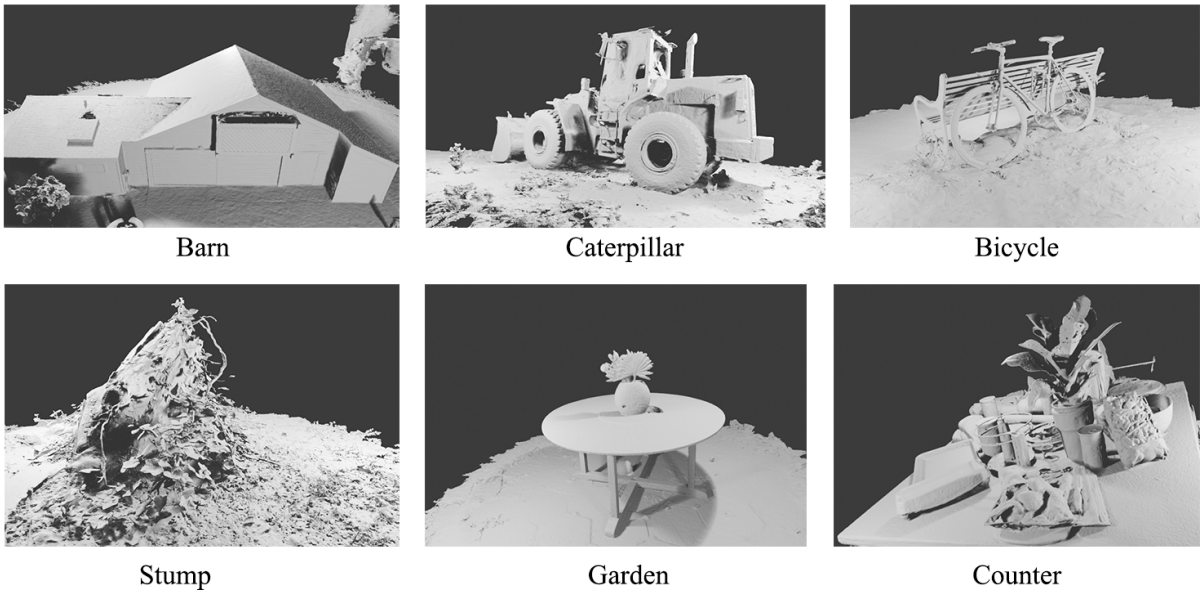


Figure 6. Qualitative geometry results for the Tanks and Temples dataset [7] and Mip-NeRF 360 dataset [1].

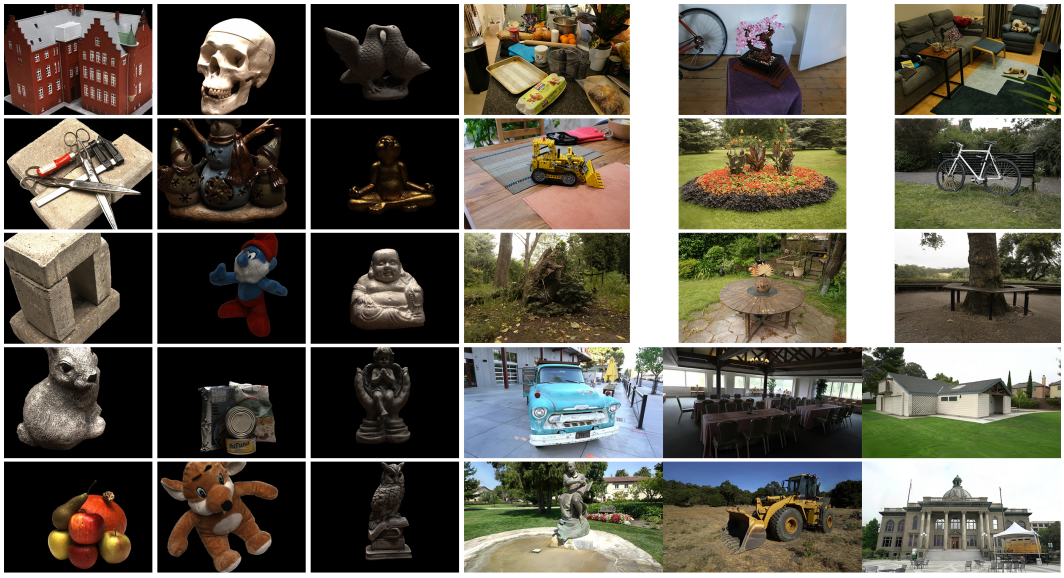


Figure 7. Qualitative appearance results for the DTU, TNT and Mip-NeRF 360 dataset.

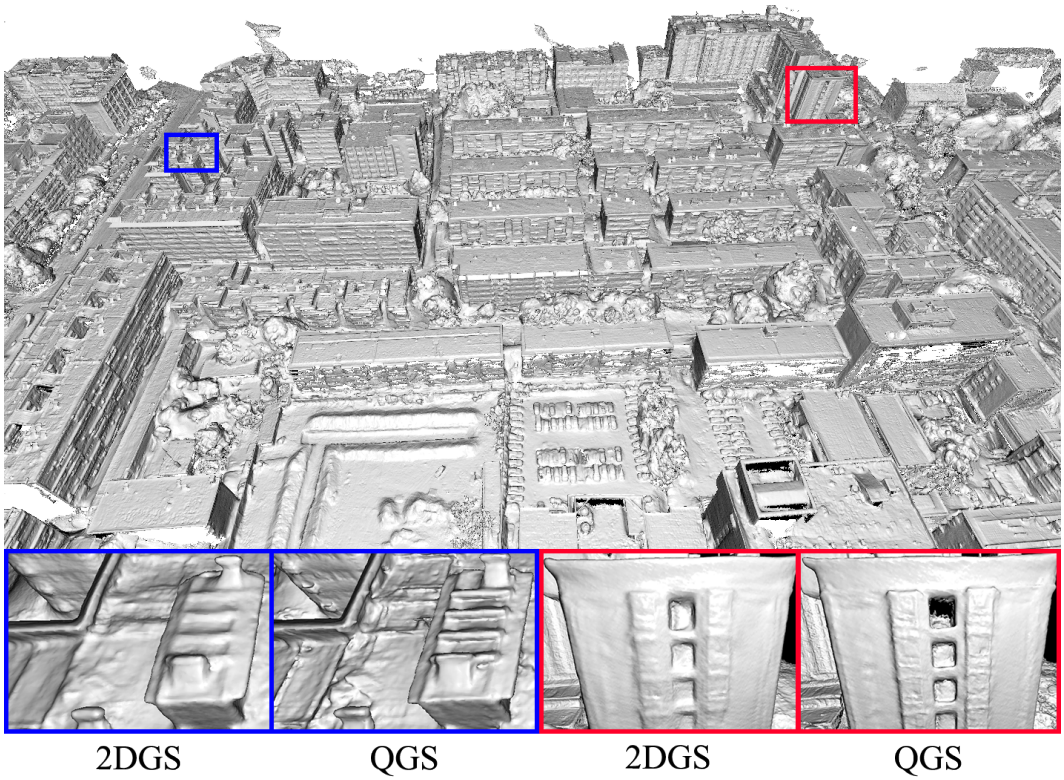


Figure 8. Qualitative geometry result for the urban scene captured from aerial view, involving over 1,000 images. In the subfigure, the left shows the results of 2DGS [4], while the right shows the results of QGS.

References

[1] 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, 4

[2] 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. 3, 4

[3] 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

[4] 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, 4, 5

[5] 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. 4

[6] 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

[7] 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, 4

[8] 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

[9] James Johnston Stoker. *Differential geometry*. John Wiley & Sons, 2011. 3

[10] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes. *arXiv preprint arXiv:2404.10772*, 2024. 3, 4