Geometry-Grounded Gaussian Splatting

BAOWEN ZHANG, Hong Kong University of Science and Technology, Hong Kong
CHENXING JIANG, Hong Kong University of Science and Technology, Hong Kong
HENG LI, Hong Kong University of Science and Technology, Hong Kong
SHAOJIE SHEN, Hong Kong University of Science and Technology, Hong Kong
PING TAN, Hong Kong University of Science and Technology, Hong Kong

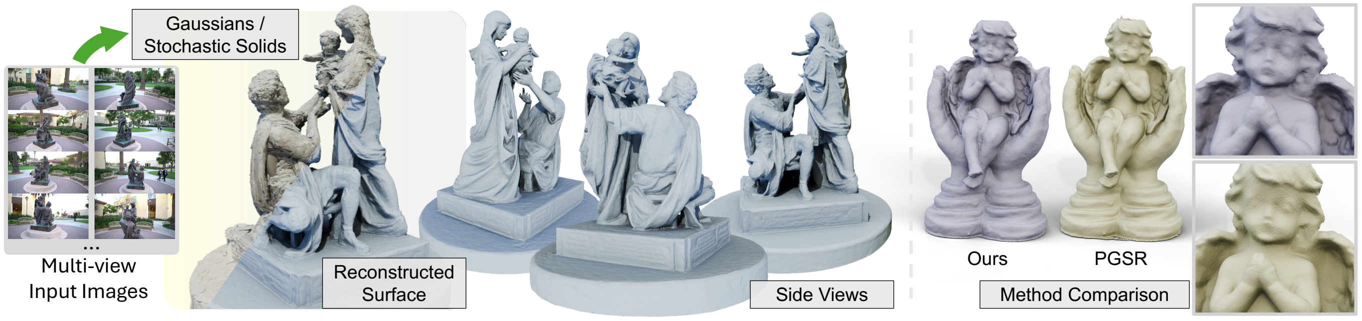

Fig. 1. We prove that Gaussian primitives are equivalent to stochastic solids, and leverage this equivalence to reconstruct high-fidelity, multi-view-consistent shapes from multi-view images.

Gaussian Splatting (GS) has demonstrated impressive quality and efficiency in novel view synthesis. However, shape extraction from Gaussian primitives remains an open problem. Due to inadequate geometry parameterization and approximation, existing shape reconstruction methods suffer from poor multi-view consistency and are sensitive to floaters. In this paper, we present a rigorous theoretical derivation that establishes Gaussian primitives as a specific type of stochastic solids. This theoretical framework provides a principled foundation for Geometry-Grounded Gaussian Splatting by enabling the direct treatment of Gaussian primitives as explicit geometric representations. Using the volumetric nature of stochastic solids, our method efficiently renders high-quality depth maps for fine-grained geometry extraction. Experiments show that our method achieves the best shape reconstruction results among all Gaussian Splatting-based methods on public datasets.

CCS Concepts: • **Computing methodologies → Point-based models**; **Volumetric models**; *Rendering*.

Additional Key Words and Phrases: Gaussian Splatting, Stochastic Solids, Shape Reconstruction

1 Introduction

3D shape reconstruction from multi-view images is a long-standing problem with broad impact in virtual reality [Snavely et al. 2006], autonomous driving [Schmied et al. 2023], and robotics [Cadena et al. 2017; Engel et al. 2014]. Recent progress has been driven by implicit neural representations, most notably NeRF [Mildenhall et al. 2020]. Many state-of-the-art methods further adopt geometry-grounded radiance fields: they start from a canonical geometry field (*e.g.*, SDF/occupancy) and derive the rendering formulation accordingly. Methods such as VolSDF [Yariv et al. 2021] and NeuS [Wang et al. 2021] follow this principle by anchoring the rendering to an explicit surface and yielding reliable geometry that is consistent across views. Despite these advances, geometry-grounded radiance fields typically rely on dense sampling, *e.g.*, ray marching, along camera rays, resulting in slow training and inference.

In contrast, Gaussian Splatting [Kerbl et al. 2023] represents scenes as a collection of Gaussian primitives and leverages efficient rasterization, enabling fast optimization and real-time novel view synthesis. Several follow-up works [Chen et al. 2024; Guédon et al. 2025b; Huang et al. 2024; Yu et al. 2024c; Zhang et al. 2024, 2025a] have extended Gaussian Splatting to shape reconstruction with promising results. Nevertheless, Gaussian Splatting does not inherently define a surface, unlike geometry-grounded NeRF methods that start from an SDF/occupancy field. Existing Gaussian Splatting–based methods therefore extract depth or surfaces from the Gaussian radiance field using heuristic rules. A more principled geometric formulation can improve cross-view consistency and enable higher-fidelity reconstruction, as shown in the right of Figure 1. Unlike these heuristic pipelines, we provide a principled geometric foundation for Gaussian primitives, enabling higher-fidelity shape reconstruction.

In this paper, we adopt the philosophy of geometry-grounded radiance fields by equipping Gaussians with a canonical geometry field. We achieve this by leveraging the theoretical foundation provided by the recent work 'Objects as Volumes' [Miller et al. 2024], which offers a stochastic interpretation of the geometry-grounded radiance field. Under this theory, we analyze the rendering equation of Gaussian Splatting and

Authors' Contact Information: Baowen Zhang, Hong Kong University of Science and Technology, Hong Kong, Hong Kong, bzhangcm@connect.ust.hk; Chenxing Jiang, Hong Kong University of Science and Technology, Hong Kong, Hong Kong, cjiangan@connect.ust.hk; Heng Li, Hong Kong University of Science and Technology, Hong Kong, Hong Kong, eehengli@ust.hk; Shaojie Shen, Hong Kong University of Science and Technology, Hong Kong, Hong Kong, eeshaojie@ust.hk; Ping Tan, Hong Kong University of Science and Technology, Hong Kong, Hong Kong, pingtan@ust.hk.

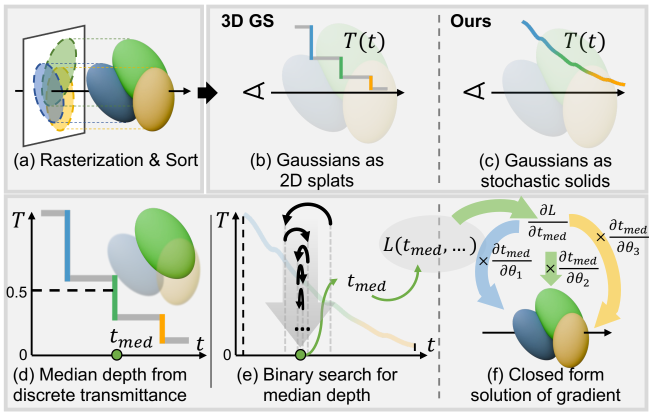
3D GS
Ours
$T(t)$
$T(t)$
(a) Rasterization & Sort
(b) Gaussians as 2D splats
(c) Gaussians as stochastic solids
T
0.5
0
t_{med}
t
(d) Median depth from discrete transmittance
(e) Binary search for median depth
$L(t_{med}, ...)$
$\frac{\partial L}{\partial t_{med}}$
$\times \frac{\partial t_{med}}{\partial \theta_1}$
$\times \frac{\partial t_{med}}{\partial \theta_2}$
$\times \frac{\partial t_{med}}{\partial \theta_3}$
(f) Closed form solution of gradient

Fig. 2. **Overview of our depth-rendering pipeline.** (a) We rasterize Gaussian primitives and sort them by depth. (b) Standard Gaussian Splatting yields step-wise transmittance under splat compositing. (c) Under our stochastic solid formulation, attenuation is modeled continuously within each primitive, yielding a smooth transmittance curve. (d) Prior work estimates the ray-wise median depth as the point where transmittance drops to 0.5. (e) Forward pass: we locate the median depth t_{med}, *i.e.*, $T = 0.5$, via binary search. (f) Backward pass: we backpropagate through t_{med} using a closed-form gradient with respect to all Gaussians contributing to the ray.

demonstrate that rendering a Gaussian primitive is identical to rendering a stochastic solid (Section 4.1). This unifies rendering formulations of Gaussian Splatting and NeRF-based methods, allowing us, for the first time, to derive a geometric field for Gaussian primitives. Using our formulation, we develop an efficient depth-rendering method that approximates the isosurface of the geometric field and extracts finer-grained geometry from Gaussian primitives (Section 4.2), exhibiting inherent multi-view consistency and robustness to floaters.

Figure 2 illustrates our depth-rendering pipeline to exemplify our advantages in detail. Prior Gaussian Splatting–based methods define the median depth along a ray as the location where transmittance drops to 0.5, as shown in Figure 2(d). However, because of discrete changes in transmittance, this method fails to capture the joint effect of overlapping Gaussians and leading to jagged depth steps. In contrast, the stochastic solids model volumetric attenuation continuously and yield a smooth transmittance curve. Building on this, we endow Gaussian primitives with the same continuous behavior, enabling more detailed depth maps. To compute the median depth, we exploit the monotonicity of transmittance and apply a binary search to locate the 0.5-transmittance crossing. We further derive a closed-form expression for the gradient of the median depth with respect to the parameters of all Gaussians along the ray for efficient backpropagation.

The main contributions of this paper are summarized as,

- We analyze the rendering equation of Gaussian Splatting and demonstrate that the Gaussian primitives can be regarded as stochastic solids, which provides theoretical guidance for shape reconstruction from Gaussian Splatting (Section 4.1).
- Based on this stochastic theory, we propose an efficient method for rendering and optimizing depth maps from Gaussian primitives, enabling accurate geometry extraction (Section 4.2).
- Extensive experiments demonstrate that our method achieves the best reconstruction accuracy among Gaussian Splatting-based methods, while maintaining optimization efficiency of the Gaussian Splatting (Section 5).

2 Related Work

2.1 Continuous Radiance Fields

NeRF [Mildenhall et al. 2020] models a scene as a continuous radiance field, typically parameterized by an MLP, and has shown strong performance in challenging effects such as reflections and scattering [Andrea et al. 2023; Levy et al. 2023; Tang et al. 2024]. Building on this backbone, Mip-NeRF introduces an anti-aliased multiscale formulation through conical-frustum rendering [Barron et al. 2021], and Mip-NeRF 360 extends it to unbounded scenes with specialized parameterization and regularization [Barron et al. 2022]. To improve efficiency, several works replace MLP ray marching with explicit volumetric parameterizations, *e.g.*, Plenoxel's voxel-grid optimization [Fridovich-Keil et al. 2022] and SVRaster's real-time rasterization of adaptive sparse voxels [Sun et al. 2025]; Instant-NGP further accelerates training with multiresolution hash grids [Müller et al. 2022].

While NeRF was originally designed for view synthesis, recovering accurate geometry from a generic density field is non-trivial, motivating surface-aware formulations that couple volume rendering with implicit surfaces. VolSDF [Yariv et al. 2021], NeuS [Wang et al. 2021], and UNISURF [Oechsle et al. 2021] parameterize density through a signed distance function (SDF) and design rendering weights to obtain more faithful surfaces. Neuralangelo further combines multiresolution hash-grid encodings with neural surface rendering to achieve high-fidelity reconstruction from RGB captures [Li et al. 2023]. GeoSVR [Li et al. 2025] explores explicit sparse voxels for geometrically accurate surface reconstruction, leveraging uncertainty-aware depth constraints and voxel surface regularization to improve detail and completeness. On the theoretical side, Objects as Volumes [Miller et al. 2024] provides a stochastic-geometry view of representing opaque solids as volumes and clarifies when exponential transmittance-based models are physically consistent, offering principled insights into surface-oriented volume rendering. Although these methods can reconstruct high-quality geometry, they generally suffer from extreme time consumption.

2.2 Primitive Based Representations

Gaussian Splatting [Kerbl et al. 2023] represents 3D scenes using a set of 3D Gaussian primitives. Combining with rasterization techniques, it avoids the time-consuming ray marching process in NeRF rendering. As a result, it achieves both real-time rendering and accelerated training. Building on this foundation, Mip-Splatting [Yu et al. 2024a] addresses aliasing by incorporating low-pass filters, while LightGaussian [Fan et al. 2024] optimizes memory usage with a compact representation. VastGaussian [Lin et al. 2024] further extends Gaussian Splatting to larger-scale scenes. StochasticSplats [Kheradmand et al. 2025] adopts a Monte Carlo estimator to enable sort-free rendering, further improving rendering efficiency.

Although 3DGS achieves high-quality novel-view synthesis, the geometry recovered from purely photometric optimization is often unreliable. To improve surface reconstruction, prior work either imposes stronger geometric priors or adds geometric supervision. SuGaR [Guédon and Lepetit 2024] and NeuGS [Chen et al. 2023] favor surface-aligned (flattened) Gaussians to better capture object boundaries and facilitate mesh extraction. Related approaches [Huang et al. 2024; Zhang et al. 2025b] replace 3D Gaussians with 2D primitives to encourage surface-like representations, although such constraints may reduce modeling flexibility and become unstable in complex scenes. GFSGS [Jiang et al. 2025] further leverages stochastic solids to construct 2D surfels for shape reconstruction. Beyond primitive design, 3DGSR [Lyu et al. 2024] and GSDF [Yu et al. 2024b] jointly optimize Gaussians with an implicit neural SDF field, improving reconstruction fidelity while retaining splatting efficiency, and PGSR [Chen et al. 2024] adds multi-view geometric regularization.

Despite promising empirical progress, geometry extraction in Gaussian Splatting still relies on heuristic depth definitions. These heuristics often yield noisy depth maps that have poor consistency across viewpoints and thus a weaker supervisory signal for optimization. This raises a fundamental question of whether Gaussian representations support an intrinsic notion of geometry akin to NeRF-based methods. We address it by adopting a stochastic approach to compute depth maps in a more principled manner for high-quality shape reconstruction.

3 Preliminary

3.1 Gaussian Splatting

We first briefly revisit Gaussian Splatting (GS). A 3D Gaussian primitive is defined as follows:

$$G(\mathbf{x}) = oe^{-(\mathbf{x}-\mathbf{x}_c)^\top \Sigma^{-1}(\mathbf{x}-\mathbf{x}_c)}, \tag{1}$$

where o is the opacity, $\Sigma \in \mathbf{R}^{3\times 3}$ is the covariance, $\mathbf{x} \in \mathbf{R}^3$ represents a point in 3D space, and $\mathbf{x}_c \in \mathbf{R}^3$ denotes the Gaussian's center. To enable fast rasterization, Gaussian Splatting (GS) methods employ a local affine approximation to project 3D Gaussian primitives to 2D Gaussians on the image plane with the covariance matrix Σ'_{2D} The opacity of the 2D Gaussian $\alpha(\mathbf{u})$ is defined as the maximum value of the projected 2D Gaussian:

$$\alpha(\mathbf{u}) = oe^{-(\mathbf{u}-\mathbf{u}_c)^\top \Sigma'^{-1}_{2D}(\mathbf{u}-\mathbf{u}_c)}, \tag{2}$$

where $\mathbf{u}$ is the coordinate of the pixels in the image space, $\mathbf{u}_c$ is the projected center of the Gaussian. In this way, 3D Gaussian primitives are projected into 2D Gaussians. These 2D Gaussians are then sorted and alpha-blended to compute the final color. More details can be found in the supplementary material.

3.2 Objects as Volumes

In this subsection, we provide a brief overview of [Miller et al. 2024], which presents a method to render stochastic solids using volume rendering. For a stochastic opaque solid characterized by its occupancy O and vacancy v, *i.e.*, $1 - O$, the authors derive the attenuation coefficient σ of the object as follows:

$$\sigma(\mathbf{x}, \omega) = |\omega \cdot \nabla log(\mathrm{v}(\mathbf{x}))| = \frac{|\omega \cdot \nabla \mathrm{v}(\mathbf{x})|}{\mathrm{v}(\mathbf{x})}, \tag{3}$$

where ω is the viewing direction and $\mathbf{x}$ is the 3D position. With this attenuation coefficient, they derive the volume rendering for a stochastic solid as,

$$\begin{aligned} \mathbf{C} &= \int_{t_n}^{t_f} p(t)\mathbf{c}(\mathbf{x}(t), \omega)\, dt, \\ p(t) &= T(t)\sigma(\mathbf{x}(t), \omega), \\ T(t) &= exp\left(-\int_{t_n}^{t} \sigma(\mathbf{x}(s), \omega)\, ds\right), \end{aligned} \tag{4}$$

where p is the free-flight distribution [Miller et al. 2024] that represents the statistical distribution of the distances that the light travels before collision and serves as the weight for color integration, and $T(t)$ is the transmittance along the ray.

In our work, we regard a 3D Gaussian primitive as a stochastic solid and design an appropriate attenuation coefficient σ for it. With this coefficient, the volume rendering of a Gaussian primitive, as described in Equation 4, is equivalent to its rasterized rendering. This enables us to study Gaussian Splatting in a more principled manner and develop a shape reconstruction method for Gaussian primitives.

4 Method

In the following sections, we first introduce our method for a single Gaussian primitive. We then design an efficient method for rendering depth maps from multiple Gaussian primitives.

4.1 Gaussian Primitives as Stochastic Solids

We treat a Gaussian primitive as a stochastic solid [Miller et al. 2024] and derive its rendering function. As shown in Figure 3, we prove that, with a proper attenuation coefficient σ, the volume rendering of this stochastic Gaussian solid is equivalent to the rasterization rendering of the original Gaussian Splatting. Specifically, the opacity α for a pixel in Equation 2 corresponds to the maximum value of the Gaussian function along that pixel's view ray (as proved in the supplementary). Therefore, the rendered color of a single Gaussian is given by:

$$\mathbf{C} = \mathbf{c}\alpha = \mathbf{c}G(t^*), \tag{5}$$

where t^* is the maximum point along the ray $l : \mathbf{o} + \omega t$, and we denote $G(\mathbf{o} + \omega t^*)$ by $G(t^*)$ for simplification.

Equation 5 cannot uniquely determine the attenuation coefficient. So, we impose three additional constraints. Given a Gaussian primitive $G(\mathbf{x})$, we assume that

- When $G(\mathbf{x}_1) \geq G(\mathbf{x}_2)$, it follows that $\mathrm{o}(\mathbf{x}_1) \geq \mathrm{o}(\mathbf{x}_2)$, indicating that positions closer to the Gaussian center have higher occupancy;

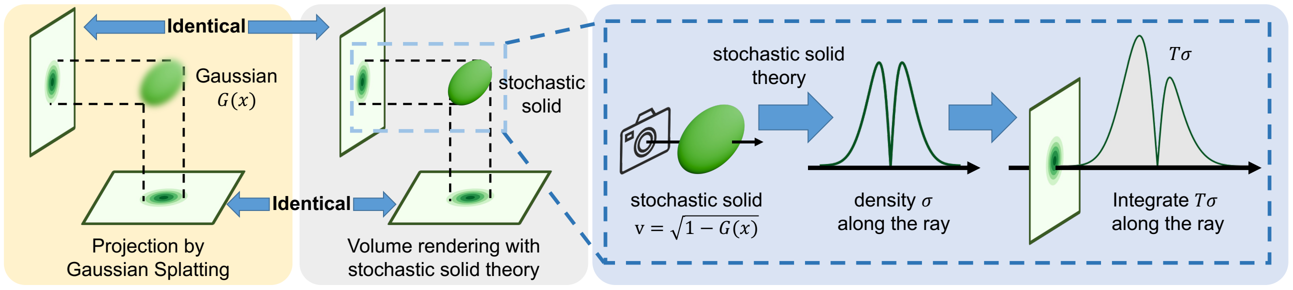
Identical
Gaussian
$G(x)$
Identical
Projection by Gaussian Splatting
stochastic solid
Volume rendering with stochastic solid theory
stochastic solid theory
stochastic solid
$v = \sqrt{1 - G(x)}$
density σ along the ray
$T\sigma$
Integrate $T\sigma$ along the ray

Fig. 3. Given a Gaussian primitive, we regard it as a stochastic solid and derive an appropriate attenuation coefficient with Equation 3. With our attenuation coefficient, the volume rendering of this stochastic solid is equivalent to the rasterization rendering developed in the original Gaussian Splatting.

– The occupancy of the solid approaches 0 when $\mathbf{x}$ is far from the Gaussian center;
– The occupancy $o(\mathbf{x})$ is differentiable from $\mathbf{x}$.

This leads us to derive a straightforward and unique expression for the vacancy:

$$v(\mathbf{x}) = \sqrt{1 - G(\mathbf{x})}. \tag{6}$$

To prove Equation 6, we first derive the volume rendering of a Gaussian primitive following the method in [Miller et al. 2024] as,

$$\mathbf{C} = \int_{-\infty}^{\infty} T(t)\sigma(\mathbf{x}(t), \omega)\mathbf{c}\, dt = \mathbf{c}(1 - v(t^*)^2), \tag{7}$$

where the attenuation coefficient σ originates from the stochastic solid as described in Equation 3, resulting in the integral in terms of vacancy $v(t^*)$. Compared with the rasterization in Equation 5 and the volumetric rendering in Equation 7, we can obtain,

$$\mathbf{C} = \mathbf{c}G(t^*) = \mathbf{c}(1 - v(t^*)^2), \tag{8}$$

In other words, we derive the following condition,

$$v(t^*) = \sqrt{1 - G(t^*)}. \tag{9}$$

Therefore, a stochastic Gaussian solid can produce the same rendering results as the rasterization in Gaussian Splatting if its vacancy adheres to Equation 6. The proof of uniqueness and other details can be found in our supplementary materials. Now, we can use Equation 3 and Equation 6 to obtain attenuation coefficients σ inside a Gaussian primitive, allowing us to obtain accurate depth maps and smooth optimization.

This property lets us move beyond heuristic geometry readouts, leading to a principled shape reconstruction approach built directly on Gaussian primitives. In the following sections, we apply this theory to Gaussian Splatting and demonstrate that it substantially improves shape reconstruction.

4.2 Depth from Stochastic Solids

In Gaussian Splatting, photometric supervision alone is insufficient to reconstruct high-quality shapes. To better recover surface geometry, recent works [Chen et al. 2024; Guédon and Lepetit 2024; Huang et al. 2024] render depth maps from Gaussian primitives and add geometric regularizers, and then backpropagate their gradients to the Gaussian parameters. Nevertheless, the rendered depth maps are noisy and have poor cross-view consistency, *e.g.*, as shown in Figure 8 and 4, providing weak geometric supervision. This motivates us to improve depth rendering in Gaussian Splatting by utilizing attenuation coefficients derived from stochastic solids.

The rendering pipeline is shown in Figure 2. We first derive our depth computation method, then show that it improves multi-view consistency and produces cleaner depth maps.

4.2.1 Depth definition. Following prior Gaussian Splatting methods, we use the median depth t_{med} for geometric regularization:

$$t_{med} = T^{-1}(0.5), \tag{10}$$

where $T^{-1}(*)$ is the inverse function of the transmittance $T(t)$. Following prior work [Blanc et al. 2025a,b; Condor et al. 2025], we assume that the events of a view ray intersecting different Gaussians are statistically independent. Under this assumption, the overall transmittance at t along the ray is the product of the transmittance calculated at each Gaussian primitive as,

$$T(t) = \prod_i T_i(t), \tag{11}$$

where $T_i(t)$ is the transmittance of the i-th Gaussian as:

$$T_i(t) = \begin{cases} v_i(t), & t \le t_i^* \\ v_i(t_i^*)^2 / v_i(t), & t > t_i^*. \end{cases} \tag{12}$$

Here, t_i^* is the Gaussian's maximum point along the camera ray. Equation 12 is derived from the continuous attenuation profile within each Gaussian as defined in Equation 3, capturing more detailed geometry information. The derivation of Equation 12 can be found in the supplementary material.

Discussion. Previous methods estimate depth either from per-view depth planes [Yu et al. 2024c; Zhang et al. 2024] that are view-dependent by design, or via opacity-weighted ray averaging [Chen et al. 2024] that is easily biased by view-specific floaters. These depth extraction strategies often lead to poor cross-view consistency. In contrast, we will show that interpreting Gaussian Splatting as a stochastic solid yields a median depth estimate with strong multi-view consistency. Recall that the median depth is the point where the transmittance first reaches a fixed threshold, *i.e.*, $T = 0.5$. From Equations 12 and 11, if the overall transmittance crossing $T = 0.5$ occurs before the

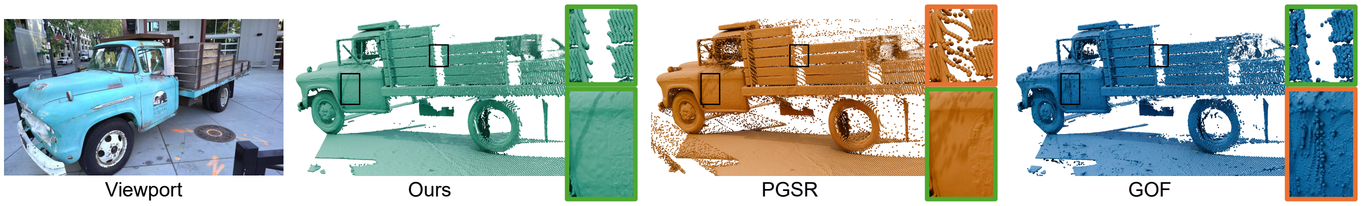

Fig. 4. **Depth maps of Gaussian Splatting-based methods.** We visualize the depth maps by converting them to 3D points. Our method produces a clean and smooth depth map. PGSR uses expected depth, yielding much noise at edges. GOF uses median depth and suffers from unsmooth depth changes.

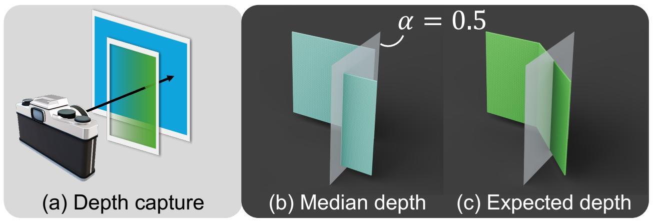

Fig. 5. **Illustration of depth rendering methods.** (a) The green plane's opacity α decreases smoothly from 1 on the right to 0 on the left. Consequently, (b) the median depth changes in a step-like manner, whereas (c) the expected depth varies continuously.

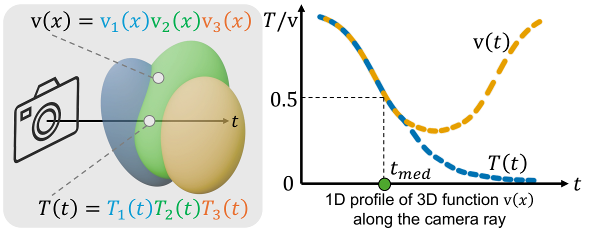
$\mathrm{v}(x) = \mathrm{v}_1(x)\mathrm{v}_2(x)\mathrm{v}_3(x)$
t
$T(t) = T_1(t)T_2(t)T_3(t)$
T/v
$\mathrm{v}(t)$
0.5
0
t_{med}
$T(t)$
t
1D profile of 3D function $\mathrm{v}(x)$ along the camera ray

Fig. 6. **Illustration of vacancy along the camera ray.** The vacancy value on the ray equals the transmittance on the front side of the Gaussians.

peak of the contributing Gaussians, then the transmittance coincides with the 3D vacancy field as illustrated in Figure 6. So, the depth map is a view-independent 0.5-level isosurface. This regime is common because optimization clusters high-opacity Gaussians near the surface, making the transmittance drop mainly on their near sides. While floaters can still perturb the ray, the median depth is more robust to such outliers than the alpha-averaged expected depth, leading to stronger multi-view consistency.

Beyond improved multi-view consistency, our method produces cleaner depth maps as shown in Figure 4. Depth obtained via alpha-weighted compositing tends to interpolate between foreground and background at boundaries, leading to blurred silhouettes as shown in Figure 5. Median depth, defined by the $T = 0.5$ crossing, gives sharper boundary transitions. However, in prior Gaussian Splatting formulations, transmittance is updated in discrete steps, so the 0.5 crossing often snaps to a single Gaussian; neighboring pixels may therefore select different Gaussians, producing jagged artifacts. Our stochastic-solid formulation models attenuation continuously within each Gaussian, yielding a smooth transmittance function and reducing staircasing while preserving sharp boundaries.

4.2.2 Implementation. In general, Equation 10 does not admit a closed-form solution. To address this, we exploit the monotonicity of transmittance along each ray and use an iterative binary search to find the median depth. During backpropagation, we do not require an iterative search. Instead, we derive the closed-form solution for the gradient of depth t_{med} with respect to the Gaussians' parameters as,

$$\frac{\partial t_{med}}{\partial \theta} = -\frac{\partial T(t_{med};\theta)}{\partial \theta} \Big/ \frac{\partial T(t;\theta)}{\partial t}\Big|_{t=t_{med}}, \tag{13}$$

where θ denotes the Gaussian parameters along the ray.

Equation 13 shows that the gradient can be distributed to all contributing Gaussians along the ray, unlike previous methods where the gradient of the median depth was only applied to a single Gaussian. This stems from our stochastic-solid formulation, which yields a differentiable transmittance function. As a result, the median depth t_{med} varies smoothly with the Gaussian parameters, providing denser supervision for optimization. The derivation of Equation 13 and implementation details are provided in the supplementary material.

4.3 Optimization with Stochastic Solids

We optimize scenes using photometric loss [Kerbl et al. 2023], normal consistency loss [Huang et al. 2024], and multi-view regularization [Chen et al. 2024]; details are provided in the supplementary material. These losses require rendering RGB images, normal maps, and depth maps. Fully volumetric rendering for all modalities is computationally expensive [Blanc et al. 2025a,b; Condor et al. 2025]. We therefore retain the standard Gaussian Splatting approximation for RGB and normals [Zhang et al. 2024], while computing depth using Eq. 10. Experiments show that this setting can significantly improve the shape reconstruction accuracy of Gaussian Splatting, while maintaining the efficiency. Nevertheless, we believe that extending our volumetric formulation to RGB and normal rendering can further improve accuracy, which we leave for future work.

5 Experiments

We evaluate our method on several public datasets and compare it with existing state-of-the-art methods.

Table 1. Quantitative comparison on the DTU dataset [Jensen et al. 2014]. We report Chamfer Distance and average optimization time for different methods. Among explicit Gaussian Splatting–based approaches, our method achieves the best results and attains accuracy comparable to GeoSVR. All Gaussian Splatting methods are evaluated using half-resolution images.

		24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean	Time
implicit	NeRF [Mildenhall et al. 2020]	1.90	1.60	1.85	0.58	2.28	1.27	1.47	1.67	2.05	1.07	0.88	2.53	1.06	1.15	0.96	1.49	> 12h
	VolSDF [Yariv et al. 2021]	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
	NeuS [Wang et al. 2021]	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
	Neuralangelo [Li et al. 2023]	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	> 12h
explicit	3D GS [Kerbl et al. 2023]	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	7.8m
	2D GS [Huang et al. 2024]	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	11.3m
	GOF [Yu et al. 2024c]	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	52m
	3DGSR [Lyu et al. 2024]	0.44	0.96	0.40	0.36	1.02	0.80	0.64	1.20	1.08	0.97	0.54	0.72	0.37	0.52	0.42	0.70	\
	RaDe-GS [Zhang et al. 2024]	0.43	0.75	0.35	0.37	0.81	0.74	0.74	1.19	1.20	0.65	0.61	0.84	0.35	0.66	0.46	0.68	8.2m
	GFSGS [Jiang et al. 2025]	0.40	0.59	0.39	0.38	0.72	0.59	0.65	1.08	0.93	0.59	0.50	0.67	0.34	0.47	0.40	0.58	16.8m
	PGSR [Chen et al. 2024]	0.34	0.54	0.44	0.37	0.78	0.57	0.49	1.06	0.63	0.59	0.47	0.50	0.30	0.37	0.34	0.52	30.5m
	GeoSVR [Li et al. 2025]	0.32	0.51	0.30	0.33	0.71	0.48	0.42	1.03	0.62	0.56	0.33	0.46	0.30	0.34	0.32	0.47	53.3m
	Ours (20k)	0.38	0.50	0.27	0.31	0.80	0.43	0.42	1.04	0.64	0.52	0.31	0.56	0.30	0.31	0.33	0.47	15.0m
	Ours (30k)	0.37	0.50	0.27	0.31	0.81	0.43	0.42	1.05	0.64	0.52	0.32	0.58	0.30	0.31	0.33	0.48	25.3m

Table 2. **Quantitative comparison on the Tanks & Temples Dataset [Knapitsch et al. 2017]**. We report the F1-score and average optimization time.

		Barn	Cat.	Cour.	Igna.	Meet.	Truc.	Mean	Time
implicit	NeuS	0.29	0.29	0.17	0.83	0.24	0.45	0.38	>24h
	Geo-NeuS	0.33	0.26	0.12	0.72	0.20	0.45	0.35	>24h
	Neurlangelo	0.70	0.36	0.28	0.89	0.32	0.48	0.50	>24h
explicit	2D GS	0.36	0.23	0.13	0.44	0.16	0.26	0.30	15.5m
	GOF	0.51	0.41	0.28	0.68	0.28	0.59	0.46	71.6m
	RaDe-GS	0.49	0.36	0.27	0.72	0.27	0.61	0.45	12.1m
	PGSR	0.66	0.44	0.20	0.81	0.33	0.66	0.52	42.9m
	GeoSVR	0.68	0.49	0.34	0.83	0.37	0.66	0.56	66.4m
	Ours	0.70	0.56	0.38	0.81	0.42	0.70	0.60	32.1m

Implementation Details. We use a local affine approximation and adopt RaDe-GS [Zhang et al. 2024] to estimate each Gaussian peak t_i^*. For efficiency, we follow gsplat [Ye et al. 2025] and use warp-level reductions for gradient accumulation. We apply the 3D filtering from Mip-Splatting [Yu et al. 2024a] (without its 2D filter), the densification strategy from GOF [Yu et al. 2024c], and the exposure compensation from PGSR [Chen et al. 2024]. Multi-view regularization is implemented with a custom CUDA kernel. We will release our code.

Datasets. We evaluate reconstruction accuracy on DTU [Jensen et al. 2014] and Tanks & Temples (TnT) [Knapitsch et al. 2017]. Following prior work, we use the standard 15-scene DTU split and the common 6-scene TnT subset. We report Chamfer Distance on DTU and F1-score on TnT.

Mesh Extraction. Following previous works [Yu et al. 2024c; Zhang et al. 2024], we apply the TSDF fusion [Curless and Levoy 1996] implemented by Open3D [Zhou et al. 2018] to extract meshes for the DTU dataset and adopt Marching Tetrahedra [Guédon et al. 2025a; Yu et al. 2024c] for large-scale scenes in the Tanks & Temples dataset. Inspired by GOF, we define an indicator function over 3D space for Marching Tetrahedra. Specifically, a point is classified as inside the mesh if it is occluded in any training view, *i.e.*, if its transmittance falls below 0.5; otherwise, it is classified as outside.

5.1 Reconstruction Comparison

We compare our method against existing state-of-the-art methods in the shape reconstruction task. Table 1 and Table 2 show the accuracy on the DTU and TnT datasets. The multi-view regularizer adopted in PGSR and GeoSVR substantially boosts DTU accuracy; using this regularization, our method achieves comparable performance to both. In TnT, our method significantly outperforms existing Gaussian Splatting-based methods because of our depth-rendering formulation, which enables finer geometric details, enforces view-consistent geometry, and is robust to floaters. Figure 7 provides qualitative comparisons among shape reconstruction methods. Our method reconstructs finer details with more accurate geometry. Additional qualitative results are shown in Figure 9 and Figure 10.

We report runtimes in Table 1 and Table 2. For the same number of iterations, our method is faster than GeoSVR (15 vs. 53 min.) and PGSR (25 vs. 30 min.), thanks to a more efficient implementation of multi-view regularization. Our runtime is higher than the fastest baselines due to the added cost of the binary search and the multi-view term. We expect further speedups by tightening the initial depth interval of the binary search, which we leave for future work.

5.2 Multi-view Consistency

Geometric consistency across views is essential for accurate shape reconstruction. To evaluate each depth-rendering method, we compute per-pixel cycle reprojection error during training. For a reference and neighboring view, we render depth maps of the reference view, back-project pixels to 3D, project into the neighbor to sample the corresponding depth, then back-project and reproject to the reference. The cycle error is the Euclidean distance between the original and reprojected pixel locations.

We compare our method with PGSR [Chen et al. 2024] and RaDe-GS [Zhang et al. 2024]. PGSR defines the ray-surface intersection using a plane orthogonal to the shortest axis of each 3D Gaussian, whereas RaDe-GS uses the ray-wise maximizer of the Gaussian response. For a fair comparison, we evaluate RaDe-GS augmented with the same multi-view regularization used in PGSR, and enable geometric regularization at 7K

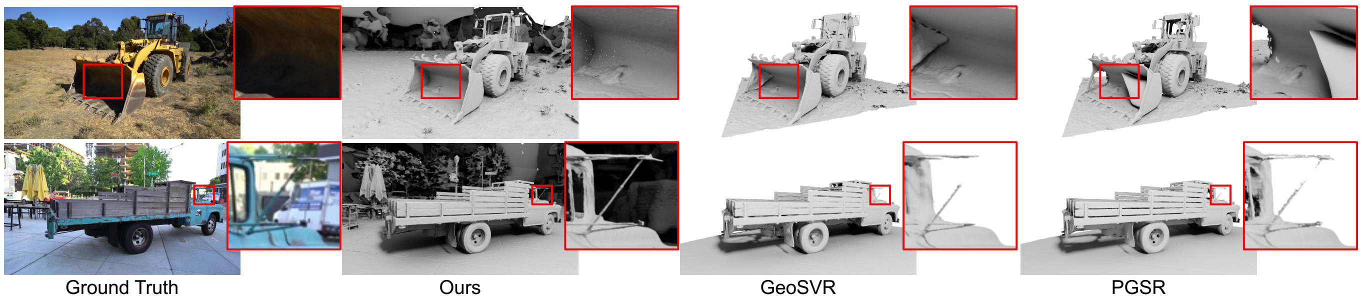

Fig. 7. **Qualitative comparison on Tanks & Temples [Knapitsch et al. 2017] dataset.** We compare our method with GeoSVR and PGSR. Our method reconstructs plausible meshes with finer geometric details

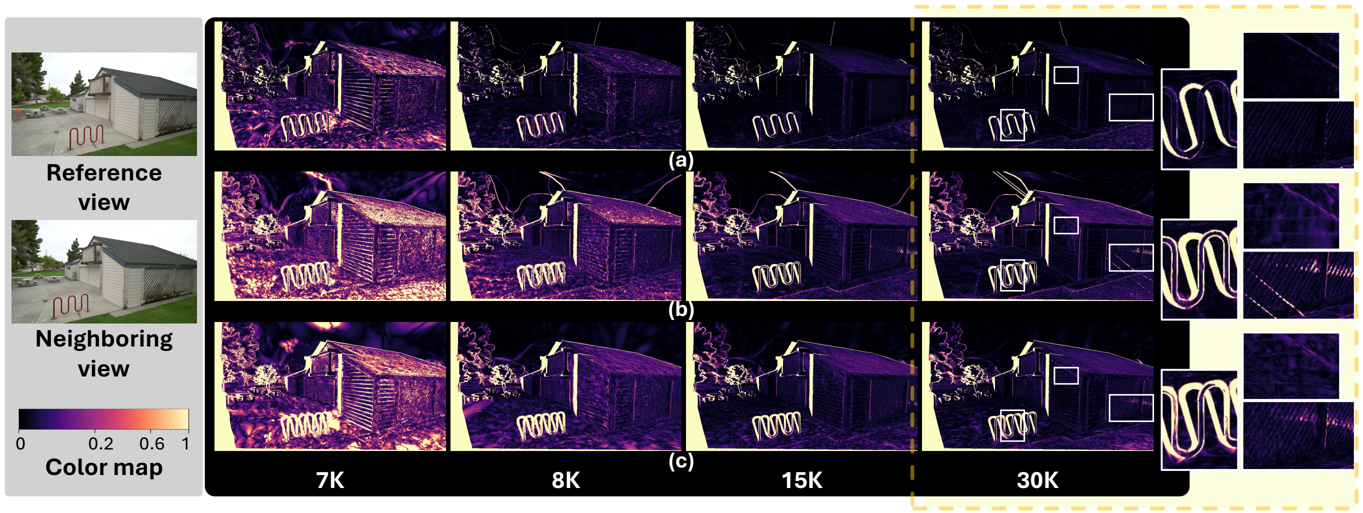

Fig. 8. **Cycle reprojection error per iteration.** We visualize the cycle reprojection error between a reference view and its nearest neighboring view throughout optimization. Our method (a) attains lower errors on foreground regions and achieves full coverage faster than the other methods. PGSR (b) relies on planar-based depth accumulation, which leads to weaker multi-view consistency than our volumetric formulation and suffers from noticeable floaters. We add the multi-view regularization to RaDe-GS (c) and train it using expected depth. Zoomed-in patches are shown on the right.

iterations for all methods. As shown in Figure 8, our method yields a better initialization and converges faster, achieving the lowest reprojection error at 30K iterations, mainly due to our depth formulation based on stochastic theory. In contrast, the other methods exhibit noticeably larger reprojection errors, and floaters that arise during training further exacerbate the inconsistency. More results can be found in Figure 13.

5.3 Ablation Study

In this section, we evaluate the contribution of each component when integrated into our method. Table 3 reports quantitative results on the TnT dataset. The geometric multi-view term L_{gc} penalizes cycle reprojection error; however, it brings only marginal gains in our setting, because our depth-rendering formulation already provides strong multi-view consistency. In contrast, the normal consistency loss and the exposure compensation module consistently improve reconstruction quality. Finally, compared with the other two depth-rendering baselines equipped with similar regularizers, our method achieves a more accurate shape reconstruction.

Table 3. **Ablation study on Tanks & Temples [Knapitsch et al. 2017].** The normal consistency loss and the single-view geometric loss in PGSR have a similar formulation. We denote them as L_n. $L_{gc}$is the geometric consistency loss. 'exposure' represents the exposure compensation module from PGSR. We toggle each term on/off (✓/–) and report the resulting reconstruction accuracy.

	PGSR	RaDe-GS	Ours			
L_{gc}	✓	✓	–	✓	✓	✓
L_n	✓	✓	✓	–	✓	✓
exposure	✓	✓	✓	✓	–	✓
F1-score	0.52	0.52	0.60	0.57	0.59	0.60

6 Conclusion

We reveal the intrinsic geometry for Gaussian Splatting. We regard Gaussian primitives as stochastic solids and design an appropriate attenuation function to make their volume rendering identical to their rasterization-based rendering. The stochastic theory enables depth rendering in a principled manner. Experiments show that our method outperforms state-of-the-art methods.

References

Ramazzina Andrea, Bijelic Mario, Walz Stefanie, Sanvito Alessandro, Scheuble Dominik, and Heide Felix. 2023. ScatterNeRF: Seeing Through Fog with Physically-Based Inverse Neural Rendering. *The IEEE International Conference on Computer Vision (ICCV)*.

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. *ICCV* (2021).

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *CVPR* (2022).

Hugo Blanc, Jean-Emmanuel Deschaud, and Alexis Paljic. 2025a. Raygauss: Volumetric gaussian-based ray casting for photorealistic novel view synthesis. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1808–1817.

Hugo Blanc, Jean-Emmanuel Deschaud, and Alexis Paljic. 2025b. RayGaussX: Accelerating Gaussian-Based Ray Marching for Real-Time and High-Quality Novel View Synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 27575–27584.

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. 2017. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics* 32, 6 (2017), 1309–1332.

Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. 2024. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics* (2024).

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

Jorge Condor, Sebastien Speierer, Lukas Bode, Aljaz Bozic, Simon Green, Piotr Didyk, and Adrian Jarabo. 2025. Don't Splat your Gaussians: Volumetric Ray-Traced Primitives for Modeling and Rendering Scattering and Emissive Media. *ACM Trans. Graph.* (Jan 2025). https://doi.org/10.1145/3711853

Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 303–312.

Jakob Engel, Thomas Schöps, and Daniel Cremers. 2014. LSD-SLAM: Large-scale direct monocular SLAM. In *European conference on computer vision*. Springer, 834–849.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. 2024. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems* 37 (2024), 140138–140158.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5501–5510.

Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. 2025a. MILo: Mesh-In-the-Loop Gaussian Splatting for Detailed and Efficient Surface Reconstruction. *ACM Transactions on Graphics* (2025). https://anttwo.github.io/milo/

Antoine Guédon, Tomoki Ichikawa, Kohei Yamashita, and Ko Nishino. 2025b. MAtCha Gaussians: Atlas of Charts for High-Quality Geometry and Photorealism From Sparse Views. In *CVPR*. 6001–6011.

Antoine Guédon and Vincent Lepetit. 2024. 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*. 5354–5363.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. (2024). https://doi.org/10.1145/3641519.3657428

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

Kaiwen Jiang, Venkataram Sivaram, Cheng Peng, and Ravi Ramamoorthi. 2025. Geometry Field Splatting with Gaussian Surfels. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 5752–5762.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Shakiba Kheradmand, Delio Vicini, George Kopanas, Dmitry Lagun, Kwang Moo Yi, Mark Matthews, and Andrea Tagliasacchi. 2025. StochasticSplats: Stochastic Rasterization for Sorting-Free 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Transactions on Graphics* 36, 4 (2017).

Deborah Levy, Amit Peleg, Naama Pearl, Dan Rosenbaum, Derya Akkaynak, Simon Korman, and Tali Treibitz. 2023. SeaThru-NeRF: Neural Radiance Fields in Scattering Media. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 56–65.

Jiahe Li, Jiawei Zhang, Youmin Zhang, Xiao Bai, Jin Zheng, Xiaohan Yu, and Lin Gu. 2025. GeoSVR: Taming Sparse Voxels for Geometrically Accurate Surface Reconstruction. *Advances in Neural Information Processing Systems* (2025).

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

Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, and Wenming Yang. 2024. VastGaussian: Vast 3D Gaussians for Large Scene Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5166–5175.

Xiaoyang Lyu, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi. 2024. 3dgsr: Implicit surface reconstruction with 3d gaussian splatting. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–12.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Bailey Miller, Hanyu Chen, Alice Lai, and Ioannis Gkioulekas. 2024. Objects as Volumes: A Stochastic Geometry View of Opaque Solids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 87–97.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530127

Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5589–5599.

Aron Schmied, Tobias Fischer, Martin Danelljan, Marc Pollefeys, and Fisher Yu. 2023. R3D3: Dense 3D Reconstruction of Dynamic Scenes from Multiple Cameras. In *Proceedings of the IEEE International Conference on Computer Vision*.

Noah Snavely, Steven M Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. In *ACM siggraph 2006 papers*. 835–846.

Cheng Sun, Jaesung Choe, Charles Loop, Wei-Chiu Ma, and Yu-Chiang Frank Wang. 2025. Sparse Voxels Rasterization: Real-time High-fidelity Radiance Field Rendering. In *CVPR*.

Yunkai Tang, Chengxuan Zhu, Renjie Wan, Chao Xu, and Boxin Shi. 2024. Neural Underwater Scene Representation. *CVPR*.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *NeurIPS* (2021).

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. *Advances in neural information processing systems* 34 (2021), 4805–4815.

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. 2025. gsplat: An open-source library for Gaussian splatting. *Journal of Machine Learning Research* 26, 34 (2025), 1–17.

Mulin Yu, Tao Lu, Linning Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai. 2024b. Gsdf: 3dgs meets sdf for improved neural rendering and reconstruction. *Advances in Neural Information Processing Systems* 37 (2024), 129507–129530.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2024a. Mip-Splatting: Alias-free 3D Gaussian Splatting. *Conference on Computer Vision and Pattern Recognition (CVPR)* (2024).

Zehao Yu, Torsten Sattler, and Andreas Geiger. 2024c. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (ToG)* 43, 6 (2024), 1–13.

Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. 2024. RaDe-GS: Rasterizing Depth in Gaussian Splatting. *arXiv preprint arXiv:2406.01467* (2024).

Ziyu Zhang, Binbin Huang, Hanqing Jiang, Liyang Zhou, Xiaojun Xiang, and Shuhan Shen. 2025a. Quadratic Gaussian splatting: High quality surface reconstruction with second-order geometric primitives. In *ICCV*. 28260–28270.

Ziyu Zhang, Binbin Huang, Hanqing Jiang, Liyang Zhou, Xiaojun Xiang, and Shuhan Shen. 2025b. Quadratic Gaussian Splatting: High Quality Surface

Reconstruction with Second-order Geometric Primitives. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 28260–28270.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).

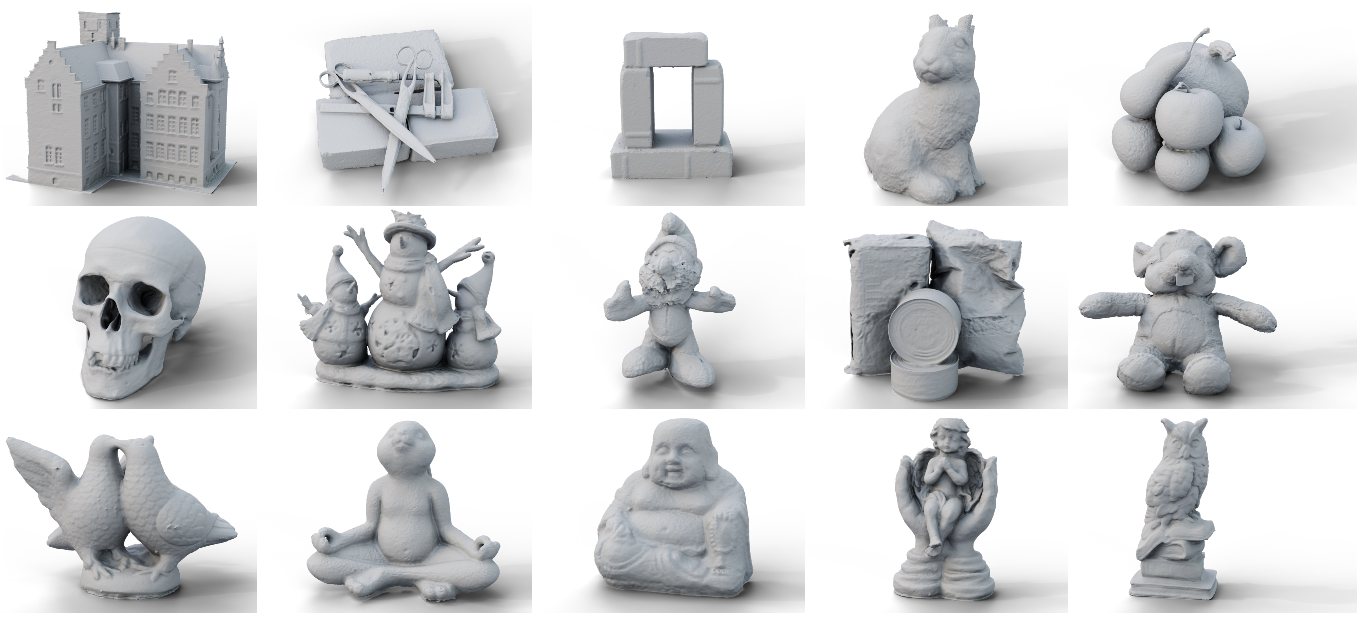

Fig. 9. **Qualitative results on the DTU [Jensen et al. 2014] dataset.**

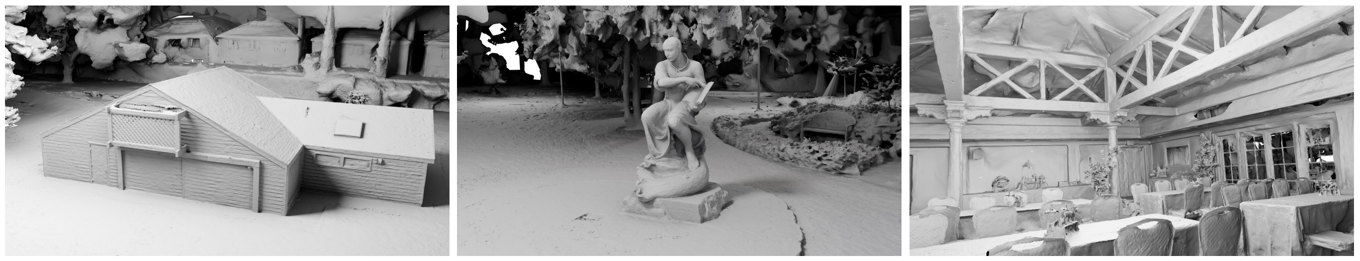

Fig. 10. **Qualitative results on the Tanks & Temples [Knapitsch et al. 2017] dataset.**

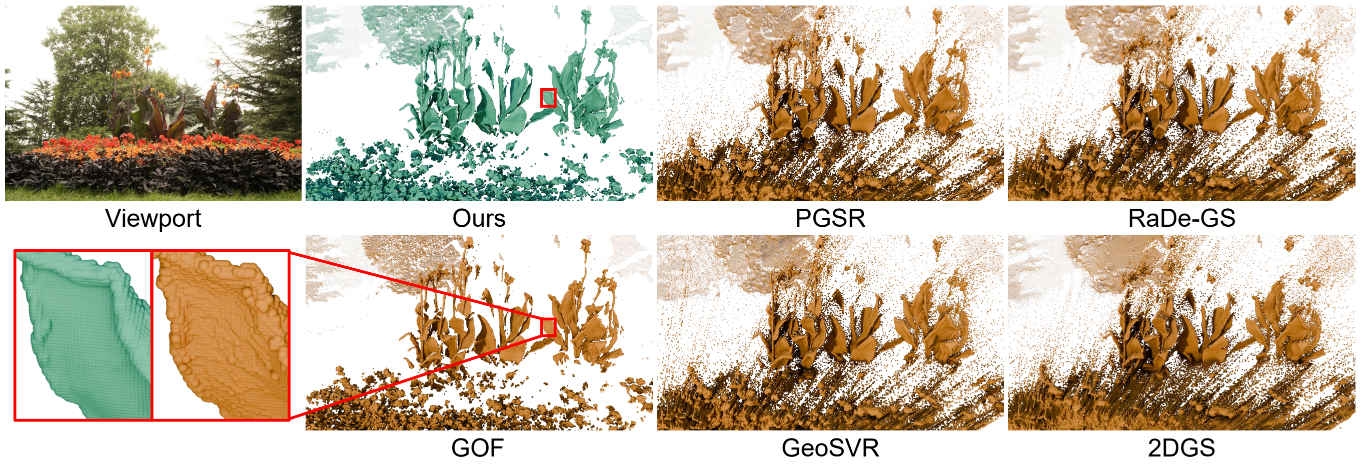

Fig. 11. **Qualitative comparison of depth rendering among our method and prior methods.** We visualize depth maps by back-projecting them into 3D point clouds. RaDe-GS uses multi-view regularization and expected depth; 2DGS uses expected depth.

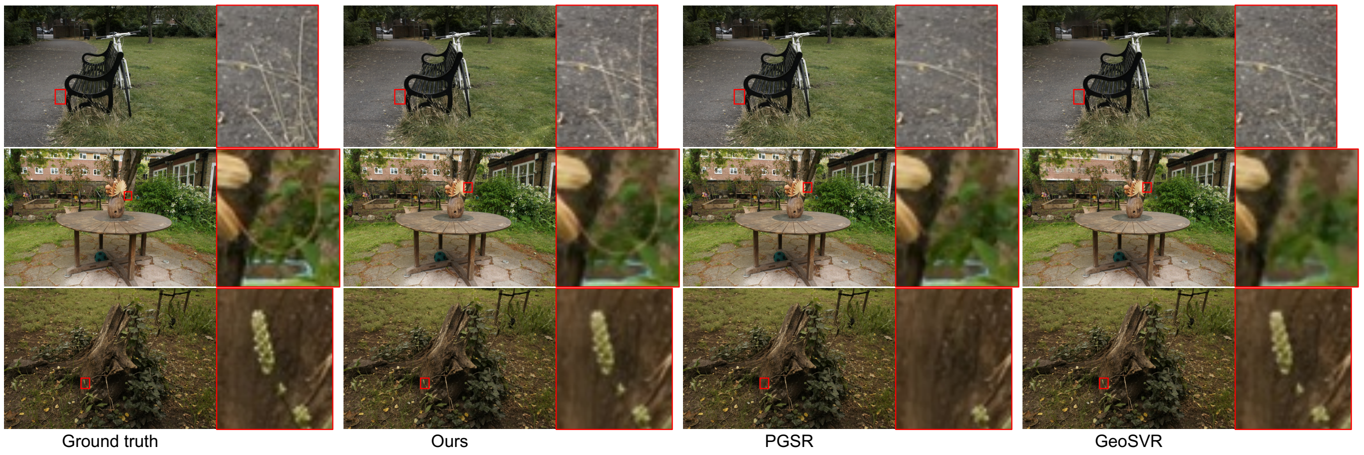

Fig. 12. **Qualitative comparison of novel view synthesis among our method and prior methods on the Mip-NeRF360 [Barron et al. 2022] dataset.**

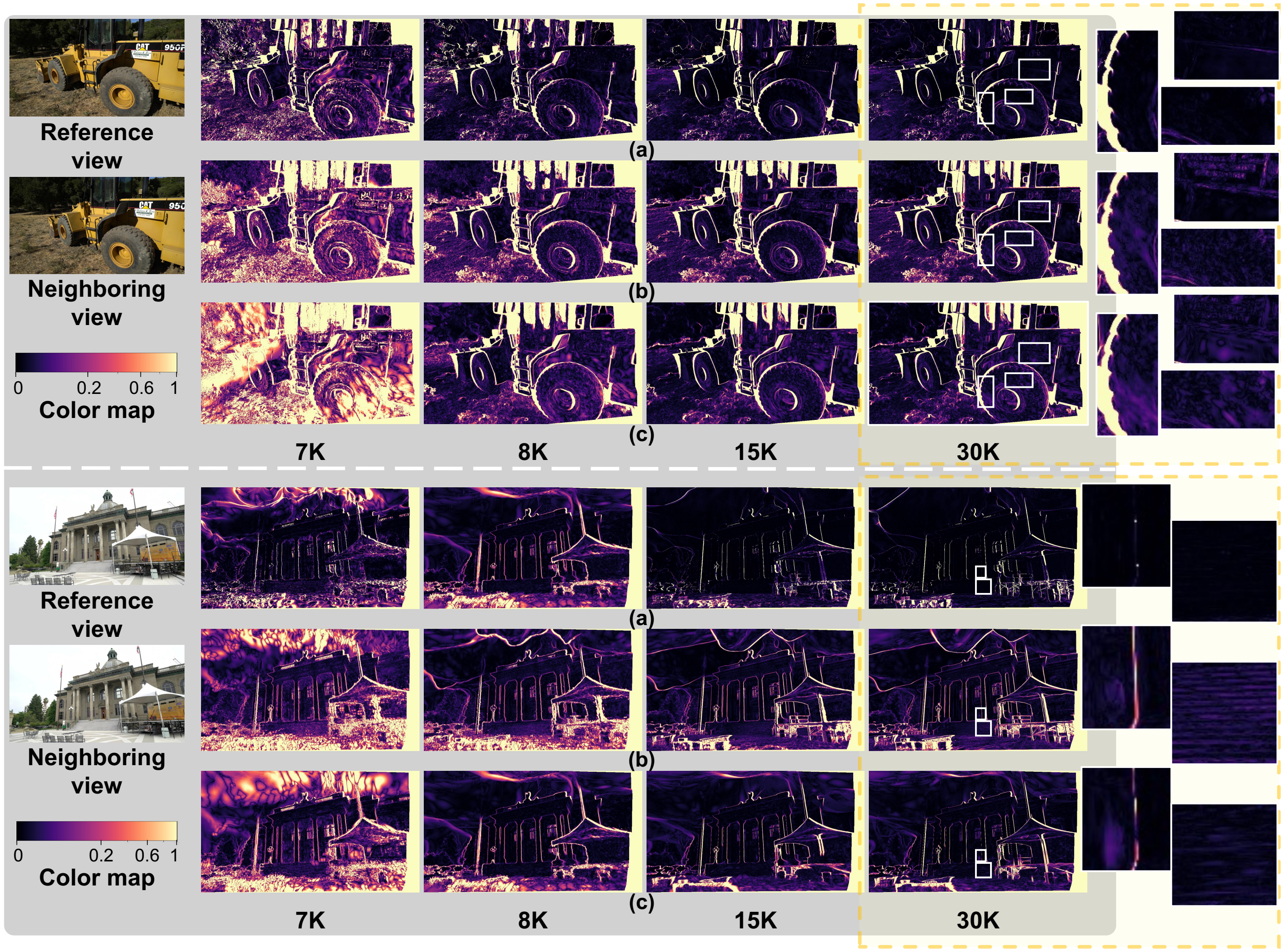

Fig. 13. **Cycle reprojection error per iteration**. We visualize the cycle reprojection error between a reference view and its nearest neighboring view throughout optimization. We show the projection error of (a) our method, (b) PGSR, and (c) RaDe-GS with multi-view regularization. Zoomed-in patches are shown on the right.

Supplementary: Geometry-Grounded Gaussian Splatting

A Implementation of Depth Rendering

In this section, we detail the implementation of the forward and backward passes for depth rendering.

A.1 Forward Pass

We begin with the initial median depth t_{init} obtained from RaDe-GS [Zhang et al. 2024]. We then establish an initial depth interval $[t_{init} - r, t_{init} + r]$ and search for the median depth within this range, setting r to 0.4 during training. To perform the binary search, we need to traverse the Gaussians along the camera ray and record the transmittance at the mid-point, comparing it to the target value of 0.5. However, traversing Gaussians can be time-consuming, so we aim to reduce the number of Gaussian traversals. Specifically, instead of splitting the interval into two segments by a midpoint, we evenly divide it into eight segments using seven segment points and record the transmittance at each segment point. After each traversal, we locate the segment whose endpoint transmittance values fall on opposite sides of 0.5 and use it as the new search interval. Under this setting, a single traversal is equivalent to three binary-search iterations. We repeat this process 5 times, gradually narrowing the interval until the final depth error is within $0.8 \times 8^{-5} = 2.441 \times 10^{-5}$. In the first pass, we also record the transmittance values at both ends of the interval, *i.e.*, $t_{init} - r$, $t_{init} + r$. If both values are above 0.5 or below 0.5, we mask the pixel for geometric regularization.

A.2 Backward Pass

The backpropagation of depth can be calculated as,

$$\frac{\partial L}{\partial t_{med}} \cdot \frac{\partial t_{med}}{\partial \theta}, \tag{14}$$

where L denotes the loss, t_{med} is the median depth computed in the forward pass, and θ represents the parameters of the Gaussians along the camera ray. We further write $\theta = \{\theta_i\}$, where θ_i denotes the parameters of the i-th Gaussian. The first term $\partial L / \partial t_{med}$ is the input of the backward function, and we need to calculate the second term. The formulation of the second term is shown in Equation 13 of the main submission and will be derived in section I. It is composed of $\partial T(t;\theta)/\partial t|_{t=t_{med}}$ and $\partial T(t_{med};\theta)/\partial \theta$. We traverse the Gaussians along the ray twice, computing the two terms in separate passes.

In the first pass, we calculate $\partial T(t;\theta)/\partial t|_{t=t_{med}}$. Equation 10 and Equation 11 of the main submission show that,

$$T(t_{med};\theta) = \prod_i T_i(t_{med};\theta_i) = 0.5. \tag{15}$$

We can obtain:

$$\begin{aligned}
\frac{\partial T(t;\theta)}{\partial t}\Big|_{t=t_{med}} &= \sum_i \sum_{j\neq i} T_j(t_{med};\theta_j) \frac{\partial T_i(t;\theta_i)}{\partial t}\Big|_{t=t_{med}} \\
&= \sum_i \frac{T(t_{med};\theta)}{T_i(t_{med};\theta_i)} \frac{\partial T_i(t;\theta_i)}{\partial t}\Big|_{t=t_{med}} \\
&= \sum_i \frac{0.5}{T_i(t_{med};\theta_i)} \frac{\partial T_i(t;\theta_i)}{\partial t}\Big|_{t=t_{med}}.
\end{aligned} \tag{16}$$

After computing Equation 16, we modify the standard Gaussian Splatting color-accumulation backward pass to additionally compute $\partial T(t_{med};\theta)/\partial \theta_i$ for each Gaussian.

$$\begin{aligned}
\frac{\partial T(t;\theta)}{\partial \theta_i} &= \sum_{j\neq i} T_j(t_{med};\theta_j) \frac{\partial T_i(t_{med};\theta_i)}{\partial \theta_i} \\
&= \frac{T(t_{med};\theta)}{T_i(t_{med};\theta_i)} \frac{\partial T_i(t_{med};\theta_i)}{\partial \theta_i} \\
&= \frac{0.5}{T_i(t_{med};\theta_i)} \frac{\partial T_i(t_{med};\theta_i)}{\partial \theta_i}.
\end{aligned} \tag{17}$$

The $\partial T_i(t;\theta_i)/\partial t|_{t=t_{med}}$ in Equation 16 and $\partial T_i(t_{med};\theta_i)/\partial \theta_i$ in Equation 17 can be easily derived from the closed formed formulation of T_i, *i.e.*, Equation 12 of the main submission. Using Equation 13 of the main submission, we plug Equation 16 and Equation 17 into Equation 14. The gradients are backpropagated to each Gaussian as,

$$\frac{\partial L}{\partial t_{med}} \cdot \frac{\partial t_{med}}{\partial \theta_i} = \frac{\partial L}{\partial t_{med}} \cdot \frac{\partial T(t_{med};\theta)}{\partial \theta_i} / (-\frac{\partial T(t;\theta)}{\partial t}|_{t=t_{med}}). \tag{18}$$

B Gaussian Splatting Preliminaries

Gaussian Splatting represents a scene as a set of 3D Gaussian primitives. A single primitive is parameterized by a center $\mathbf{x}_c \in \mathbb{R}^3$, an opacity $o \in [0,1]$, and a symmetric positive definite covariance $\Sigma \in \mathbb{R}^{3\times 3}$,

$$G(\mathbf{x}) = o \exp\Big(- (\mathbf{x} - \mathbf{x}_c)^\top \Sigma^{-1} (\mathbf{x} - \mathbf{x}_c) \Big). \tag{19}$$

From 3D to screen-space Gaussians. To render efficiently, Gaussian Splatting rasterizes each 3D Gaussian as an elliptical 2D Gaussian on the image plane. Let the camera extrinsics map world coordinates to camera coordinates, and denote the world-to-camera rotation as $\mathbf{W}$. The covariance in the camera frame is

$$\Sigma_{cam} = \mathbf{W} \Sigma \mathbf{W}^\top. \tag{20}$$

Let $\pi(\cdot)$ be the perspective projection and $\mathbf{u}_c = \pi(\mathbf{x}_{c,cam})$ be the projected center on the image plane. Using a local affine approximation of π around $\mathbf{x}_{c,cam}$, the screen-space covariance is obtained via multiplying with a Jacobian matrix:

$$\Sigma'_{2D} = \mathbf{J}\, \Sigma_{cam}\, \mathbf{J}^\top = \mathbf{J}\mathbf{W}\Sigma\mathbf{W}^\top\mathbf{J}^\top, \tag{21}$$

where $\mathbf{J} \in \mathbb{R}^{2\times 3}$ is the Jacobian of the perspective projection evaluated at $\mathbf{x}_{c,cam}$.

Per-pixel alpha. Given a pixel location $\mathbf{u} \in \mathbb{R}^2$, the Gaussian contributes an opacity (alpha) determined by its screen-space ellipse:

$$\alpha(\mathbf{u}) = o\, e^{-(\mathbf{u}-\mathbf{u}_c)^\top \Sigma_{2D}'^{-1} (\mathbf{u}-\mathbf{u}_c)}, \tag{22}$$

where $\mathbf{u}_c$ is the projected Gaussian center. In practice, each Gaussian is evaluated only on pixels within a finite screen-space support to keep rasterization fast.

Alpha compositing. For each pixel, let N denote the set of Gaussians whose projected support overlaps that pixel. These Gaussians are depth-sorted and accumulated using standard alpha blending. Denoting the per-Gaussian color as $\mathbf{c}_i$ and $\alpha_i = \alpha_i(\mathbf{u})$, the rendered color is

$$\mathrm{C}(\mathbf{u}) = \sum_{i \in N} \mathbf{c}_i\, \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{23}$$

C Loss Functions

During training, we employ three loss terms: the photometric loss from Gaussian Splatting [Kerbl et al. 2023], the normal consistency loss from 2D GS [Huang et al. 2024], and the multi-view regularization from PGSR [Chen et al. 2024]. We describe each term below.

Photometric loss. We follow [Kerbl et al. 2023] and define the photometric loss as a weighted combination of an L_1 term and a D-SSIM term between the rendered image and the ground-truth image:

$$L_c = (1 - \lambda)\, L_1 + \lambda\, L_{SSIM}, \tag{24}$$

where λ is a hyperparameter.

Normal consistency. Photometric supervision alone is insufficient to constrain geometry, so we introduce additional geometric regularization. Specifically, we adopt the normal-consistency loss from 2D GS [Huang et al. 2024], which encourages the Gaussian normals to agree with the surface normal estimated from the rendered depth map. Concretely, we compute a reference normal $\tilde{\mathbf{n}}$ by applying finite differences to the depth map and penalize its angular deviation from each Gaussian normal:

$$\mathcal{L}_n = \sum_i \omega_i \left(1 - \mathbf{n}_i^\top \tilde{\mathbf{n}}\right), \tag{25}$$

where $\mathbf{n}_i$ is the normal of the i-th Gaussian along the ray, and $\omega_i = \alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j)$ is its alpha-compositing weight.

Multi-view regularization. We adopt the multi-view regularization of PGSR [Chen et al. 2024] to our method, which combines a photometric term with an explicit geometric cycle-consistency term. Concretely, for each reference pixel $\mathbf{u}_r$, we approximate the local surface as a plane using the rendered depth and normal, and use the induced plane homography to relate the reference view to a neighboring view:

$$\mathbf{H}_{rn} = \mathbf{K}_n \left(\mathbf{R}_{rn} + \frac{\mathbf{T}_{rn} \mathbf{n}_r^\top}{\mathbf{p}_r^\top \mathbf{n}_r} \right) \mathbf{K}_r^{-1}, \tag{26}$$

where $\mathbf{K}_r$ and $\mathbf{K}_n$ are the intrinsics, $(\mathbf{R}_{rn}, \mathbf{T}_{rn})$ is the relative pose from the reference to the neighboring camera, $\mathbf{n}_r$ is the rendered normal at $\mathbf{u}_r$, and $\mathbf{p}_r$ is the 3D point in the reference camera frame obtained from the rendered depth along the ray through $\mathbf{u}_r$.

Using $\mathbf{H}_{rn}$, we warp the neighboring image into the reference frame and enforce patch-level photometric consistency via normalized cross-correlation (NCC):

$$L_{pc} = \sum_{\mathbf{u}_r} w(\mathbf{u}_r) \Big(1 - \mathrm{NCC}\big(I_r(\mathbf{u}_r),\, I_n(\mathbf{H}_{rn}\mathbf{u}_r) \big) \Big), \tag{27}$$

where I_r and I_n denote the reference and neighboring images. To handle occlusions and unreliable correspondences, PGSR defines a confidence weight from a forward–backward reprojection cycle. Specifically, letting $\mathbf{H}_{nr}$ denote the homography that maps from the neighboring view back to the reference view, the cycle reprojection error is

$$\phi(\mathbf{u}_r) = \|\mathbf{u}_r - \mathbf{H}_{nr}\mathbf{H}_{rn}\mathbf{u}_r\|_2\,, \tag{28}$$

which is the same reprojection error introduced in the main submission. The confidence is then

$$w(\mathbf{u}_r) = \begin{cases} \exp\big(-\phi(\mathbf{u}_r)\big), & \phi(\mathbf{u}_r) < 1, \\ 0, & \phi(\mathbf{u}_r) \geq 1, \end{cases} \tag{29}$$

thus discarding pixels with large cycle error.

In addition to the photometric term, PGSR directly penalizes the cycle reprojection error to encourage view-consistent geometry:

$$L_{gc} = \sum_{\mathbf{u}_r} w(\mathbf{u}_r)\, \phi(\mathbf{u}_r). \tag{30}$$

The overall multi-view regularization is

$$L_{mv} = w_{pc} L_{pc} + w_{gc} L_{gc}, \tag{31}$$

where w_{pc} and w_{gc} control the relative strength of photometric and geometric consistency.

Our final training loss $\mathcal{L}$ is,

$$\mathcal{L} = \mathcal{L}_c + w_n \mathcal{L}_n + L_{mv}. \tag{32}$$

We use $w_n = 0.05$, $w_{pc} = 0.6$, $w_{gc} = 0.02$, and set $\lambda = 0.2$ in Equation 24.

D Comparison on Novel View Synthesis

We further compare novel view synthesis results across methods. Table 4 shows the quantitative results on the Mip-NeRF 360 dataset. Our method achieves competitive performance compared with existing surface reconstruction baselines, while RayGaussX produces the overall best rendering quality. To isolate the effect of specular modeling, we augment our model with the spherical Gaussian mixture used in RayGaussX, denoted as Ours (SG). Notably, we use these Gaussian lobes only in this experiment; all other experiments use our default model without Gaussian lobes. Figure 14 shows the qualitative results of our method.

E Limitation and Future Work

To maintain the efficiency of Gaussian Splatting, this paper only considers the volumetric effects when rendering depth. Future works can combine the stochastic theory with existing volume rendering methods [Blanc et al. 2025b; Kheradmand

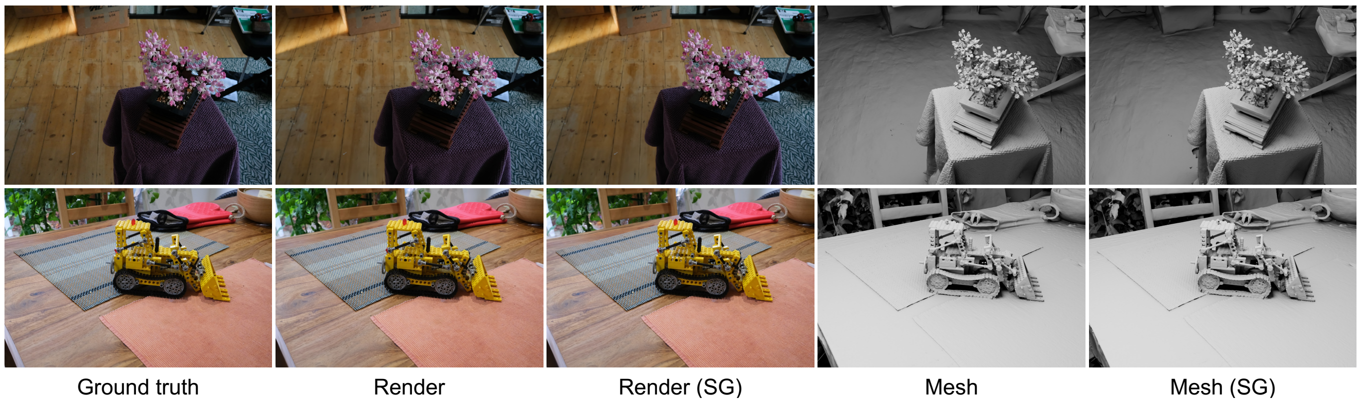

Fig. 14. **Qualitative results on the Mip-NeRF 360 dataset [Barron et al. 2022].** We visualize novel view synthesis and shape reconstruction results for our method, and for our method augmented with the spherical Gaussian appearance model of RayGauss [Blanc et al. 2025a]. Incorporating spherical Gaussians improves rendering quality in specular regions.

Table 4. **Quantitative results on Mip-NeRF 360 dataset.** The best scores are highlighted with colors.

		Outdoor Scene			Indoor Scene		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NVS	NeRF	21.46	0.458	0.515	26.84	0.790	0.370
	Deep Blending	21.54	0.524	0.364	26.40	0.844	0.261
	Instant NGP	22.90	0.566	0.371	29.15	0.880	0.216
	Mip-NeRF 360	24.47	0.691	0.283	31.72	0.917	0.180
	3DGS	24.67	0.728	0.240	30.96	0.924	0.187
	SVRaster	24.68	0.738	0.206	30.65	0.927	0.161
	RayGaussX	25.24	0.761	0.167	32.43	0.943	0.146
Surface Recon.	BakedSDF	22.47	0.585	0.349	27.06	0.836	0.258
	SuGaR	22.93	0.629	0.356	29.43	0.906	0.225
	2DGS	24.34	0.717	0.246	30.40	0.916	0.195
	GOF	24.82	0.750	0.202	30.79	0.924	0.184
	VCR-GauS	24.31	0.707	0.280	30.53	0.921	0.184
	PGSR	24.76	0.752	0.203	30.36	0.934	0.147
	GeoSVR	24.83	0.738	0.218	30.46	0.921	0.172
	Ours (SG)	24.97	0.754	0.200	32.18	0.938	0.150
	Ours	25.09	0.760	0.196	31.02	0.934	0.154

et al. 2025] to fully utilize the volumetric nature of stochastic when rendering color and normal maps. We believe this will lead to further improvement in shape reconstruction.

To compute the median depth, our binary search is initialized with a fixed depth interval. This interval must be sufficiently wide, which increases the number of search steps and slows training. For large-scale scenes, the true median depth may even fall outside this preset range, hindering effective optimization. We leave it to future work to develop adaptive bracketing strategies that reliably locate the median and tighten the initial interval, further accelerating depth rendering.

In Marching Tetrahedra, while the vertex placement is Gaussian-aware, the subsequent 3D Delaunay triangulation step remains general-purpose. In practice, reconstructing thin or near-planar structures often requires dense vertices. Designing Gaussian-specific tetrahedralization and refinement strategies is a meaningful direction for future work.

As we have bridged the gap between Gaussian and NeRF reconstruction methods, future work could consider adopting geometric regularization from NeRF-based methods, *e.g.*, Neuralangelo [Li et al. 2023], for Gaussian Splatting-based methods to enhance shape quality.

F Proof of the Equation 6 in the main submission

As shown in Figure 15, Gaussian Splatting [Kerbl et al. 2023] applies a local affine approximation when projecting a Gaussian primitive. As a result, the light rays from the camera center are parallel to each other. The 3D Gaussian values on each ray form a 1D Gaussian function, which is denoted as $G_{uv}(t)$. It is

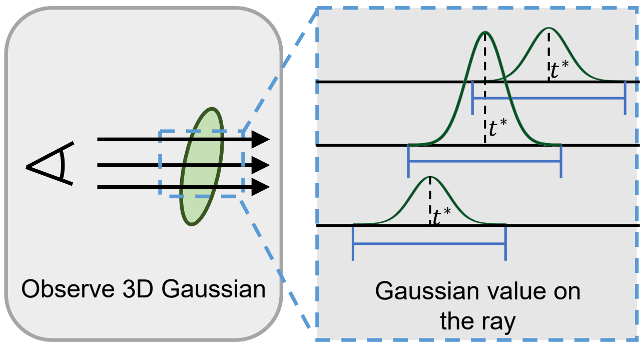

Fig. 15. Gaussian Splatting uses local affine approximation so that rays are parallel to each other. Gaussian functions on the rays have the same variance but different maximum values.

interesting to notice that these 1D Gaussians on different light rays have the same variance but different maximum values.

Gaussian Splatting performs volume rendering on a single primitive as:

$$G'_{2D}(u,v) = \int_{-\infty}^{+\infty} G_{uv}(t)\,dt, \tag{33}$$

which is proportional to the maximum value of $G_{uv}(t)$ because of the identical spread, shown in Figure 15.

Furthermore, as shown in Equation 22, Gaussian Splatting normalizes the maximum value of the 2D Gaussian to opacity o, which aligns the maximum value between the 2D and the 3D Gaussians, ensuring that **the 2D Gaussian value matches the maximum value of the corresponding 1D Gaussian along the ray**. This conclusion facilitates the derivation of the stochastic Gaussian solid. While our result is derived from local affine projection, our method can be easily extended to perspective projection.

G Proof of "Gaussians as Stochastic Solids"

Miller et al. [2024] propose a method to render stochastic opaque solids by converting the vacancy to the attenuation coefficient:

$$\sigma(\mathbf{x},\omega) = |\omega \cdot \nabla log(\mathrm{v}(\mathbf{x}))| = \frac{|\omega \cdot \nabla \mathrm{v}(\mathbf{x})|}{\mathrm{v}(\mathbf{x})}, \tag{34}$$

where $\mathbf{v}$ represents the vacancy of the stochastic solid.

In this section, we will prove that given a Gaussian primitive $G(\mathbf{x})$ rendered by Gaussian Splatting, we can find a solid that generates the same rendering results by the stochastic theory. The vacancy of the solid should satisfy:

$$\mathrm{v}(\mathbf{x}) = \sqrt{1 - G(\mathbf{x})}. \tag{35}$$

We will prove that under the following constraints, the stochastic solid can be uniquely determined:

- When $G(\mathbf{x}_1) \geq G(\mathbf{x}_2)$, it follows that $\mathrm{o}(\mathbf{x}_1) \geq \mathrm{o}(\mathbf{x}_2)$, indicating that positions closer to the Gaussian center have higher occupancy;
- The occupancy of the solid approaches 0 when $\mathbf{x}$ is far from Gaussian center;
- The occupancy $\mathrm{o}(\mathbf{x})$ is differentiable with respect to $\mathbf{x}$.

Proof: Assume a line l parameterized by t passes through $G(\mathbf{x})$. We get the value of $G(\mathbf{x})$ on the line forming a 1D Gaussian function $G(t)$, where t goes from $-\infty$ to $+\infty$ and reaches the maximum of the 1D Gaussian at t^*.

Firstly, we derive the color from volume rendering. According to the first assumption, the vacancy function along this line l has the opposite monotonicity compared to the Gaussian function. We will get the attenuation coefficient from Equation 34:

$$\begin{aligned}\sigma(t) &= |\omega \cdot \nabla log(\mathrm{v}(\mathbf{x}))| \\ &= |\frac{\partial log(\mathrm{v}(\mathbf{x}))}{\partial t}| \\ &= \begin{cases} -\frac{\partial log(\mathrm{v}(\mathbf{x}))}{\partial t}, & t \leq t^* \\ \frac{\partial log(\mathrm{v}(\mathbf{x}))}{\partial t}, & t > t^* \end{cases}\end{aligned} \tag{36}$$

Since a Gaussian kernel has a uniform color, we can simplify the volume rendering:

$$\begin{aligned}\mathbf{C} &= \int_{t=-\infty}^{t=+\infty} T(t)\sigma(\mathbf{x}(t),\omega)\mathbf{c}\, dt \\ &= \mathbf{c}\int_{t=-\infty}^{t=+\infty} T(t)\sigma(\mathbf{x}(t),\omega)\, dt \\ &= \mathbf{c}\int_{t=-\infty}^{t=+\infty} -dT(t) \\ &= \mathbf{c}T(t)\Big|_{t=+\infty}^{t=-\infty} = \mathbf{c}(1 - T(+\infty))\end{aligned} \tag{37}$$

We then substitute the Equation 36 into Equation 37 to get the form of color from volume rendering:

$$\begin{aligned}T(\infty) &= T(-\infty, t^*) \times T(t^*, +\infty) \\ &= e^{-\int_{-\infty}^{t^*}\sigma(\mathbf{x}(s),\omega)} \times e^{-\int_{t^*}^{+\infty}\sigma(\mathbf{x}(s),\omega)} \\ &= e^{-(-log(\mathrm{v}(t))\big|_{-\infty}^{t^*})} \times e^{-(log(\mathrm{v}(t))\big|_{t^*}^{+\infty})} \\ &= \frac{\mathrm{v}(t^*)}{\mathrm{v}(-\infty)} \times \frac{\mathrm{v}(t^*)}{\mathrm{v}(+\infty)} \\ &= \mathrm{v}(t^*)^2 \\ \mathbf{C} =& \mathbf{c}(1 - T(+\infty)) = \mathbf{c}(1 - \mathrm{v}(t^*)^2),\end{aligned} \tag{38}$$

where we use the second assumption that $\mathrm{v}(\infty) = 1 - \mathrm{o}(\infty) = 1$.

Secondly, with the color derived from Gaussian Splatting and volume rendering, we can find the relationship between $\mathrm{v}(t^*)$ and $G(t^*)$:

$$\mathbf{c}G(t^*) = \mathbf{c}(1 - \mathrm{v}(t^*)^2) \Rightarrow \mathrm{v}(t^*) = \sqrt{1 - G(t^*)} \tag{39}$$

Finally, we will generalize Equation 39 from maximum points to all 3D points. Different lines l have different maximum points, and Equation 39 should hold for the maximum point on any line. Given any $\mathbf{x} \in \mathbf{R}^3$, we can always find the direction $\omega \in \mathrm{S}^2$ satisfying $\omega \cdot \nabla G(\mathbf{x}) = \frac{\partial G(\mathbf{x})}{\partial \omega} = 0$, indicating that $\mathbf{x}$ is the maximum point along ray $l : \mathbf{x} + t\omega$. Therefore, the equation should hold for any position $\mathbf{x}$, which reaches the unique solution of vacancy in Equation 35.

H Derivation of Equation 12 of the main submission

In this section, we will derive the closed form $T_i(t)$ in Equation 12 of the main submission, which is also the negative integral of the free-flight distribution $-\int p(t)\, dt$. For brief notation, we use T to denote the transmittance of a single Gaussian.

We start from $t_n = -\infty$. Similar to Equation 38, when $t < t^*$,

$$T(-\infty, t) = e^{-\int_{-\infty}^{t^*}\sigma(\mathbf{x}(s),\omega)} = \mathrm{v}(t). \tag{40}$$

When $t > t^*$,

$$\begin{aligned}T(-\infty, t) &= T(-\infty, t^*) \times e^{-\int_{-\infty}^{t^*}\sigma(\mathbf{x}(s),\omega)} \\ &= \mathrm{v}(t^*) \times \frac{\mathrm{v}(t^*)}{\mathrm{v}(t)} \\ &= \frac{\mathrm{v}(t^*)^2}{\mathrm{v}(t)}.\end{aligned} \tag{41}$$

In most cases, the Gaussian primitive is far from the camera, so we can simply use $T(-\infty, t)$ as $T(t)$.

I Derivation of Equation 13 of the main submission

In this section, we will derive the gradient of the depth t_{med} with respect to the parameters of all the Gaussians along the ray. Since $T(t_{med})$ is a constant value of 0.5, its differential is 0.

$$T(t_{med}; \theta) \equiv 0.5, \tag{42}$$

$$dT(t_{med}; \theta) \equiv 0, \tag{43}$$

where θ represents the parameters of Gaussians along the ray. We then expand the dT and plug t_{med} into it to derive the gradient:

$$dT(t; \theta) = \frac{\partial T}{\partial t} dt + \frac{\partial T}{\partial \theta} \cdot d\theta \tag{44}$$

$$0 = \frac{\partial T}{\partial t} dt_{med} + \frac{\partial T}{\partial \theta} \cdot d\theta \tag{45}$$

$$dt_{med} = \left(-\frac{\partial T}{\partial \theta} / \frac{\partial T}{\partial t}\right) \cdot d\theta. \tag{46}$$

So that the gradient of depth is derived as,

$$\frac{\partial t_{med}}{\partial \theta} = -\frac{\partial T(t_{med}; \theta)}{\partial \theta} / \frac{\partial T(t; \theta)}{\partial t}\Big|_{t=t_{med}}, \tag{47}$$