UniGS: Unified Geometry-Aware Gaussian Splatting for Multimodal Rendering

Yusen Xie
Robotics and Autonomous Systems Thrust
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou 511453, China
yxie827@connect.hkust-gz.edu.cn

Zhenmin Huang
Robotics and Autonomous Systems Thrust
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou 511453, China
zhuangdf@connect.ust.hk

Jianhao Jiao
Department of Computer Science
University College London
Gower Street, WC1E 6BT, London, UK and Archimedes/Athena RC, Greece
jiaojh1994@gmail.com

Dimitrios Kanoulas
Department of Computer Science
University College London
Gower Street, WC1E 6BT, London, UK and Archimedes/Athena RC, Greece
d.kanoulas@ucl.ac.uk

Jun Ma*
Robotics and Autonomous Systems Thrust
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou 511453, China
Cheng Kar-Shun Robotics Institute
The Hong Kong University of Science and Technology
Hong Kong SAR, China
jun.ma@ust.hk

Abstract

In this paper, we propose UniGS, a unified map representation and differentiable framework for high-fidelity multimodal 3D reconstruction based on 3D Gaussian Splatting. Our framework integrates a CUDA-accelerated rasterization pipeline capable of rendering photo-realistic RGB images, geometrically accurate depth maps, consistent surface normals, and semantic logits simultaneously. We redesign the rasterization to render depth via differentiable ray-ellipsoid intersection rather than using Gaussian centers, enabling effective optimization of rotation and scale attribute through closed-form analytic depth gradients. Furthermore, we derive the

*Corresponding Author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

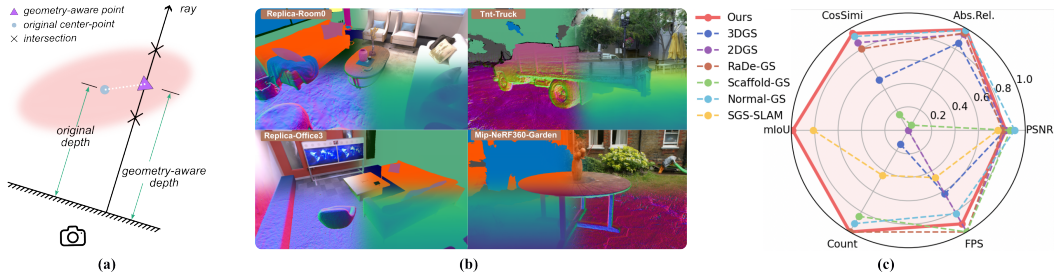


Figure 1: We propose a unified 3D Gaussian Splatting framework that jointly predicts RGB, depth, normal, and semantic map through a single forward rasterization process. (a) Our geometry-aware rendering method explicitly incorporates rotation and scaling attribute into both the rasterization and gradient propagation processes, where we derive an analytical solution for gradient propagation to significantly accelerate optimization. (b) We present multimodal rendering results across diverse scenes, showcasing RGB images, depth maps, surface normals, and semantic segmentation map in a four-panel visualization layout. (c) A comprehensive comparison with widely-used baselines in *Replica-Room0* dataset, underscoring the full modality coverage and strong performance of our framework.

analytic gradient formulation for surface normal rendering, ensuring geometric consistency among the reconstructed 3D scenes. To improve computational and storage efficiency, we introduce a learnable attribute that enables differentiable pruning of Gaussians with minimal contribution during training. Quantitative and qualitative experiments demonstrate the state-of-the-art reconstruction accuracy across several datasets, validating the efficacy and robust optimization stability of our geometry-aware paradigm.

1 Introduction

High-quality novel view synthesis (NVS) and fast multimodal 3D reconstruction remains a challenging problem in computer graphics and attracts widespread attention in fields such as autonomous driving [66, 19, 18, 80] and robotics [84, 39]. In recent years, neural radiance fields (NeRF) [55, 2, 1, 42] and 3D Gaussian Splatting (3DGS) [26, 25] have made big breakthroughs in this field. NeRF uses coordinate-based networks to store 3D scene in multi-layer perceptrons (MLPs) via ray sampling, bringing in photo-realistic reconstruction. 3DGS, another approach, uses explicit 3D Gaussian primitives and modern GPU-accelerated splatting, and this design has been shown to enable high-quality real-time NVS by many 3DGS-based frameworks [40, 29, 12, 45, 17, 65, 49, 66, 37, 38, 75]. However, the two types of existing methods still have many issues, which mainly focus on the following aspects. NeRF requires long training times, lacks easy scene editability, and its implicit 3D information storage method renders geometric reconstruction less intuitive [81, 42]. Additionally, it performs poorly when applied to large-scale scenes [50, 53]. 3DGS and its related frameworks have more challenges: 1) Poor geometric structure and consistency: conventional image-centric 3DGS often ignores objects' geometric structure and semantic info. It only uses Gaussian centers to represent depth, leaving out Gaussian rotation and scale attributes. This stops gradient propagation for geometric constraints (e.g. depth and normal), leading to inconsistent surface normals [40, 25, 33, 61] and misalignment between Gaussians primitives and reconstructed surfaces. 2) Inefficient densification and pruning: some 3DGS methods create too many redundant Gaussians primitives to fit RGB images well [26, 33, 25, 66]. Though this improves metrics, it slows down rendering and uses more GPU memory, losing 3DGS's real-time edge. 3) Poor unified multimodal framework: some 3DGS-related methods use separate frameworks for multimodal rendering [32, 15], which causes repeated calculations and inconsistent 3D scene representations, failing to unify multimodal modeling. Existing studies have tried single or multiple scene modalities [40, 61, 33, 10, 76], but there is still no unified framework that combines all these modalities while keeping storage efficient and rendering fast. This gap serves as the key motivation for our approach.

Against this backdrop, we propose **UniGS**, a *unified* geometry-aware 3D Gaussian Splatting map representation and framework that establishes a new benchmark for consistent 3D scene reconstruction. During rendering, each pixel is shaded via backward ray tracing to determine its intersection with

individual Gaussian ellipsoids (Fig. 1(a)). This formulation explicitly integrates Gaussian rotation and scale attributes into the rendering pipeline. By leveraging analytical gradients derived from depth and normal supervision, it enables efficient and stable gradient-based optimization of these geometric attributes. Furthermore, we introduce a learnable attribute to continuously prune Gaussians. Experimental results demonstrate that this attribute significantly improves computational and storage efficiency without sacrificing rendering or geometric accuracy. Crucially, our framework achieves high-fidelity rendering of *RGB, depth, normals, and semantic logits* through a single differentiable pipeline simultaneously (Fig. 1(b)). The loss from each modality can be back-propagated via analytical gradients to the shared 3D Gaussian representation, thereby maintaining consistency across all modalities in the reconstructed environment (Fig. 1(c)). Particularly, our method achieves a 66.4% improvement in depth estimation accuracy while reducing the Gaussian primitive count by 17.2%. In this paper, our contributions are summarized as follows:

- We develop a novel differentiable depth rasterization method based on ray-ellipsoid intersection, which enables fast and stable optimization of Gaussian rotation and scale attributes via analytic gradients propagation. This approach ensures that Gaussian primitives conform closely to underlying surface geometries, significantly improving geometric consistency.
- We propose a trainable gradient factor in map representation for dynamically pruning insignificant Gaussians in a differentiable manner, which enhances rendering speed and storage efficiency.
- We propose a unified map representation and differentiable rasterization framework that integrates multimodal data. This integration not only enables mutual enhancement across modalities but also improves rendering efficiency and strengthens the consistency of the reconstructed environment.
- Through comprehensive experimental evaluation, we demonstrate that our approach achieves state-of-the-art reconstruction quality across all modalities while consistently maintaining real-time performance.

2 Related Works

Photo-Realistic Rendering and Densification. In the domain of photo-realistic reconstruction and rendering, NeRF and 3DGS represent implicit and explicit techniques, respectively. NeRF [42, 1] and its variants pioneer the use of implicit neural representations for NVS, achieving high-fidelity rendering by optimizing a continuous volumetric scene network using multi-view images. However, this approach typically requires extremely long training times. Subsequent works incorporate geometric prior structures [73, 2, 54] or efficient encoding [44] to accelerate the process. Despite these improvements, NeRF still faces inherent limitations, such as the difficulty in editing implicitly represented scenes. In contrast, 3DGS [25, 67] employs explicit Gaussian primitives as scene representations. Through compression [40, 29, 12, 45, 17] and efficient geometric structures [26, 4], several studies extend 3DGS to large-scale outdoor [65, 49, 66, 34] and urban [37, 38] environments while maintaining high-quality reconstruction. But the aforementioned research primarily focuses on RGB-based rendering and seldom incorporates explicit geometric information of the environment. Moreover, impressive efforts have been made to enhance optimization and storage efficiency. SteepestGS [57] uncovers fundamental principles of density control. DashGaussian [5] significantly accelerates 3DGS optimization. GeoTexDensifier [20] utilizes geometric information for Gaussian densification, and Pixel-GS [79] introduces pixel-level regularization for density control. These methods are regarded as post-processing techniques and lack integration within a unified framework.

Geometry-Aware Rendering. Geometry-aware approaches enhance reconstruction accuracy by incorporating surfel [17, 61, 22, 14, 9], normals [61, 50], and relighting [13, 64, 71] capabilities, enabling more physically realistic renderings and interactions. Some studies [58, 46, 68, 50] attempt to achieve accurate geometry within the NeRF framework, and while significant progress has been made, these methods still suffer from inherent limitations that are difficult to overcome with implicit representations. 3DGS [25, 40] does not inherently excel in geometric reconstruction either, as its Gaussian primitives are independent of each other, making precise geometry recovery an open challenge. Several works unitize 3DGS or 2DGS to recover depth [25, 40, 56, 61, 75], surfel [17, 61, 22] and meshes [17, 14, 9, 76], including normal estimation [61, 75] and relighting [13, 64, 71] techniques, which can effectively reconstruct smooth object surfaces. Nevertheless, these

approaches remain largely limited to small-scale objects. When extended to large environments such as outdoor or urban scenes [43, 63], the required computational time remains substantial. Some methods attempt to incorporate SDF-based surfel constraints [74, 68, 59, 35, 69], inverse rendering [51, 70], or restrictions [33, 24] on Gaussian properties. However, these approaches lack rigorous analytical derivation [75] for geometric modeling, employ relatively limited quantitative metrics, and demonstrate constrained generalization across diverse scenarios. Consequently, the improvements achieve thus far remain marginal in terms of geometric fidelity.

Semantic Rendering Frameworks. Several works attempt to integrate semantic information into both NeRF [81, 6] and 3DGS [15, 47], including techniques like Open-Vocabulary Segmentation [77, 28, 10] and Language-Driven Semantic Segmentation [48, 36, 41, 31], with promising results. Embedding semantic information facilitates downstream tasks such as scene understanding [21, 82], navigation [30, 23], and scene editing [72, 11]. For instance, SGS-SLAM [32] directly uses semantic ground truth as supervision signal and rendering output in 3DGS, though it fails to formulate a unified 3DGS map representation. Feature-GS [83] extracts semantic features from the environment using 3DGS, and SEGS-SLAM [62] enhances structural awareness with appearance embeddings. Nevertheless, such approaches exhibit limitations in reconstructing accurate environmental geometry and fail to achieve consistent reconstruction of scene geometry and semantics.

3 Methodology

The methodology section is organized as follows. Firstly, we define our map representation and objective of this work (Sec. 3.1). Then, the subsequent sections detail the key components of our framework: geometry-aware depth rasterization and normal estimation (Sec. 3.2); the novel gradient factor and differentiable pruning strategy (Sec. 3.3); and finally, the overall multimodal rendering pipeline, loss functions, and implementation details (Sec. 3.4). The brief overview of our framework is illustrated in Fig. 2.

3.1 Unified Map Representation and Objectives

Unified Map Representation. In our framework, each Gaussian $\mathcal{G}$ is defined by position $\boldsymbol{\mu} \in \mathbb{R}^3$, quaternion $\mathbf{q} \in \mathbb{R}^4$, scale $\mathbf{s} \in \mathbb{R}^3$, opacity $\alpha \in \mathbb{R}$, and Spherical Harmonics (SHs) $\mathbf{h} \in \mathbb{R}^{C_\mathrm{h}}$ (C_h is the pre-defined SHs number per color channel), semantic logits (encode the probability distribution across C_o semantic categories) $\mathbf{o} \in \mathbb{R}^{C_\mathrm{o}}$ and gradient factor $k \in \mathbb{R}$:

$$\mathcal{G} = \{\boldsymbol{\mu},\, \mathbf{q},\, \mathbf{s},\, \alpha,\, \mathbf{h},\, \mathbf{o},\, k\}. \tag{1}$$

Objective. The objective of this framework is to render RGB color image, geometric depth, surface normals, semantic logits, and a gradient factor map (used in differentiable pruning) *simultaneously* within a *unified* rasterization pipeline. These rendered outputs are then compared against their corresponding 2D ground truth to compute respective loss, which guide the optimization of the 3D Gaussian scene through a differentiable backward process. To accelerate convergence and enhance computational efficiency, we derive the relevant *analytical solutions* and implement them using *CUDA-accelerated* gradient propagation. Finally, we can obtain a geometry consistent photo-realistic 3DGS map representation.

3.2 Geometry-Aware Rasterization

Ray-Gaussian Primitive Intersection Formulation. We define an ellipsoid $\mathcal{E}$ associated with Gaussian $\mathcal{G}$ by its center $\boldsymbol{\mu}$, rotation matrix $\mathbf{R}$ (converted from quaternion $\mathbf{q}$) and diagonal scaling matrix $\mathbf{S} = \mathrm{diag}(\mathbf{s}_x, \mathbf{s}_y, \mathbf{s}_z)$, and this ellipsoid consists of all 3D points $\mathbf{x}$ on its surface that satisfy the following equation:

$$\left(\mathbf{S}^{-1}\mathbf{R}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)^\top \left(\mathbf{S}^{-1}\mathbf{R}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) = 1. \tag{2}$$

For each pixel $p = (u, v)$, the corresponding camera ray $\mathbf{r}(t) = \mathbf{t}_{\text{cam}}^{\text{world}} + t\mathbf{d}$ is calculated through a unprojection process:

$$\mathbf{r}(t) = \mathbf{t}_{\text{cam}}^{\text{world}} + \text{normalize}\left(\mathbf{R}_{\text{cam}}^{\text{world}} \begin{bmatrix} \frac{u-c_x}{f_x} & \frac{v-c_y}{f_y} & 1 \end{bmatrix}^\top\right), \tag{3}$$

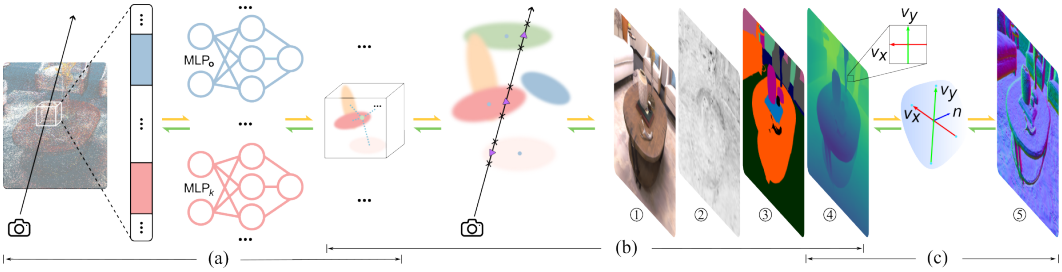


Figure 2: **Overview of our framework**. The yellow arrows (→) indicate the forward rasterization pipeline. (a) First, we predict the semantic logits $\mathbf{o}$, contribution k, and other 3DGS attributes (i.e., position $\boldsymbol{\mu}$, rotation $\mathbf{q}$, scale $\mathbf{s}$, opacity α, SHs $\mathbf{h}$) in the anchor set through MLPs. For each anchor, we predict M raw 3DGS data available for rendering. (b) Then, during the rasterization stage, for each pixel, we use ray tracing to calculate its intersection ($\times$) with the Gaussian ellipsoid, which is closely related to the rotation and scale attributes. We replace the depth of the Gaussian center point (●) with the depth of the midpoint (▲) between the two intersection points. After splatting rasterization, we obtain RGB (①), gradient factor map (②), semantic (③), and depth (④). (c) To obtain the surface normal (⑤), we back-project the depth into three-dimensional space and then use a differential method to obtain the normals in the world coordinate system, ultimately producing the normal rendering map. The final rendered results are compared with their respective ground truths to calculate the loss. The green arrows (←) indicate the direction of gradient propagation through a CUDA-accelerated analytical solution.

where $\mathbf{t}_{\text{cam}}^{\text{world}}$ and $\mathbf{R}_{\text{cam}}^{\text{world}}$ are the translation vector and rotation matrix that transform a point in the camera frame to the world frame, respectively. The focal lengths f_x, f_y and principal point c_x, c_y are camera parameters.

To find the intersection between the ray $\mathbf{r}(t)$ and the ellipsoid $\mathcal{E}$, we transform the ray $\mathbf{r}(t)$ into the *local* (l) *scaled* (s) coordinate system of $\mathcal{E}$:

$$\begin{aligned} \mathbf{v}_s &= \mathbf{S}^{-1}\mathbf{v}_l, \quad \mathbf{v}_l = \mathbf{R}^{-1}(\mathbf{t}_{\text{cam}}^{\text{world}} - \boldsymbol{\mu}), \\ \mathbf{d}_s &= \mathbf{S}^{-1}\mathbf{d}_l, \quad \mathbf{d}_l = \mathbf{R}^{-1}\mathbf{d}. \end{aligned} \tag{4}$$

The intersection problem is then reduced to solving for t in the unit sphere equation:

$$\|\mathbf{v}_s + t\mathbf{d}_s\|^2 = 1. \tag{5}$$

The solution to this equation can be treated as that of a quadratic equation: for a light ray intersecting an ellipsoid, there are two valid solutions (t_1, t_2) for two intersection points, $t_1 = t_2$ for one point, and no valid solutions for no intersection. Details are in the supplementary materials.

Depth Rasterization. We take the *midpoint* $t_{\text{mid}} = (t_1 + t_2)/2$ of the two valid solutions (t_1, t_2) as the intersection results. The replaced depth d of corresponding 3D primitive in camera coordinate can be computed by

$$\mathbf{P}_{\text{cam}} = \mathbf{R}_{\text{world}}^{\text{cam}}(\mathbf{t}_{\text{cam}}^{\text{world}} + t_{\text{mid}}\mathbf{d}), \quad d = \mathbf{P}_{\text{cam},\,z}. \tag{6}$$

The final depth value $\tilde{D}(u, v)$ for pixel $p = (u, v)$ is obtained by alpha-rendering the depths d of all overlapping Gaussians $\mathcal{N}$:

$$\tilde{\mathbf{D}}(u, v) = \sum_{i \in \mathcal{N}} \alpha_i d_i \prod_{j=1}^{i-1}(1 - \alpha_j). \tag{7}$$

The above method allows to directly incorporate the geometric information (rotation $\mathbf{q}$ and scale $\mathbf{s}$ of the Gaussian $\mathcal{G}$) into the rasterization pipeline. This enables us to compute the gradients of the loss function with respect to the rotation $\mathbf{q}$ and scale $\mathbf{s}$, thereby effectively optimizing the Gaussian $\mathcal{G}$. Related gradient backpropagation derivations are provided in the supplementary materials.

Normal Rasterization from Depth. As mentioned previously, obtaining sufficiently smooth and accurate surface normals directly via rasterization is challenging due to the strong directional discontinuities inherent in normal vectors. To circumvent this, we compute normals indirectly from the differentiable depth rasterization output. Our approach begins by unprojecting the depth map into world coordinates using camera parameters, followed by estimating surface normals via a multi-step

finite difference scheme. Detailed procedures for depth back-projection and view-direction alignment are provided in the supplementary material.

Closed-form Geometry Grad Backpropagation. Deriving the gradient flow through this pipeline constitutes the most critical closed-form solution in our framework. Specifically, by leveraging our ray-ellipsoid forward rendering algorithm, the rotation and scale parameters of each Gaussian primitive are explicitly embedded into the differentiable computation graph and ultimately reflected in the final loss. Consequently, analytic gradients can be seamlessly backpropagated to these geometric attributes, driving the primitives to align precisely with true object surfaces and ensuring robust geometric consistency. Moreover, the analytic nature of this closed-form solution yields remarkably fast convergence during the initial training phases with minimal iterations, which not only stabilizes the optimization process but also holds strong potential for downstream applications such as real-time systems and end-to-end training. The complete mathematical derivation of the closed-form backpropagation is provided in the supplementary material.

3.3 Differentiable Pruning

Gradient Factor Rasterization and Backpropagation. We define a new attribute k in (1) for each Gaussian $\mathcal{G}$ to measure its gradient level during the rendering process. Alongside the rendering of RGB image, we can render a 2D gradient factor map $\tilde{\mathbf{K}}$ by alpha-blending:

$$\tilde{\mathbf{K}}(u,v) = \sum_{i\in\mathcal{N}} \alpha_i k_i \prod_{j=1}^{i-1}(1-\alpha_j). \tag{8}$$

Our objective is to drive each Gaussian's k toward 1. Experimental results show this state corresponds to the optimal optimization effect of Gaussians. To enforce this, we minimize the L1-loss between $\tilde{\mathbf{K}}$ and an all-ones matrix $\mathbf{1}$:

$$\mathcal{L}_K = \|\tilde{\mathbf{K}} - \mathbf{1}\|_1. \tag{9}$$

Detailed analysis on why $k = 1$ is optimal and gradient backpropagation are provided in the supplementary materials.

Pruning. After a fixed interval of iterations K_p ($K_p = 3000$ in our experiments), our framework directly eliminates Gaussians with abnormal gradient factor $\|k - \mathbf{1}\|_1 > \mathcal{T}_k$, effectively pruning the set and reducing the number of Gaussians to enhance overall efficiency. After pruning operation, the k value of all Gaussians will be reset to 0.9 to maintain the sustainability of updates. Please refer to our code for details.

3.4 Multimodal Rendering Framework

Architecture Overview. Our system is built upon a modified Scaffold-GS [40] architecture. For each anchor point, we maintain a foundational 3D Gaussian representation (1). During every training iteration, multiple MLPs are used to predict attribute offsets for each anchor.

Attribute Prediction. For semantic logits $\mathbf{o} \in \mathbb{R}^{C_\mathbf{o}}$ and gradient factor $k \in \mathbb{R}$ of a Gaussian $\mathcal{G}$, we replicate (denoted as **Repeat**) M times and modulate by the output of an MLP branch to obtain Gaussian features for rasterization:

$$\begin{bmatrix} \mathbf{o}_{\text{vis}} \\ k_{\text{vis}} \\ \dots \end{bmatrix} = \textbf{Repeat}\left(\begin{bmatrix} \mathbf{o}_{\text{ah}} \\ k_{\text{ah}} \\ \dots \end{bmatrix}, M \right) \otimes \sigma \begin{bmatrix} \textbf{MLP}_\mathbf{o}(\mathbf{f}_{\text{ah}}) \\ \textbf{MLP}_k(\mathbf{f}_{\text{ah}}) \\ \dots \end{bmatrix}, \tag{10}$$

where $\mathbf{o}_{\text{ah}} \in \mathbb{R}^{C_\mathbf{o}}$ is the base semantic logits of the anchor, $k_{\text{ah}} \in \mathbb{R}$ is the base gradient factor of the anchor. M is the replication factor. $\mathbf{f}_{\text{ah}}$ represents the input feature for the anchor. $\text{MLP}_\text{o} : \mathbb{R}^{Fdim} \to \mathbb{R}^{C_\mathbf{o}}$ denotes the MLP branch for semantics. $\text{MLP}_k : \mathbb{R}^{Fdim} \to \mathbb{R}$ denotes the MLP branch for gradient factor. σ is the sigmoid activation function. $\otimes$ denotes element-wise multiplication. $Fdim$ is the feature dimension. The resulting modulated attributes are then combined with other predicted properties (i.e., position $\boldsymbol{\mu}$, rotation $\mathbf{q}$, scale $\mathbf{s}$, opacity α, SHs $\mathbf{h}$, denoted as ...) to form the final 3D Gaussians ready for projection and rendering.

Loss Function. For the RGB image loss, we employ a combination of L1 loss $\mathcal{L}_{l1}$ and SSIM [60] loss $\mathcal{L}_{\text{ssim}}$. For the normal loss $\mathcal{L}_{\text{normal}}$, we use cosine similarity, which is calculated as

$$\mathcal{L}_{normal} = 1 - \text{Cosine Similarity}(\tilde{\mathbf{N}}, \tilde{\mathbf{N}}_{\text{gt}}). \tag{11}$$

Incorporating the gradient factor loss $\mathcal{L}_K$ and semantic loss $\mathcal{L}_{seg}$, we construct the comprehensive loss function as

$$\mathcal{L}_{all} = \lambda_1 \mathcal{L}_{l1} + \lambda_2 \frac{|\mathcal{L}_{l1}|\mathcal{L}_{ssim}}{|\mathcal{L}_{ssim}|} + \lambda_3 \frac{|\mathcal{L}_{l1}|\mathcal{L}_{normal}}{|\mathcal{L}_{normal}|} + \lambda_4 \frac{|\mathcal{L}_{l1}|\mathcal{L}_{depth}}{|\mathcal{L}_{depth}|} + \lambda_5 \frac{|\mathcal{L}_{l1}|\mathcal{L}_{seg}}{|\mathcal{L}_{seg}|} + \lambda_6 \frac{|\mathcal{L}_{l1}|\mathcal{L}_K}{|\mathcal{L}_K|}. \tag{12}$$

where weights are applied as $\lambda_1 = 1, \lambda_2 = \lambda_3 = \lambda_4 = \lambda_4 = \lambda_5 = \lambda_6 = 0.1$.

Implementation Details. Our framework is implemented by CUDA using the LibTorch framework [7], incorporating CUDA code for Gaussian Splatting and trained on a server with a 2.60 GHz Intel(R) Xeon(R) Platinum 8358P CPU, 1T GB RAM, and a NVIDIA RTX 4090 24 GB GPU. The multimodal forward rasterization and backward gradient propagation pipeline are outlined in the supplementary materials.

4 Experiments

In this section, we first introduce the experimental setup in Sec. 4.1, including datasets, baselines, and parameter settings, etc. Then in Sec. 4.2, we evaluate the geometry rendering performance (e.g. depth and normal). The quantitative RGB rendering comparison will be provided in Sec. 4.3, and the qualitative comparison will be provided in supplementary materials accordingly. In Sec. 4.4, we provide the analysis of running time and GPU memory consumption of the framework. Sec. 4.5 shows ablation experiments of the proposed framework.

4.1 Experiment Setup

Datasets. We conduct comparative experiments on widely used datasets, including 7 scenes from Mip-NeRF [1], 2 scenes from Tanks & Temples [27] and 2 scenes from Deep Blending [16]. All of these datasets do not provide precise geometric and semantic ground truth. To address this, we use the results obtained from the depth rendering pipeline in Normal-GS [61] as geometric priors for data preparation. At the same time, we utilize open vocabulary queries to pre-process semantic labels with OpenSeed [77]. For accurate quantitative comparison experiments, we use the Replica [52] simulation dataset, which provides complete multimodal ground truth. We employ the pre-rendered Replica dataset from HiSplat [54], which includes 14 motion sequences across 7 scenes. The initial points in the Replica [52] dataset are obtained through mesh sampling, with a sampling interval of 5 in all sequences.

Baselines and Metrics. We compare our method with the existing SOTA NeRF-based dense framework Mip-NeRF [1], iNGP [44] and 3DGS framework [25], Scaffold-GS [40], GeoGaussian [33], 2DGS [17], Normal-GS [61], SemanticGaussian [15], RaDe-GS [75]. We evaluate the RGB rendering performance using PSNR, SSIM [60], and LPIPS [78]. The evaluation metrics for depth estimation we used are *Abs.Rel*. and RMSE. The evaluation metric we used for normal estimation is cosine similarity (CosSimi), for semantic segmentation, we employed mean mIoU.

Table 1: Quantitative comparison experiments on the Replica dataset show that our method significantly outperforms other baselines.

Dataset	O0S1			O1S2			R1S2			R2S2		
Method | Metrics	Abs.Rel ↓	RMSE↓	CS ↑	Abs.Rel↓	RMSE↓	CS ↑	Abs.Rel↓	RMSE↓	CS ↑	Abs.Rel↓	RMSE ↓	CS↑
GeoGaussian [33]	0.5828	0.8006	0.3046	2.8959	1.1092	0.1861	0.1336	0.3236	0.5700	0.1082	0.2592	0.5998
3DGS [25]	1.2231	0.9183	0.2304	0.8711	0.6238	0.3201	0.2134	0.4092	0.4963	1.4422	1.0972	0.2905
Scaffold-GS [40]	2.2857	1.2102	0.2195	1.4609	0.9413	0.2854	4.0575	1.4786	0.1555	-	-	0.1286
RaDe-GS [75]	0.0136	0.0772	0.8601	0.0300	0.0839	0.8072	0.0848	0.2326	0.8462	0.0674	0.2059	0.7999
2DGS [17]	0.0089	0.0679	0.8775	0.0172	0.0399	0.9299	0.0451	0.1400	0.9112	0.0735	0.1759	0.8480
Normal-GS [61]	0.0074	0.0657	0.8755	0.0154	0.0344	0.9234	0.0243	0.0766	0.9234	0.0192	0.0589	0.8987
Ours	0.0062	0.0310	0.8880	0.0015	0.0073	0.9686	0.0136	0.0467	0.9545	0.0144	0.0465	0.9076

4.2 Geometry Rendering Evaluation

We conduct detailed qualitative and quantitative comparisons of depth and normal estimation against methods [25] [40] [22] [33] [61] [17] [75] on the Replica [52] dataset. The comparison results are shown in Tab. 1 and Fig. 3. Normal-GS [61] and our method optimize the surface normals, and both qualitatively and quantitatively outperform other 3DGS frameworks. Our approach also exhibits

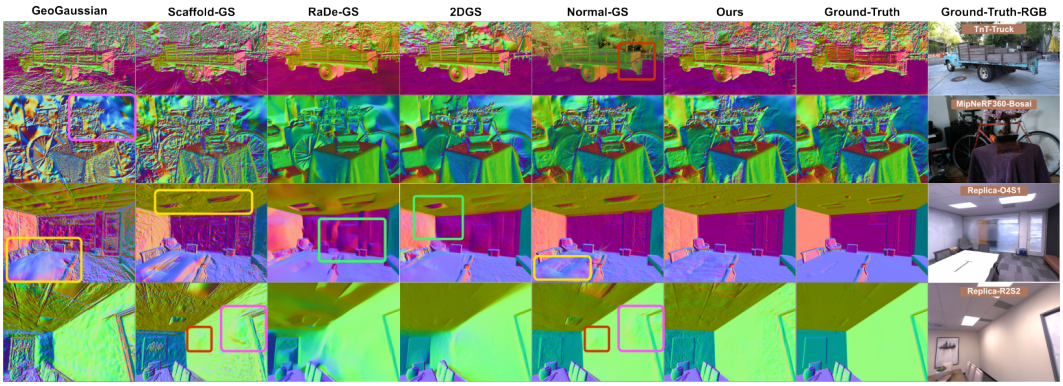


Figure 3: Across various dataset tests, our method demonstrates impressive results in geometric reconstruction. Even in textureless areas and edge regions, it remains capable of recovering fine geometric details.

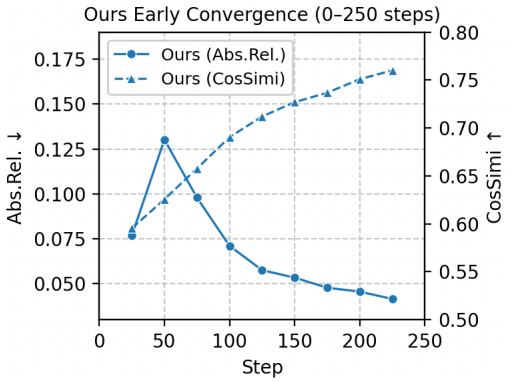


Figure 4: Convergence curves over the first 250 iterations (sampled every 25 steps).

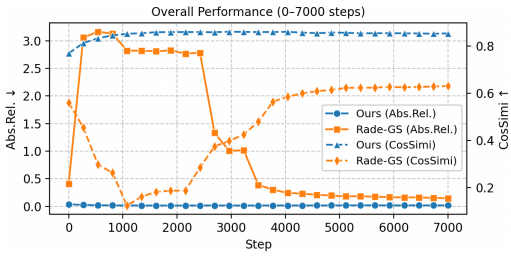


Figure 5: Convergence curves over the first 7000 iterations, sampled every 250 steps.

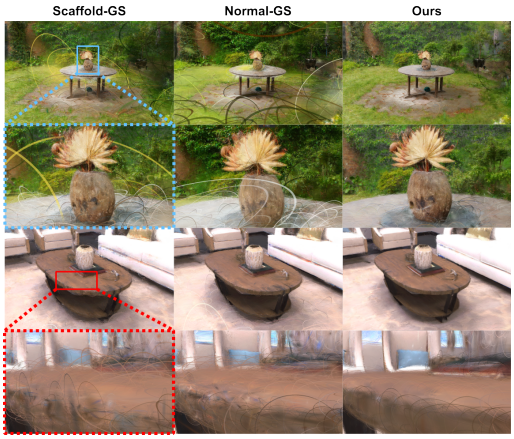


Figure 6: A qualitative demonstration showing that the Gaussian primitives reconstructed by our method are geometrically closer to the real surface.

Table 2: Quantitative geometric comparison between our method and other methods after sampling the reconstructed Gaussian model, compared to the initial points or mesh points.

Dataset	MipNeRF-Garden				Replica-Room0			
Method | Metrics	Radio↓	Mean↓	Std↓	Hau-Dis↓	Radio↓	Mean↓	Std↓	Hau-Dis↓
Scaffold-GS [40]	0.0390	0.0299	0.0383	0.592	0.0476	0.0210	0.0083	0.141
RaDe-GS [75]	0.0479	0.0471	0.0972	0.887	0.0520	0.0403	0.0212	0.364
2DGS [17]	0.0607	0.0984	0.1272	1.091	0.0628	0.0519	0.0244	0.462
Normal-GS [61]	0.0388	0.0303	0.0377	0.936	0.0477	0.0210	0.0082	0.131
Ours	0.0341	0.0272	0.0298	0.405	0.0339	0.0137	0.0081	0.078

superior geometric accuracy in some extreme scenarios, particularly in texture-less areas, surpassing even Normal-GS [61], RaDe-GS [75] and 2DGS [17]. More comparisons about geometry rendering can be found in supplementary materials.

Furthermore, in contrast to Rade-GS [75], as demonstrated in Fig. 4, Fig. 5 and our supplementary video, our closed-form solution rapidly achieves a well-optimized state while preserving stability in the early optimization phase (Fig. 4). In contrast, RaDe-GS remains unstable during this critical period (Fig. 5), highlighting the superior optimization performance of our method, a property particularly advantageous for real-time systems.

We further compare the geometric impact after modifying our depth rendering method. Firstly, we complete the reconstruction process of a 3D Gaussian scene and then identify a detailed area in the reconstructed scene (as shown in Fig. 6). It is evident that our method, compared to others, results in optimized Gaussian primitives that tend to be closer to the actual surface of the object. This is

Table 3: Quantitative comparison of RGB rendering on widely used real and simulated datasets. Some metrics (*) are extracted from the original data in respective papers. The results ranked from best to worst are highlighted as first, second, and third.

Dataset	Mip-NeRF360			Tanks & Temples			Deep Blending			Replica		
Method | Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
iNPG* [44]	26.43	0.725	0.339	21.72	0.723	0.330	23.62	0.797	0.423	-	-	-
Mip-NeRF* [1]	29.23	0.844	0.207	22.22	0.759	0.257	29.40	0.901	0.245	-	-	-
RaySplats* [3]	27.31	0.846	0.237	22.20	0.829	0.202	29.57	0.900	0.320	-	-	-
GeoGaussian [33]	26.26	0.813	0.219	20.00	0.802	0.261	26.07	0.873	0.310	24.55	0.946	0.114
3DGS [25]	28.69	0.870	0.182	23.14	0.841	0.183	29.41	0.903	0.243	36.50	0.940	0.120
Scaffold-GS [40]	28.84	0.848	0.220	23.96	0.853	0.177	30.21	0.906	0.254	37.45	0.951	0.090
RaDe-GS [75]	26.93	0.800	0.242	23.07	0.850	0.195	27.34	0.856	0.307	36.09	0.933	0.112
2DGS [17]	27.34	0.814	0.234	22.89	0.833	0.206	27.96	0.873	0.281	36.47	0.943	0.107
Normal-GS [61]	29.34	0.869	0.194	24.22	0.854	0.174	30.19	0.910	0.252	38.19	0.977	0.043
Ours	30.10	0.887	0.148	23.57	0.860	0.168	29.66	0.899	0.231	36.68	0.962	0.083



Figure 7: Qualitative comparison results on NVS demonstrate that our RGB rendering achieves better performance in certain scenarios. The top-left corner of the image displays the PSNR value computed against the ground truth image.

Table 4: Quantitative comparison results in terms of Gaussian splat count and rendering FPS.

Dataset	Mip-NeRF360		Tanks & Temples		Deep Blending		Replica	
Method | Metrics	Count ↓	FPS ↑	Count ↓	FPS ↑	Count ↓	FPS ↑	Count ↓	FPS ↑
GeoGaussian [33]	126021	440.9	94755	441.1	53457	436.2	100877	423.6
3DGS [25]	504167	128.7	421167	136.9	285394	162.4	543581	137.5
Scaffold-GS [40]	301839	240.1	247081	249.1	182304	324.0	235446	346.2
Normal-GS [61]	318064	180.3	226283	178.7	185243	250.7	204358	180.0
Ours	161081	169.8	119947	163.1	48970	296.7	169273	197.8

because our method integrates rotation and scale attribute into the optimization process, thereby achieving geometric consistency.

Additionally, we perform point sampling [33] of the Gaussian primitives within specific small regions, followed by a Z-score statistical comparison with the initial ground truth and a Hausdorff distance comparison, as shown in Tab. 2. It is evident that our method achieves consistent geometric surface reconstruction. The calculation method using the Z-score statistic is explained in the supplementary materials.

4.3 RGB Rendering Evaluation

To validate the effectiveness of our method in providing valuable information for NVS, we compare its performance on novel view RGB rendering with state-of-the-art frameworks [1] [44] [73] [25] [40] [33] [75]. All methods utilize their provided original code and unified datasets. The comparisons are presented in Tab. 3 and Fig. 7, and more qualitative results of the RGB rendering are shown in the supplementary materials (Fig. 10). It can be observed that our method achieves state-of-the-art results in novel view synthesis for certain scenes. By incorporating geometric constraints into the framework, our algorithm achieves SOTA geometric performance while maintaining RGB reconstruction quality and significantly improving memory and rendering efficiency.

4.4 Runtime and Memory Cost Analysis

In Tab. 4, we present a comparison of the Count of Gaussians and rendering FPS against other methods. Our approach employs an effective differentiable pruning module, which removes a significant number of redundant Gaussians. As a result, the number of Gaussian splats in our reconstructed scenes is the smallest. However, since our method requires rendering multiple modalities simultaneously, the FPS experiences a certain degree of decrease. Nevertheless, this does not severely hinder our rendering speed. In some scenarios, our rendering speed remains superior even compared to Normal-GS [61]. For the averaged training time (TT), please refer to Tab. 6 in the supplementary materials.

4.5 Ablation Study

Geometry-Aware Rasterization. We compare the performance of depth rasterization that considers geometric information with that of depth rasterization using only the center points in terms of image

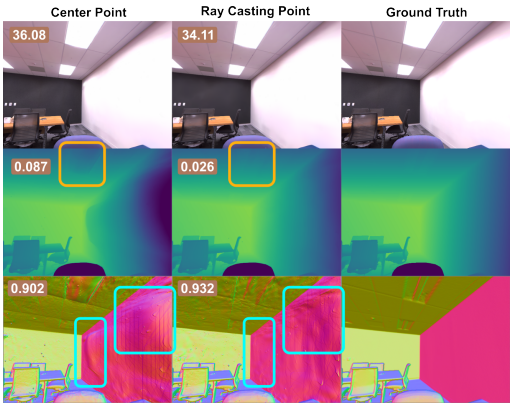


Figure 8: The use of a ray casting-based depth rendering approach significantly enhances the algorithm's robustness in textureless regions. Furthermore, the Gaussian splats provide a tighter fit to the actual object surfaces.

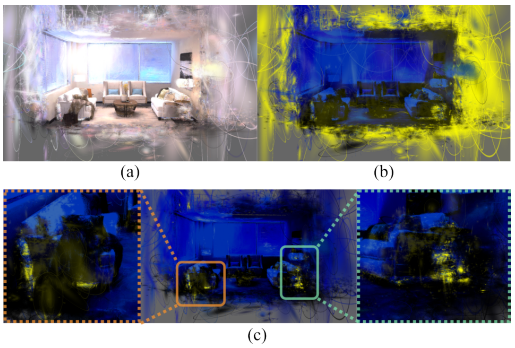


Figure 9: Visualization of the distribution of the gradient factor. To enhance the visualization of the gradient factor, the brightness of all Gaussians is uniformly minimized, resulting in blue regions as rendered by SuperSplat [8]. (a) Scene with original brightness. (b) Areas highlighted in yellow indicate regions where the gradient factor $k > 1.5$. (c) Areas highlighted in yellow indicate regions where $k < 0.5$.

quality. From the qualitative results presented in Fig. 8, it can be observed that incorporating set information forces the Gaussian spheres to align more closely with the actual surface of the objects. More quantitative ablation experiments on each modules can be found in our supplementary materials.

Differentiable Pruning. We conduct continuous optimization of the gradient factor without pruning any Gaussians to examine its distribution, as shown in Fig. 9. It can be observed that regions with anomalous gradient factor, which deviate significantly from the ideal value of 1, are typically areas lacking observational data or corresponding to occluded edges. In these regions, due to insufficient supervisory information from images, the gradients fail to converge to the desired value. Such areas represent environmental redundancy that can be removed to improve computational efficiency.

5 Conclusion

In this paper, we present **UniGS**, a *unified* and differentiable framework for high-fidelity multimodal 3D reconstruction using 3DGS. Our rasterization pipeline effectively renders photo-realistic RGB images, accurate depth maps, consistent surface normals, and semantic labels simultaneously. By implementing differentiable ray-ellipsoid intersection, we optimize rotation and scaling parameters with CUDA-accelerated analytic gradient backpropagation, ensuring geometric consistency in surface. We also introduce a learnable attribute for efficient pruning of minimally contributing Gaussians. Extensive experiments demonstrate state-of-the-art performance in terms of both reconstruction accuracy across all modalities and storage efficiency.

6 Limitations

Real-world environments exhibit complex optical interactions including extensive light reflections and inter reflections. Image captured by a camera is the product of multi-path light propagation and global illumination effects. Our current approach does not explicitly model these radiometric phenomena such as light reflection and composition.

Acknowledgments and Disclosure of Funding

Funding in direct support of this work: none. Author Yusen Xie is a recipient of the NeurIPS 2026 Financial Aid Program (accommodation and registration fee support for conference attendance only). The authors declare no competing interests.

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 *CVPR*, pages 5470–5479, 2022.

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. In *ICCV*, pages 19697–19705, 2023.

[3] Krzysztof Byrski, Marcin Mazur, Jacek Tabor, Tadeusz Dziarmaga, Marcin Kadziolka, Dawid Baran, and Przemysław Spurek. RaySplats: Ray Tracing based Gaussian Splatting. *arXiv preprint arXiv:2501.19196*, 2025.

[4] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-Grid Assisted Context for 3D Gaussian Splatting Compression. In *ECCV*, pages 422–438, 2024.

[5] Youyu Chen, Junjun Jiang, Kui Jiang, Xiao Tang, Zhihao Li, Xianming Liu, and Yinyu Nie. DashGaussian: Optimizing 3D Gaussian Splatting in 200 Seconds. In *CVPR*, pages 11146–11155, 2025.

[6] Zi-Ting Chou, Sheng-Yu Huang, I Liu, Yu-Chiang Frank Wang, et al. GSNeRF: Generalizable Semantic Neural Radiance Fields with Enhanced 3D Scene Understanding. In *CVPR*, pages 20806–20815, 2024.

[7] PyTorch Contributors. Installing C++ Distributions of PyTorch, 2024. URL `https://pytorch.org/cppdocs/installing.html`.

[8] SuperSplat Contributors. SuperSplat - 3D Gaussian Splat Editor, 2025. URL `https://superspl.at/editor`.

[9] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-Quality Surface Reconstruction Using Gaussian Surfels. In *ACM SIGGRAPH*, pages 1–11, 2024.

[10] Yinan Deng, Yufeng Yue, Jianyu Dou, Jingyu Zhao, Jiahui Wang, Yujie Tang, Yi Yang, and Mengyin Fu. OmniMap: A General Mapping Framework Integrating Optics, Geometry, and Semantics. *IEEE TRO*, 2025.

[11] Jiahua Dong and Yu-Xiong Wang. 3DGS-Drag: Dragging Gaussians for Intuitive Point-based 3D Editing. *arXiv preprint arXiv:2601.07963*, 2026.

[12] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. *NeurIPS*, 37:140138–140158, 2024.

[13] Jian Gao, Chun Gu, Youtian Lin, Zhihao Li, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3D Gaussians: Realistic Point Cloud Relighting with BRDF Decomposition and Ray Tracing. In *ECCV*, pages 73–89, 2024.

[14] Antoine Guédon and Vincent Lepetit. SUGAR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. In *CVPR*, pages 5354–5363, 2024.

[15] Jun Guo, Xiaojian Ma, Yue Fan, Huaping Liu, and Qing Li. Semantic Gaussians: Open-Vocabulary Scene Understanding with 3D Gaussian Splatting. *arXiv preprint arXiv:2403.15624*, 2024.

[16] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep Blending for Free-Viewpoint Image-Based Rendering. *ACM SIGGRAPH*, 37(6): 1–15, 2018.

[17] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shengha Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *ACM SIGGRAPH*, pages 1–11, 2024.

[18] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. GaussianFormer: Scene as Gaussians for Vision-Based 3D Semantic Occupancy Prediction. In *ECCV*, pages 376–393, 2024.

[19] Yuanhui Huang, Amonnut Thammatadatrakoon, Wenzhao Zheng, Yunpeng Zhang, Dalong Du, and Jiwen Lu. GaussianFormer-2: Probabilistic Gaussian Superposition for Efficient 3D Occupancy Prediction. In *CVPR*, pages 27477–27486, 2025.

[20] Hanqing Jiang, Xiaojun Xiang, Han Sun, Hongjie Li, Liyang Zhou, Xiaoyu Zhang, and Guofeng Zhang. GeoTexDensifier: Geometry-Texture-Aware Densification for High-Quality Photorealistic 3D Gaussian Splatting. *arXiv preprint arXiv:2412.16809*, 2024.

[21] Minchao Jiang, Shunyu Jia, Jiaming Gu, Xiaoyuan Lu, Guangming Zhu, Anqi Dong, and Liang Zhang. VoteSplat: Hough Voting Gaussian Splatting for 3D Scene Understanding. *arXiv preprint arXiv:2506.22799*, 2025.

[22] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. GaussianShader: 3D Gaussian Splatting with Shading Functions for Reflective Surfaces. In *CVPR*, pages 5322–5332, 2024.

[23] Rui Jin, Yuman Gao, Yingjian Wang, Yuze Wu, Haojian Lu, Chao Xu, and Fei Gao. GS-Planner: A Gaussian-Splatting-Based Planning Framework for Active High-Fidelity Reconstruction. In *IROS*, pages 11202–11209, 2024.

[24] Nikhil Varma Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM. *CVPR*, pages 21357–21366, 2023.

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 42(4):139–1, 2023.

[26] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A Hierarchical 3D Gaussian Representation for Real-time Rendering of Very Large Datasets. *ACM TOG*, 43(4):1–15, 2024.

[27] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM TOG*, 36(4):1–13, 2017.

[28] Zihang Lai. Exploring Simple Open-Vocabulary Semantic Segmentation. In *CVPR*, pages 30221–30230, 2025.

[29] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian Representation for Radiance Field. In *CVPR*, pages 21719–21728, 2024.

[30] Xiaohan Lei, Min Wang, Wengang Zhou, and Houqiang Li. GaussNav: Gaussian Splatting for Visual Navigation. *IEEE TPAMI*, 47(5):4108–4121, 2025.

[31] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-Driven Semantic Segmentation. *arXiv preprint arXiv:2201.03546*, 2022.

[32] Mingrui Li, Shuhong Liu, Heng Zhou, Guohao Zhu, Na Cheng, Tianchen Deng, and Hongyu Wang. SGS-SLAM: Semantic Gaussian Splatting for Neural Dense SLAM. In *ECCV*, pages 163–179, 2024.

[33] Yanyan Li, Chenyu Lyu, Yan Di, Guangyao Zhai, Gim Hee Lee, and Federico Tombari. GeoGaussian: Geometry-Aware Gaussian Splatting for Scene Rendering. In *ECCV*, pages 441–457, 2024.

[34] Yue Li, Qi Ma, Runyi Yang, Huapeng Li, Mengjiao Ma, Bin Ren, Nikola Popovic, Nicu Sebe, Ender Konukoglu, Theo Gevers, et al. SceneSplat: Gaussian Splatting-Based Scene Understanding with Vision-Language Pretraining. *arXiv preprint arXiv:2503.18052*, 2025.

[35] 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 *CVPR*, pages 8456–8455, 2023.

[36] Feng Liang, Bichen Wu, Xiaoliang Dai, Kunpeng Li, Yinan Zhao, Hang Zhang, Peizhao Zhang, Peter Vajda, and Diana Marculescu. Open-Vocabulary Semantic Segmentation with Mask-Adapted CLIP. In *CVPR*, pages 7061–7070, 2023.

[37] Yang Liu, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. CityGaussian: Real-Time High-Quality Large-Scale Scene Rendering with Gaussians. In *ECCV*, pages 265–282, 2024.

[38] Yang Liu, Chuanchen Luo, Zhongkai Mao, Junran Peng, and Zhaoxiang Zhang. CityGaussianV2: Efficient and Geometrically Accurate Reconstruction for Large-Scale Scenes. *arXiv preprint arXiv:2411.00771*, 2024.

[39] Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. ManiGaussian: Dynamic Gaussian Splatting for Multi-Task Robotic Manipulation. In *ECCV*, pages 349–366. Springer, 2024.

[40] 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 *CVPR*, pages 20654–20664, 2024.

[41] Huaishao Luo, Junwei Bao, Youzheng Wu, Xiaodong He, and Tianrui Li. SegCLIP: Patch Aggregation with Learnable Centers for Open-Vocabulary Semantic Segmentation. In *ICML*, pages 23033–23044, 2023.

[42] 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.

[43] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM SIGGRAPH Asia*, 2024.

[44] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM TOG*, 41(4):1–15, 2022.

[45] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis. In *CVPR*, pages 10349–10358, 2024.

[46] Michael Oechsle, Songyou Peng, and Andreas Geiger. UniSurf: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *ICCV*, pages 5589–5599, 2021.

[47] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. LangSplat: 3D Language Gaussian Splatting. In *CVPR*, pages 20051–20060, 2024.

[48] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, pages 8748–8763, 2021.

[49] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-GS: Towards Consistent Real-Time Rendering with LOD-Structured 3D Gaussians. *arXiv preprint arXiv:2403.17898*, 2024.

[50] Ji Shi, Xianghua Ying, Ruohao Guo, Bowei Xing, and Wenzhen Yue. Normal-NeRF: Ambiguity-Robust Normal Estimation for Highly Reflective Scenes. In *AAAI*, volume 39, pages 6869–6877, 2025.

[51] Yahao Shi, Yanmin Wu, Chenming Wu, Xing Liu, Chen Zhao, Haocheng Feng, Jian Zhang, Bin Zhou, Errui Ding, and Jingdong Wang. GIR: 3D Gaussian Inverse Rendering for Relightable Scene Factorization. *IEEE TPAMI*, 2025.

[52] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The Replica Dataset: A Digital Replica of Indoor Spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[53] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben P. Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable Large Scene Neural View Synthesis. In *CVPR*, pages 8238–8248, 2022.

[54] Shengji Tang, Weicai Ye, Peng Ye, Weihao Lin, Yang Zhou, Tao Chen, and Wanli Ouyang. HiSplat: Hierarchical 3D Gaussian Splatting for Generalizable Sparse-View Reconstruction. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *ICLR*, volume 2025, pages 48436–48449, 2025.

[55] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. In *CVPR*, pages 5481–5490, 2022.

[56] Evangelos Ververas, Rolandos Alexandros Potamias, Jifei Song, Jiankang Deng, and Stefanos Zafeiriou. SAGS: Structure-Aware 3D Gaussian Splatting. In *ECCV*, pages 221–238, 2024.

[57] Peihao Wang, Yuehao Wang, Dilin Wang, Sreyas Mohan, Zhiwen Fan, Lemeng Wu, Ruisi Cai, Yu-Ying Yeh, Zhangyang Wang, Qiang Liu, et al. Steepest Descent Density Control for Compact 3D Gaussian Splatting. In *CVPR*, pages 26663–26672, 2025.

[58] 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.

[59] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. NeuS2: Fast Learning of Neural Implicit Surfaces for Multi-View Reconstruction. In *ICCV*, pages 3295–3306, 2023.

[60] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Sssessment: from Error Visibility to Structural Similarity. *IEEE TIP*, 13(4):600–612, 2004.

[61] Meng Wei, Qianyi Wu, Jianmin Zheng, Hamid Rezatofighi, and Jianfei Cai. Normal-GS: 3D Gaussian Splatting with Normal-Involved Rendering. *arXiv preprint arXiv:2410.20593*, 2024.

[62] Tianci Wen, Zhiang Liu, and Yongchun Fang. SEGS-SLAM: Structure-Enhanced 3D Gaussian Splatting SLAM with Appearance Embedding. *arXiv preprint arXiv:2501.05242*, 2025.

[63] Qi Wu, Janick Martinez Esturo, Ashkan Mirzaei, Nicolas Moenne-Loccoz, and Zan Gojcic. 3DGUT: Enabling Distorted Cameras and Secondary Rays in Gaussian Splatting. *CVPR*, pages 26036–26046, 2025.

[64] Tong Wu, Jia-Mu Sun, Yu-Kun Lai, Yuewen Ma, Leif Kobbelt, and Lin Gao. DeferredGS: Decoupled and Editable Gaussian Splatting with Deferred Shading. *arXiv preprint arXiv:2404.09412*, 2024.

[65] Yusen Xie, Zhenmin Huang, Jin Wu, and Jun Ma. GS-LIVM: Real-time photo-realistic LiDAR-Inertial-Visual mapping with Gaussian splatting. In *ICCV*, pages 26869–26878, 2025.

[66] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street Gaussians: Modeling Dynamic Urban Scenes with Gaussian Splatting. In *ECCV*, pages 156–173, 2024.

[67] Runyi Yang, Zhenxin Zhu, Zhou Jiang, Baijun Ye, Xiaoxue Chen, Yifei Zhang, Yuantao Chen, Jian Zhao, and Hao Zhao. Spectrally Pruned Gaussian Fields with Neural Compensation. *arXiv preprint arXiv:2405.00676*, 2024.

[68] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume Rendering of Neural Implicit Surfaces. *NeurIPS*, 34:4805–4815, 2021.

[69] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. In *ACM SIGGRAPH*, pages 1–9, 2023.

[70] Kai Ye, Chong Gao, Guanbin Li, Wenzheng Chen, and Baoquan Chen. GeoSplatting: Towards Geometry Guided Gaussian Splatting for Physically-Based Inverse Rendering. *arXiv preprint arXiv:2410.24204*, 2024.

[71] Keyang Ye, Qiming Hou, and Kun Zhou. 3D Gaussian Splatting with Deferred Reflection. In *ACM SIGGRAPH*, pages 1–10, 2024.

[72] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian Grouping: Segment and Edit Anything in 3D Scenes. In *ECCV*, pages 162–179, 2024.

[73] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields Without Neural Networks. In *CVPR*, volume 2, page 6, 2021.

[74] Mulin Yu, Tao Lu, Linning Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai. GSDF: 3DGS Meets SDF for Improved Neural Rendering and Reconstruction. *NeurIPS*, 37:129507–129530, 2024.

[75] 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.

[76] Baowen Zhang, Chenxing Jiang, Heng Li, Shaojie Shen, and Ping Tan. Geometry-Grounded Gaussian Splatting. *arXiv preprint arXiv:2601.17835*, 2026.

[77] Hao Zhang, Feng Li, Xueyan Zou, Shilong Liu, Chunyuan Li, Jianwei Yang, and Lei Zhang. A Simple Framework for Open-Vocabulary Segmentation and Detection. In *ICCV*, pages 1020–1031, 2023.

[78] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, pages 586–595, 2018.

[79] Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-GS: Density Control with Pixel-Aware Gradient for 3D Gaussian Splatting. In *ECCV*, pages 326–342, 2024.

[80] Wenzhao Zheng, Junjie Wu, Yao Zheng, Sicheng Zuo, Zixun Xie, Longchao Yang, Yong Pan, Zhihui Hao, Peng Jia, Xianpeng Lang, et al. GaussianAD: Gaussian-Centric End-to-End Autonomous Driving. *arXiv preprint arXiv:2412.10371*, 2024.

[81] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place Scene Labelling and Understanding with Implicit Scene Representation. In *ICCV*, pages 15838–15847, 2021.

[82] Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. HuGS: Holistic Urban 3D Scene Understanding via Gaussian Splatting. In *CVPR*, pages 21336–21345, 2024.

[83] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields. In *CVPR*, pages 21676–21685, 2024.

[84] Siting Zhu, Guangming Wang, Xin Kong, Dezhi Kong, and Hesheng Wang. 3D Gaussian Splatting in Robotics: A Survey. *arXiv preprint arXiv:2410.12262*, 2024.

A Methodology

In Sec. A.1, we provide additional details on solving the intersection points between pixel rays and Gaussians. In Sec. A.2, we focus on deriving the analytical solution for propagating depth gradients to the Gaussian parameters, specifically addressing the transition from depth to rotation, scale and positions. In Sec. A.3, we include the implementation details on normal estimation from depth (Alg. 1 and Alg. 2). In Sec. A.4, we develop the derivation of the gradient propagation from the cosine similarity loss of the normal estimation to the rendered depth map, enabling the normal gradients to flow back to depth gradient. This connection allows the normal gradients to be backpropagated further, thus linking the normal consistency loss directly to the Gaussian parameters. In Sec. A.5, we present semantic rendering and backpropagation, the key module underlying our unified gaussian architecture. In Sec. A.6, we expand the discussion on differentiable pruning, with a particular focus on elucidating why $k = 1$ serves as the optimization objective. Finally, in Sec. A.7, we provide the implementation process for the backward rasterization gradient propagation corresponding to Alg. 3.

A.1 Ray-Ellipse Intersection Problem

The ray-ellipse intersection problem in (5) aim to solve for t in the equation of a unit sphere

$$\|\mathbf{v}_s + t\mathbf{d}_s\|^2 = 1, \tag{13}$$

which simplifies to the quadratic equation:

$$at^2 + bt + c = 0, \tag{14}$$

where the coefficients are defined as:

$$\begin{aligned} a &= \mathbf{d}_s \cdot \mathbf{d}_s, \\ b &= 2\mathbf{v}_s \cdot \mathbf{d}_s, \\ c &= \mathbf{v}_s \cdot \mathbf{v}_s - 1. \end{aligned} \tag{15}$$

The solutions are given by $t = \frac{-b \pm \sqrt{\Delta}}{2a}$, where $\Delta = b^2 - 4ac$. This is a simple quadratic equation solving problem.

A.2 Backward Propagation for Depth Rendering

The gradient of the loss $\mathcal{L}_{depth}$ with respect to the a single Gaussian parameters $\boldsymbol{\theta} \in \{\boldsymbol{\mu}, \mathbf{s}, \mathbf{R}\}$ is propagated by

$$\frac{\partial \mathcal{L}_{depth}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}_{depth}}{\partial \tilde{\mathbf{D}}} \cdot \frac{\partial \tilde{\mathbf{D}}}{\partial d_i} \cdot \frac{\partial d_i}{\partial \boldsymbol{\theta}}, \tag{16}$$

where $\frac{\partial \tilde{\mathbf{D}}}{\partial d_i}$ can be calculated from (7). The primary challenge is to compute the gradients of d_i with respect to the Gaussian parameters $\boldsymbol{\mu}$, $\mathbf{s}$, and $\mathbf{R}$, which requires backpropagating through the ray-ellipsoid intersection. The gradient flow is structured as

$$\frac{\partial d_i}{\partial \boldsymbol{\theta}} = \frac{\partial d_i}{\partial t_{\text{mid}}} \cdot \frac{\partial t_{\text{mid}}}{\partial (a,b,c)} \cdot \frac{\partial (a,b,c)}{\partial (\mathbf{v}_s, \mathbf{d}_s)} \cdot \frac{\partial (\mathbf{v}_s, \mathbf{d}_s)}{\partial (\mathbf{v}_l, \mathbf{d}_l)} \cdot \frac{\partial (\mathbf{v}_l, \mathbf{d}_l)}{\partial \boldsymbol{\theta}}. \tag{17}$$

The key steps in this chain will be derived below.

Gradient of Depth w.r.t. t_{mid}. The depth d_i of the intersection point is a function of the midpoint ray parameter t_{mid}. The gradient is computed as the dot product of the ray direction $\mathbf{d}$ with the z-axis of the camera view matrix $\mathbf{R}^{\text{cam}}_{\text{world}}$:

$$\frac{\partial d_i}{\partial t_{\text{mid}}} = \mathbf{R}^{\text{cam}}_{\text{world}}[2,0] \cdot \mathbf{d} \to x + \mathbf{R}^{\text{cam}}_{\text{world}}[2,1] \cdot \mathbf{d} \to y + \mathbf{R}^{\text{cam}}_{\text{world}}[2,2] \cdot \mathbf{d} \to z. \tag{18}$$

Gradient of t_{mid} w.r.t. Quadratic Coefficients. The midpoint parameter $t_{\text{mid}} = (t_1 + t_2)/2 = -b/(2a)$ is a function of the quadratic coefficients a and b. Its derivatives are

$$\frac{\partial t_{\text{mid}}}{\partial a} = \frac{b}{2a^2}, \ \frac{\partial t_{\text{mid}}}{\partial b} = -\frac{1}{2a}, \ \frac{\partial t_{\text{mid}}}{\partial c} = 0. \tag{19}$$

Gradients of Coefficients w.r.t. Local Coordinates. The coefficients a, b, and c are functions of the scaled local ray origin $\mathbf{v}_s$ and direction $\mathbf{d}_s$ from (15). Their gradients are

$$\frac{\partial a}{\partial \mathbf{d}_s} = 2\mathbf{d}_s,\ \frac{\partial b}{\partial \mathbf{v}_s} = 2\mathbf{d}_s,\ \frac{\partial b}{\partial \mathbf{d}_s} \quad = 2\mathbf{v}_s,\ \frac{\partial c}{\partial \mathbf{v}_s} = 2\mathbf{v}_s. \tag{20}$$

Applying the chain rule through t_{mid} and the coefficients, the gradients w.r.t. the *local* (l) *scaled* (s) coordinates are:

$$\begin{aligned} \frac{\partial t_{\text{mid}}}{\partial \mathbf{v}_s} &= \frac{\partial t_{\text{mid}}}{\partial b} \cdot \frac{\partial b}{\partial \mathbf{v}_s} + \frac{\partial t_{\text{mid}}}{\partial c} \cdot \frac{\partial c}{\partial \mathbf{v}_s} = -\frac{1}{a} \cdot \mathbf{d}_s, \\ \frac{\partial t_{\text{mid}}}{\partial \mathbf{d}_s} &= \frac{\partial t_{\text{mid}}}{\partial a} \cdot \frac{\partial a}{\partial \mathbf{d}_s} + \frac{\partial t_{\text{mid}}}{\partial b} \cdot \frac{\partial b}{\partial \mathbf{d}_s} = \frac{b}{a^2} \cdot \mathbf{d}_s - \frac{1}{a} \cdot \mathbf{v}_s. \end{aligned} \tag{21}$$

Gradients w.r.t. Scale $\mathbf{s}$. The *local* (l) *scaled* (s) coordinates are defined as $\mathbf{v}_s = \mathbf{v}_l \oslash \mathbf{s}$ and $\mathbf{d}_s = \mathbf{d}_l \oslash \mathbf{s}$, where $\oslash$ denotes element-wise division. The gradients with respect to the scale $\mathbf{s}$ are computed as:

$$\frac{\partial(\mathbf{v}_s,\, \mathbf{d}_s)}{\partial \mathbf{s}} = -\mathbf{v}_{l,k} \oslash \mathbf{s} - \mathbf{d}_{l,k} \oslash \mathbf{s}. \tag{22}$$

Gradients w.r.t. Position $\boldsymbol{\mu}$ and Rotation $\mathbf{R}$. The local coordinates before scaling are obtained by rotating the world-space vectors into the Gaussian's local frame: $\mathbf{v}_l = \mathbf{R}^\top \mathbf{v}$ and $\mathbf{d}_l = \mathbf{R}^\top \mathbf{d}$, where $\mathbf{v} = \mathbf{o} - \boldsymbol{\mu}$. The gradient with respect to the position $\boldsymbol{\mu}$ is

$$\frac{\partial d_i}{\partial \boldsymbol{\mu}} = -\mathbf{R} \left(\frac{\partial d_i}{\partial \mathbf{v}_l} \oslash \mathbf{s} \right). \tag{23}$$

The gradient with respect to the rotation matrix $\mathbf{R}$ is accumulated from both $\mathbf{v}$ and $\mathbf{d}$:

$$\frac{\partial d_i}{\partial \mathbf{R}} = \left(\frac{\partial d_i}{\partial \mathbf{v}_l} \oslash \mathbf{s} \right) \mathbf{v}^\top + \left(\frac{\partial d_i}{\partial \mathbf{d}_l} \oslash \mathbf{s} \right) \mathbf{d}^\top. \tag{24}$$

Finally, the gradient $\frac{\partial d_i}{\partial \mathbf{R}}$ is converted to a gradient $\frac{\partial \mathbf{R}}{\partial \mathbf{q}}$ with respect to the quaternion $\mathbf{q}$ using the standard adjoint-sensitive conversion, ensuring the constraint of staying on the S^3 manifold through tangent space projection.

A.3 Normal Estimation from Depth

Following the rendering of the depth map $\tilde{\mathbf{D}}$, we estimate a corresponding surface normal map $\tilde{\mathbf{N}}$ in a post-processing step. In this supplementary section, we will elaborate on two key operations of Alg. 1: backprojecting depth pixels and orienting towards the view direction.

Depth Backprojection to 3D Points. Each pixel $p = (u, v)$ with depth value $\tilde{\mathbf{D}}(u, v)$ is backprojected to a 3D point $\mathbf{P}_{\text{cam}}$ in the camera coordinate system using the intrinsic camera parameters:

$$\mathbf{P}_{\text{world}} = \begin{bmatrix} \mathbf{R}_{\text{cam}}^{\text{world}} & \mathbf{t}_{\text{cam}}^{\text{world}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \frac{(u-c_x)}{f_x} \cdot \tilde{\mathbf{D}}(u, v) \\ \frac{(v-c_y)}{f_y} \cdot \tilde{\mathbf{D}}(u, v) \\ \tilde{\mathbf{D}}(u, v) \\ 1 \end{bmatrix}, \tag{25}$$

where $c_x = W/2$, $c_y = H/2$ are the coordinates of the principal point, and f_x, f_y are the focal lengths.

View Direction Correction. Finally, the normal's orientation is corrected to face the camera. The normal is flipped if it points away from the camera, determined by the sign of the dot product between the normalized view direction $\mathbf{d}_{\text{view}}$ and the normal:

$$\tilde{\mathbf{N}}(u, v) \leftarrow \begin{cases} \tilde{\mathbf{N}}(u, v), & \text{if } \tilde{\mathbf{N}}(u, v) \cdot \mathbf{d}_{\text{view}} \leq 0 \\ -\tilde{\mathbf{N}}(u, v), & \text{otherwise} \end{cases}, \tag{26}$$

where $\mathbf{d}_{\text{view}} = \frac{\mathbf{C} - \mathbf{P}_{\text{world}}}{\|\mathbf{C} - \mathbf{P}_{\text{world}}\|}$ is the normalized vector from the 3D point $\mathbf{P}_{\text{world}}$ to the camera center $\mathbf{C}$. This ensures all normals adhere to a consistent front-facing convention.

Algorithm 1: Surface Normal Estimation by Depth

Input: Depth $\tilde{\mathbf{D}}\,(u,v)$, `STEP_1`, `STEP_2`, λ
Output: Normal $\tilde{\mathbf{N}}(u,v)$
for *each pixel* $p=(u,v)$ *in parallel* **do**
  1. Backproject to World Coordinate
  2. Compute Multi-Scale Normals
    for $g_i \in \{\texttt{STEP_1}, \texttt{STEP_2}\}$ **do**
      Finite differences of neighboring points: $\mathbf{v}_{x,i} = \mathbf{P}(u+g_i,v) - \mathbf{P}(u-g_i,v)$, $\mathbf{v}_{y,i} = \mathbf{P}(u,v+g_i) - \mathbf{P}(u,v-g_i)$,
      Estimate normal: $\mathbf{n}_i = \mathbf{v}_{x,i} \times \mathbf{v}_{y,i}$
  3. Fuse and Normalize
    Ensure consistency: $\mathbf{n}_2 \leftarrow \mathbf{n}_2 \cdot \text{sign}(\mathbf{n}_1 \cdot \mathbf{n}_2)$
    Fuse estimates: $\mathbf{n}_{\text{fused}} = \lambda \mathbf{n}_1 + (1-\lambda)\mathbf{n}_2$
    Normalize: $\tilde{\mathbf{N}}(u,v) = \mathbf{n}_{\text{fused}} / \|\mathbf{n}_{\text{fused}}\|$
  4. Orient Towards View Direction

Algorithm 2: Backward Propagation from Normals to Depth

Input: $\frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{N}}}$, `STEP_1`, `STEP_2`, λ
Output: $\frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{D}}}$
for *each pixel* $p=(u,v)$ *in parallel* **do**
  1. View-Facing Orientation Flip (30)$\leftarrow \frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{N}}(u,v)}$
  2. Grad Normalization (31)
  3. Grad Fusion (32)$\leftarrow \lambda$
  4. Compute Multi-Scale Gradient
    for $\texttt{STEP_X} \in \{\texttt{STEP_1}, \texttt{STEP_2}\}$ **do**
      Gradient Atom Addition (33) (34)
  5. Backprojection Gradient (35)
  $\frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{D}}(u,v)} \leftarrow$ (27)

A.4 Backward Propagation from Normals to Depth

To enable optimization via losses defined on the rendered normal map $\tilde{\mathbf{N}}$, the gradients must be backpropagated through the normal estimation pipeline to the depth map $\tilde{\mathbf{D}}$ and subsequently to the underlying 3D Gaussian parameters. Overall gradient backpropagation pipeline is shown in Alg. 2. The gradient of the loss $\mathcal{L}_{normal}$ with respect to the depth map $\tilde{\mathbf{D}}$ is propagated by

$$\frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{D}}(u,v)} = \frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{N}}(u,v)} \cdot \frac{\partial \tilde{\mathbf{N}}(u,v)}{\partial \mathbf{n}_{\text{fused}}^{(\text{orient})}} \cdot \frac{\partial \mathbf{n}_{\text{fused}}^{(\text{orient})}}{\partial \mathbf{n}_{\text{fused}}} \cdot \frac{\partial \mathbf{n}_{\text{fused}}}{\partial (\mathbf{n}_1, \mathbf{n}_2)} \cdot \left(\frac{\partial \mathbf{n}_1}{\partial \mathbf{P}_{\text{world}}^{n_1}} + \frac{\partial \mathbf{n}_2}{\partial \mathbf{P}_{\text{world}}^{n_2}} \right) \cdot \frac{\partial \mathbf{P}_{\text{world}}}{\partial \tilde{\mathbf{D}}(u,v)}, \tag{27}$$

where $\mathbf{P}_{\text{world}}^{n_1}$ and $\mathbf{P}_{\text{world}}^{n_2}$ represent the sets of 3D points in the neighborhoods used to compute $\mathbf{n}_1$ and $\mathbf{n}_2$, respectively.

Cosine Similarity w.r.t. Normal. For encouraging geometric consistency (11) is the negative cosine similarity between the predicted normal $\tilde{\mathbf{N}}$ and a target normal $\tilde{\mathbf{N}}_{\text{target}}$:

$$\mathcal{L}_{\text{normal}} = 1 - \frac{1}{|\Omega|} \sum_{(u,v)\in\Omega} \tilde{\mathbf{N}}(u,v) \cdot \tilde{\mathbf{N}}_{\text{target}}(u,v), \tag{28}$$

where Ω denotes the set of pixels. The gradient of this loss with respect to the predicted normal at a specific pixel is straightforward:

$$\frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{N}}(u,v)} = -\frac{1}{|\Omega|} \tilde{\mathbf{N}}_{\text{target}}(u,v). \tag{29}$$

This gradient $\frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{N}}}$ is the seed that initiates the backward pass through the normal estimation pipeline.

View-facing Orientation Flip. If the normal is flipped in the forward pass to face the viewer (defined by (26)), the incoming gradient is correspondingly flipped:

$$\frac{\partial \mathcal{L}_{normal}}{\partial \mathbf{n}_{\text{fused}}^{(\text{orient})}} = \begin{cases} -\frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{N}}(u,v)}, & \text{if flipped} \\ \frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{N}}(u,v)}, & \text{otherwise} \end{cases}, \tag{30}$$

Normalization. The gradient is then backpropagated through the normalization operation $\tilde{\mathbf{N}} = \mathbf{n}_{\text{fused}}/\|\mathbf{n}_{\text{fused}}\|$. The gradient w.r.t. the unnormalized vector $\mathbf{n}_{\text{fused}}$ is given by:

$$\frac{\partial \mathbf{n}_{\text{fused}}^{(\text{orient})}}{\partial \mathbf{n}_{\text{fused}}} = \frac{\left(\mathbf{I} - \tilde{\mathbf{N}}(u,v)\tilde{\mathbf{N}}(u,v)^\top\right)}{\|\mathbf{n}_{\text{fused}}\|}. \tag{31}$$

Normal Fusion. The gradient is distributed to the two constituent normal estimates $\mathbf{n}_1$ and $\mathbf{n}_2$ from which $\mathbf{n}_{\text{fused}} = \lambda \mathbf{n}_1 + (1-\lambda)\mathbf{n}_2$ is computed. If their directions were inconsistent in the forward pass ($\mathbf{n}_1 \cdot \mathbf{n}_2 < 0$), the sign of the gradient for $\mathbf{n}_2$ is flipped:

$$\frac{\partial \mathbf{n}_{\text{fused}}}{\partial \mathbf{n}_1} = \lambda, \; \frac{\partial \mathbf{n}_{\text{fused}}}{\partial \mathbf{n}_2} = (1-\lambda) \cdot \text{sign}(\mathbf{n}_1 \cdot \mathbf{n}_2). \tag{32}$$

Finite Differences to 3D Points. The core of the backward pass involves computing how the estimated normals $\mathbf{n_1}$ and $\mathbf{n_2}$ change with respect to the 3D points $\mathbf{P}_{\text{world}}$ in their finite-difference neighborhood. For simplicity, we will take $\mathbf{n_X} \in \{\mathbf{n_1}, \mathbf{n_2}\}$, $\texttt{STEP_X} \in \{\texttt{STEP_1}, \texttt{STEP_2}\}$ as an example for the subsequent derivation. For a normal estimate $\mathbf{n_X}$ calculated from points $\mathbf{P}$ in a $\texttt{STEP_X} \times \texttt{STEP_X}$ window, the gradient w.r.t. each involved point $\mathbf{P}_{\text{world}}^{\mathbf{nx}}$ is derived from the cross product operation.

From Alg. 1, $\mathbf{v}_x^{\mathbf{nx}} = \mathbf{P}_{\text{world}}^{\mathbf{nx}}(u + \texttt{STEP_X}, v) - \mathbf{P}_{\text{world}}^{\mathbf{nx}}(u - \texttt{STEP_X}, v)$ and $\mathbf{v}_y^{\mathbf{nx}} = \mathbf{P}_{\text{world}}^{\mathbf{nx}}(u, v + \texttt{STEP_X}) - \mathbf{P}_{\text{world}}^{\mathbf{nx}}(u, v - \texttt{STEP_X}))$, the normal is $\mathbf{n_X} = \mathbf{v}_x^{\mathbf{nx}} \times \mathbf{v}_y^{\mathbf{nx}}$. The gradients of $\mathbf{n_X}$ w.r.t. the vectors $\mathbf{v}_x^{\mathbf{nx}}$ and $\mathbf{v}_y^{\mathbf{nx}}$ are:

$$\frac{\partial \mathbf{n_X}}{\partial \mathbf{v}_x^{\mathbf{nx}}} = \mathbf{v}_y^{\mathbf{nx}}, \; \frac{\partial \mathbf{n_X}}{\partial \mathbf{v}_y^{\mathbf{nx}}} = \mathbf{v}_x^{\mathbf{nx}}. \tag{33}$$

These gradients are then distributed to the specific points that contributed to $\mathbf{v}_x^{\mathbf{nx}}$ and $\mathbf{v}_y^{\mathbf{nx}}$:

$$\begin{aligned} \frac{\partial \mathbf{n_X}}{\partial \mathbf{P}_{\text{world}}^{\mathbf{nx}}(u + \texttt{STEP_X}, v)} &+= \mathbf{v}_x^{\mathbf{nx}}, \quad \frac{\partial \mathbf{n_X}}{\partial \mathbf{P}_{\text{world}}^{\mathbf{nx}}(u - \texttt{STEP_X}, v)} -= \mathbf{v}_x^{\mathbf{nx}}, \\ \frac{\partial \mathbf{n_X}}{\partial \mathbf{P}_{\text{world}}^{\mathbf{nx}}(u, v + \texttt{STEP_X})} &+= \mathbf{v}_y^{\mathbf{nx}}, \quad \frac{\partial \mathbf{n_X}}{\partial \mathbf{P}_{\text{world}}^{\mathbf{nx}}(u, v - \texttt{STEP_X})} -= \mathbf{v}_y^{\mathbf{nx}}. \end{aligned} \tag{34}$$

This process is repeated for *both* normal estimates $\mathbf{n}_1$ and $\mathbf{n}_2$ and their respective neighborhoods $\mathbf{P}_{\text{world}}^{n_1}$ and $\mathbf{P}_{\text{world}}^{n_2}$, and the gradients are accumulated atomically.

3D Points w.r.t Depth. The Jacobian matrix $\frac{\partial \mathbf{P}_{\text{world}}}{\partial \tilde{\mathbf{D}}(u,v)}$ is derived from the backprojection and transformation chain $\mathbf{P}_{\text{world}} = \mathbf{R}_{\text{cam}}^{\text{world}} \mathbf{P}_{\text{cam}}(\tilde{\mathbf{D}}(u,v))$:

$$\frac{\partial \mathbf{P}_{\text{world}}}{\partial \tilde{\mathbf{D}}(u,v)} = \mathbf{R}_{\text{cam}}^{\text{world}} \frac{\partial \mathbf{P}_{\text{cam}}}{\partial \tilde{\mathbf{D}}(u,v)} = \mathbf{R}_{\text{cam}}^{\text{world}} \begin{bmatrix} (u - c_x)/f_x \\ (v - c_y)/f_y \\ 1 \\ 0 \end{bmatrix}. \tag{35}$$

A.5 Semantic Rendering and Backpropagation

We render a 2D semantic logits $\tilde{\mathbf{O}} \in \mathbb{R}^{H \times W \times C_{\mathbf{o}}}$ along with rasterization pipeline. The forward process for the semantic logits in pixel $p = (u, v)$ and channel ch is defined as

$$\tilde{\mathbf{O}}_{[ch]}(u,v) = \sum_{i \in \mathcal{N}} \alpha_i \mathbf{o}_{i[ch]} \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{36}$$

Then the rendered semantic logits $\tilde{\mathbf{O}}$ will be compared to its ground truth $\mathbf{O}^{\text{gt}}$ using Cross-Entropy Loss, resulting in semantic loss $\mathcal{L}_{\text{seg}}$. Let $\frac{\partial \mathcal{L}_{\text{seg}}}{\partial \tilde{\mathbf{o}}_{[ch]}}$ be the gradient of the loss with respect to the rendered ch-th channel of a specific pixel. The chain rule for the gradient of $\mathcal{L}_{\text{seg}}$ with respect to a specific semantic channel $\mathbf{o}_{i[ch]}$ of a Gaussian $\mathcal{G}$ is given by

$$\frac{\partial \mathcal{L}_{\text{seg}}}{\partial \mathbf{o}_{i[ch]}} = \frac{\partial \mathcal{L}_{\text{seg}}}{\partial \tilde{\mathbf{O}}_{[ch]}} \cdot \frac{\partial \tilde{\mathbf{O}}_{[ch]}}{\partial \mathbf{o}_{i[ch]}}, \tag{37}$$

where $\frac{\partial \tilde{\mathbf{O}}_{[ch]}}{\partial \mathbf{o}_{i[ch]}}$ can be derived from (36).

A.6 Differentiable Pruning

Gradient Propagation. During backpropagation of k, the gradient can be propagated to the gradient factor k through the rendering equation. The chain rule for the gradient of the loss with respect to a specific k_i can be conceptually expressed as

$$\frac{\partial \mathcal{L}_K(u,v)}{\partial k_i} = \frac{\partial \mathcal{L}_K(u,v)}{\partial \tilde{\mathbf{K}}(u,v)} \cdot \frac{\partial \tilde{\mathbf{K}}(u,v)}{\partial k_i}, \tag{38}$$

where $\frac{\partial \tilde{\mathbf{K}}(u,v)}{\partial k_i}$ can be derived from (8) in main text.

Detailed Analysis of optimal state $k = 1$. The gradient factor k serves as a core indicator for Gaussian pruning. Its deviation from 1 directly reflects the optimization state of Gaussians, and the underlying causes along with pruning rationale are elaborated as follows.

• **Gaussians with $k > 1$: Anomalies from Gradient Fluctuations Caused by Data Noise.** Gaussians with extremely large k-values are induced by gradient anomalies during multi-view optimization. During the rendering of specific views, data noise such as sensor noise and texture inconsistencies at scene edges leads to significant fluctuations in pixel gradients. These anomalous gradients propagate to the opacity α through backpropagation, causing α to deviate from its optimal value. This deviation results in unexpected optimization under specific view. Such abnormal k-values are usually accompanied by extreme Gaussian parameters including position, scale as shown in Fig. 9 in the main text.

• **Gaussians with $k < 1$: Redundancy Due to Insufficient Rendering Participation.** Gaussians with small k-values can be identified based on long-term optimization, especially considering the periodic k-reset strategy. During training, we periodically reset the k-value of all Gaussians to 0.9. This initial value is designed to provide a starting point for optimization driven by $\mathcal{L}_K$. The expectation is that after multiple optimization iterations, well-optimizing Gaussians will adjust their k-values from 0.9 toward 1. The ideal state is when $k = 1$, which makes the rendering formula of $\tilde{\mathbf{K}}$ satisfy

$$\tilde{\mathbf{K}}(u,v) = \sum_{i \in \mathcal{N}} \alpha_i \cdot 1 \cdot \prod_{j=1}^{i-1} (1 - \alpha_j) = 1. \tag{39}$$

If a Gaussian's k-value remains near 0.9 or changes minimally after a long optimization horizon, it indicates the Gaussian has not deeply participated in the rendering of any view. Specifically, it lacks sufficient effective alpha-rendering contributions to pixels across views. This situation will occur because the Gaussian is located in unobservable regions such as persistent occlusions. Such Gaussians are redundant, and their exclusion does not affect the accuracy of scene reconstruction as they contribute little to the rendering of $\tilde{\mathbf{K}}$ or RGB images.

Pruning Mechanism Based on k. In summary, well-optimized Gaussians can adjust their k-values toward 1 under the constraint of $\mathcal{L}_K$, and their α and geometric parameters can well match the scene without the need for extreme k-compensation. In contrast, Gaussians with $|k - 1| > \mathcal{T}_k$ are either anomalous with $k > 1$ or redundant with $k < 1$. By pruning these Gaussians, we can effectively reduce the total number of Gaussians while preserving the quality of scene reconstruction, thereby improving the overall computational efficiency.

Algorithm 3: Multimodal Forward Rasterization Pipeline

Input: $\mathcal{G} = \{\boldsymbol{\mu}, \mathbf{q}, \mathbf{s}, \alpha, \mathbf{h}, \mathbf{o}, \mathbf{k}\}$, Image dimensions W, H
Output: $\tilde{\mathbf{C}} \in \mathbb{R}^{W\times H\times 3}$, $\tilde{\mathbf{O}} \in \mathbb{R}^{W\times H\times C}$, $\tilde{\mathbf{D}} \in \mathbb{R}^{W\times H}$, $\tilde{\mathbf{K}} \in \mathbb{R}^{W\times H}$

for *each pixel* = (u, v) **do in parallel**
  `// Compute Ray of pixel`
  computeRay (3);
  Initialize accumulators: $\mathbf{C} \leftarrow \mathbf{0}$, $\mathbf{O} \leftarrow \mathbf{0}$, $D \leftarrow 0$, $K \leftarrow 0$, $T \leftarrow 0$;
  for *each Gaussian* i *intersecting pixel* (u, v) **do**
    ...
    $\mathbf{h}_i, \mathbf{o}_i, \alpha_i \leftarrow$ Gaussian i parameters;
    `// Ray-Ellipsoid Intersection`
    rayEllipsoidIntersection (5) ;
    `// Use Midpoint for Depth`
    $d_i \leftarrow$ obtainDepth (6) ;
    `// Alpha-Blending for All Attributes`
    for *each ch in* C_h **do**
      $\mathbf{C}_{[ch]} \leftarrow \mathbf{C} + \mathbf{h}_i \cdot \alpha_i \cdot T$;
    for *each ch in* C_o **do**
      $\mathbf{O}_{[ch]} \leftarrow \mathbf{O} + \mathbf{o}_i \cdot \alpha_i \cdot T$;
    $D \leftarrow D + d_i \cdot \alpha_i \cdot T$;
    $K \leftarrow K + t_i \cdot \alpha_i \cdot T$;
    ...
  `// Pixel-Wise Assignment`
  $\tilde{\mathbf{C}}[x, y] \leftarrow \mathbf{C}$; $\tilde{\mathbf{O}}[x, y] \leftarrow \mathbf{O}$;
  $\tilde{\mathbf{D}}[x, y] \leftarrow D$; $\tilde{\mathbf{K}}[x, y] \leftarrow K$;
  Save $T_{\text{final}} = T$;

A.7 Tile-Based Forward Rasterization and Gradient Backpropagation

We extend the original 3DGS's tile-based rendering pipeline [25]. During rasterization, for each pixel $p = (u, v)$, we calculate the direction $\mathbf{d}$ of the camera ray $r(t)$ and then compute the intersection t_1/t_2 with the Gaussian projected onto this pixel. Finally, we obtain the corresponding attributes using alpha blending rendering. The multimodal forward rasterization pipeline is outlined in Alg. 3.

In the previous sections, we obtain the losses for the semantic logits, depth, normal, and gradient factor map with respect to their respective ground truths, which are given by $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{O}}}$, $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{D}}}$, $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{N}}}$, and $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{K}}}$. Corresponding to Alg. 3, we optimize the Gaussian parameters through the backward propagation of gradients via the alpha rendering process. The backward propagation pipeline is illustrated in Alg. 4. We omit the gradient propagation process for RGB (indicated as ...), which is consistent with the original implementation [25].

B More Experiments

B.1 More Qualitative RGB Rendering Evaluation

In Tab. 10 of the main paper, we only present quantitative results for RGB rendering. In this section, we provide qualitative results and part of quantitative results across multiple datasets, as shown in Fig. 7. It can be observed that our method achieves highly accurate geometry while also maintaining strong RGB rendering performance.

B.2 Semantic Rendering Evaluation

We compare our method with the SGS-SLAM [32] framework, which employs a task-specific Gaussian map representation and is supervised using semantic result images for semantic tasks, differing from our unified framework that predicts semantic logits. The comparison across multiple

Algorithm 4: Tile-based Backward Gradient Propagation Pipeline

Input: Forward rendering results: $\tilde{\mathbf{D}}$, $\tilde{\mathbf{N}}$, $\tilde{\mathbf{O}}$, $\tilde{\mathbf{K}}$;
Loss gradients: $\frac{\partial \mathcal{L}_{depth}}{\partial \tilde{\mathbf{D}}}$, $\frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{N}}}$, $\frac{\partial \mathcal{L}_{seg}}{\partial \tilde{\mathbf{O}}}$, $\frac{\partial \mathcal{L}_{K}}{\partial \tilde{\mathbf{K}}}$
Output: Parameter gradients: $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{R}}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{s}}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{o}}$, $\frac{\partial \mathcal{L}}{\partial K}$

for *each pixel* (x, y) **do in parallel**
  `// Restore Saved Forward Values`
  Restore: $T \leftarrow T_{\text{final}}$;
  for *each Gaussian* i *intersecting pixel* pix *(in reverse depth order)* **do**
    ...
    $T \leftarrow T/(1-\alpha_i)$;
    $\mathbf{w}_\alpha \leftarrow \alpha_i \cdot T$;
    `// Compute Gradients for Rendered Attributes`
    for *each* ch *in* C_o **do**
      $\frac{\partial \mathcal{L}_{all}}{\partial \mathbf{o}_{i[ch]}} \leftarrow \underbrace{\frac{\partial \mathcal{L}_{seg}}{\partial \tilde{\mathbf{O}}_{[ch]}}}_{(37)} \cdot \mathbf{w}_\alpha$;
    $\frac{\partial \mathcal{L}_{all}}{\partial k_i} \leftarrow \underbrace{\frac{\partial \mathcal{L}_{K}}{\partial \tilde{\mathbf{K}}}}_{(38)} \cdot \mathbf{w}_\alpha$;
    `// Depth Gradient Propagation`
    $\frac{\partial \mathcal{L}_{all}}{\partial d_i} \leftarrow (\underbrace{\frac{\partial \mathcal{L}_{normal}}{\partial \tilde{\mathbf{N}}}}_{(29)} + \frac{\partial \mathcal{L}_{depth}}{\partial \tilde{\mathbf{D}}}) \cdot \mathbf{w}_\alpha$;
    `// Propagate Depth Gradient to Ellipsoid Parameters`
    $\frac{\partial \mathcal{L}_{all}}{\partial \mathbf{s}} = \frac{\partial \mathcal{L}_{all}}{\partial d_i} \cdot \underbrace{\frac{\partial d_i}{\partial \mathbf{s}}}_{(22)}$;
    $\frac{\partial \mathcal{L}_{all}}{\partial \boldsymbol{\mu}} = \frac{\partial \mathcal{L}_{all}}{\partial d_i} \cdot \underbrace{\frac{\partial d_i}{\partial \boldsymbol{\mu}}}_{(23)}$;
    $\frac{\partial \mathcal{L}_{all}}{\partial \mathbf{q}} = \frac{\partial \mathcal{L}_{all}}{\partial d_i} \cdot \underbrace{\frac{\partial d_i}{\partial \mathbf{R}}}_{(24)} \cdot \frac{\partial \mathbf{R}}{\partial \mathbf{q}}$;
    ...

scenes is shown in Fig. 11 and Tab. 5. From the quantitative results, it can be observed that our method achieves superior performance in both image rendering quality (PSNR) and semantic segmentation quality (mIoU) within a very short training time. Thanks to our unified rendering architecture, the image rendering FPS is also fast. Additional results can be found in our supplementary video.

B.3 Extra Quantitative Experiments on Replica Dataset

We supplement the evaluation with **averaged** quantitative results over all 14 sequences of the Replica synthetic dataset, including baseline comparisons and ablation studies. For precise geometric evaluation, we exclude real-world datasets due to missing accurate depth/normal GT.

The results in Tab. 6 demonstrate that: ① even without depth supervision (w/o $\mathcal{L}_{depth}$), our method still outperforms other baselines in geometry consistency. This is because our normals are derived from depth, enabling gradients to flow from $\mathcal{L}_{normal}$ to the Gaussian primitives without requiring direct depth supervision. In RGB rendering, NormalGS [61] achieves better results as it incorporates factors such as light reflection and uses a larger number of Gaussian primitives. ② After removing

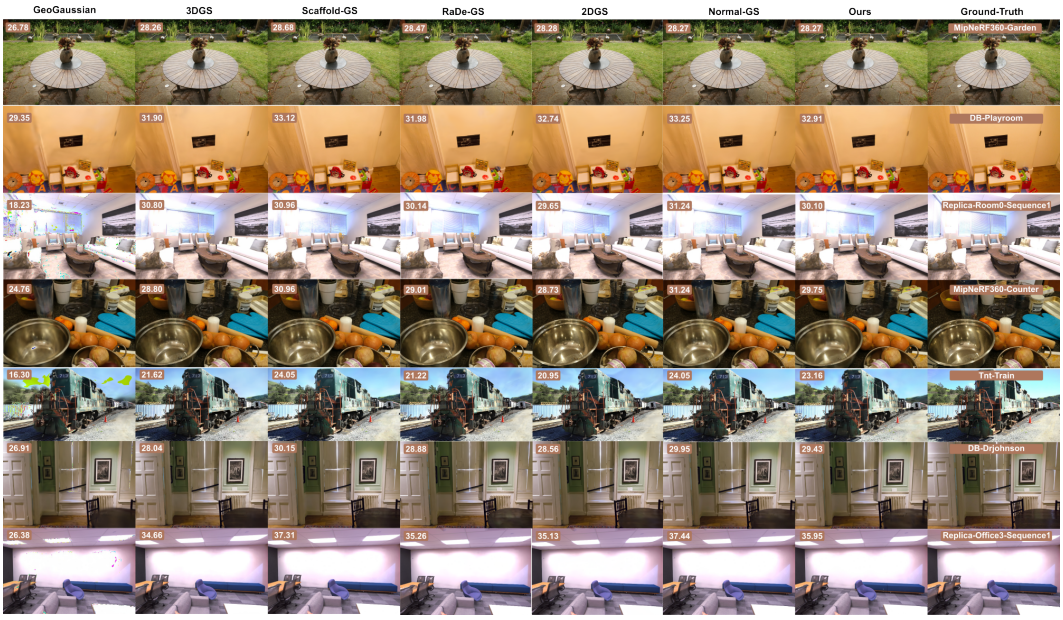


Figure 10: Qualitative comparison results on NVS demonstrate that our RGB rendering achieves better performance in certain scenarios. The top-left corner of the image displays the PSNR value computed against the ground truth image.

Table 5: Quantitative comparison with the SGS-SLAM [32] method in terms of RGB rendering, semantic rendering and FPS. TT represents training time (minutes). Best results are underlined.

Dataset	All	Room0			Room2			Office2		
Method | Metrics	TT↓	PSNR ↑	FPS ↑	mIoU ↑	PSNR ↑	FPS ↑	mIoU↑	PSNR ↑	FPS ↑	mIoU ↑
SGS-SLAM [32]	212.6	36.28	96.4	0.976	39.14	72.2	0.978	36.94	112.3	0.944
Ours	25.8	36.66	184.2	0.987	40.06	234.4	0.984	38.16	229.3	0.975

Table 6: Averaged comparison on Replica dataset. **Bold** is the best.

	PSNR↑	Abs.Rel.↓	CosSimi↑	Count↓	TT↓ (Unit: seconds)
RaDe-GS [75]	36.09	0.0639	0.8017	80392	742
NormalGS [61]	**38.19**	0.0185	0.8941	264359	1306
Ours w/o Seg & w/o $\mathcal{L}_{depth}$	36.68	**0.0109**	**0.9245**	174797	**1219**
Ours w/ Seg & w/o $\mathcal{L}_{depth}$	36.65	0.0110	0.9238	**174261**	1546

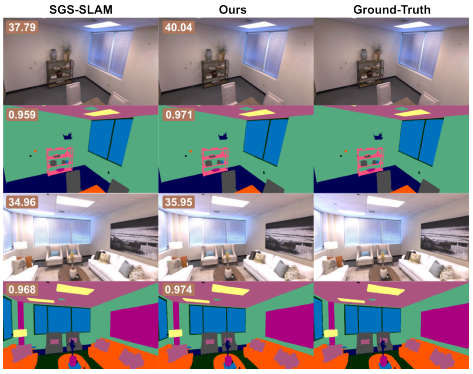


Figure 11: Quantitative comparison between our method and SGS-SLAM [32] in terms of RGB rendering and semantic segmentation clearly shows that our approach outperforms SGS-SLAM. The top-left corner of the image displays the PSNR value computed against the ground truth image.

semantic component (w/o Seg), our geometric optimization yields greater training efficiency. ③ Even with added semantic components (w/ Seg), our method consumes no significantly extra training time (TT) while maintaining comparable performance.

B.4 Statistical Outlier Analysis

To quantify the quality of predicted point clouds relative to ground truth data in Replica [52], we employ a statistical approach based on Z-score analysis of point-to-point distances. This method allows us to identify and measure outliers in predicted point clouds with respect to the ground truth mesh points.

Distance. For each point p_i in the predicted point cloud P, we compute the minimum Euclidean distance to the ground truth point cloud G:

$$d_i = \min_{g \in G} |p_i - g|, \tag{40}$$

where $|\cdot|$ denotes the Euclidean norm. This computation is efficiently implemented using a KD-tree data structure for nearest neighbor search.

Statistical Measures. From the computed distances $d_1, d_2, \ldots, d_n$ where n is the number of points in the predicted point cloud, we calculate the following statistical measures:

$$\mu_d = \frac{1}{n}\sum_{i=1}^{n} d_i, \tag{41}$$

$$\sigma_d = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \mu_d)^2}, \tag{42}$$

where μ_d represents the **Mean** distance and σ_d represents the standard deviation (**Std**) of distances.

For each distance d_i, we compute its Z-score, which measures how many standard deviations the distance is from the mean:

$$z_i = \frac{|d_i - \mu_d|}{\sigma_d}. \tag{43}$$

Points are classified as outliers based on a threshold parameter τ ($\tau = 0.1$ in our experiments):

$$\mathcal{F}_i = \begin{cases} 1 & \text{if } z_i > \tau \\ 0 & \text{otherwise} \end{cases}. \tag{44}$$

The outlier ratio (**Radio**) is then calculated as the proportion of points classified as outliers:

$$\text{Radio} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{F}_i. \tag{45}$$

Hausdorff Distance. In addition to the statistical outlier analysis, we compute the Hausdorff distance to measure the maximum deviation between point clouds:

$$H(G, P) = \max\left\{ \sup_{g\in G} \inf_{p\in P} |g - p|, \sup_{p\in P} \inf_{g\in G} |p - g| \right\}. \tag{46}$$

B.5 More Results on Geometric Rendering

We present additional comparative results of depth and surface normal predictions across multiple datasets in Fig. 12, which demonstrate the superiority of our method in geometric rendering. Particularly in texture-sparse regions, where other methods fail to accurately reconstruct object surfaces, our approach consistently ensures high consistency among RGB, depth, and normal estimations.

It is worth noting that in the Mip-NeRF [1], Tanks & Temples [27], and Deep Blending [16] datasets, the depth and normal information used for supervision is provided by Normal-GS [61] and may still be inaccurate in certain scenarios. Nevertheless, our method achieves competitive results in both depth and normal estimation.

B.6 Reduced-View Evaluation

We further add experiments with reduced training views on *Room0-Sequence1* of the Replica dataset, as reported in Table 7. With the full training set, both formulations perform on par. When 75% of the training views are removed, however, the center-point projection suffers a severe drop, while Ray-Ellipsoid degrades gracefully. This confirms that the Ray-Ellipsoid formulation is substantially more robust under sparse-view settings.

B.7 Ablation on Algorithm Components

We test the quantitative rendering quality and runtime analysis with and without the differentiable pruning module, as shown in Tab. 8. It is evident that our differentiable pruning module effectively reduces the number of Gaussians used without significantly compromising image rendering quality. thereby improving rendering FPS and reducing stray points in the infinite distance.

Table 7: Reduced-view evaluation on *Room0-Sequence1 (Replica)*.

Train Views	Method	PSNR↑	Abs.Rel↓	CS↑
Reduce 0%	Ray-Ellipsoid	31.34	**0.0062**	**0.855**
	Center-point	**31.80**	0.0071	0.849
Reduce 75%	Ray-Ellipsoid	**29.72**	**0.0086**	**0.829**
	Center-point	26.57	0.1049	0.645

Table 8: Quantitative comparisons of the ablation experiments on each module.

Datasets	Room0-Sequence1						Office3-Sequence0					
Settings | Metrics	PSNR↑	Abs.Rel.↓	CosSimi↑	mIoU↑	Count↓	FPS↑	PSNR↑	Abs.Rel.↓	CosSimi↑	mIoU↑	Count↓	FPS↑
RayCasting-Point (All)	31.341	0.0062	0.855	0.607	170859	183.8	36.892	0.0208	0.931	0.534	178905	184.8
Center Point	31.800	0.0071	0.849	0.607	174177	181.7	36.873	0.0204	0.932	0.534	179128	187.8
w/o prune	31.509	0.0070	0.851	0.606	229487	156.3	37.100	0.0315	0.928	0.534	237914	165.5
w/prune $\mathcal{T}_k$=0.2	31.273	0.0068	0.847	0.605	144185	204.1	36.925	0.0161	0.932	0.534	148167	189.7
w/o seg	31.605	0.0226	0.837	0.021	174230	188.0	36.833	0.0225	0.929	0.024	184952	168.9
w/o normal	32.750	0.0061	0.774	0.608	167494	206.5	37.488	0.0089	0.895	0.535	182019	203.8
w/o depth	32.161	0.0184	0.850	0.607	171462	189.7	37.318	0.0985	0.925	0.539	176346	184.1
w/o depth&normal	33.691	0.0214	0.658	0.609	167198	202.6	37.449	0.1106	0.782	0.539	180747	189.7

Furthermore, Tab. 8 and Tab. 9 demonstrates the performance gains achieved by our ray-casting based method on depth and normal estimation, as well as the impact of integrating various loss functions on their respective evaluation metrics. It can be observed that our ray-casting based approach enhances the algorithm's performance on geometric metrics.

Unlike methods that optimize for a single modality, our focus lies in achieving a consistent representation of all modalities in the scene. We therefore strive for a balanced performance across all metrics, which ultimately leads to a more faithful reconstruction of the real world.

Regarding the pruning strategy in Tab. 10, it effectively removes redundant Gaussians to improve speed without significantly compromising accuracy, more like a trade-off between accuracy and efficiency. Naturally, aggressively reducing $\mathcal{T}_k$ will trigger catastrophic degradation.

Table 9: Averaged comparison of point calculation settings.

	RayCasting-Point	Center-Point
Abs.Rel.↓	**0.0109**	0.0136
CosSimi↑	**0.9245**	0.8925

Table 10: Comparison of pruning thresholds.

	w/o prune	$\mathcal{T}_k$=0.3	$\mathcal{T}_k$=0.2	$\mathcal{T}_k$=0.1
PSNR↑	**38.14**	37.23	36.68	32.10
Abs.Rel.↓	**0.0098**	0.0102	0.0109	0.0359
CosSimi↑	**0.9265**	0.9251	0.9245	0.8637
Count↓	251246	189084	**174529**	130456

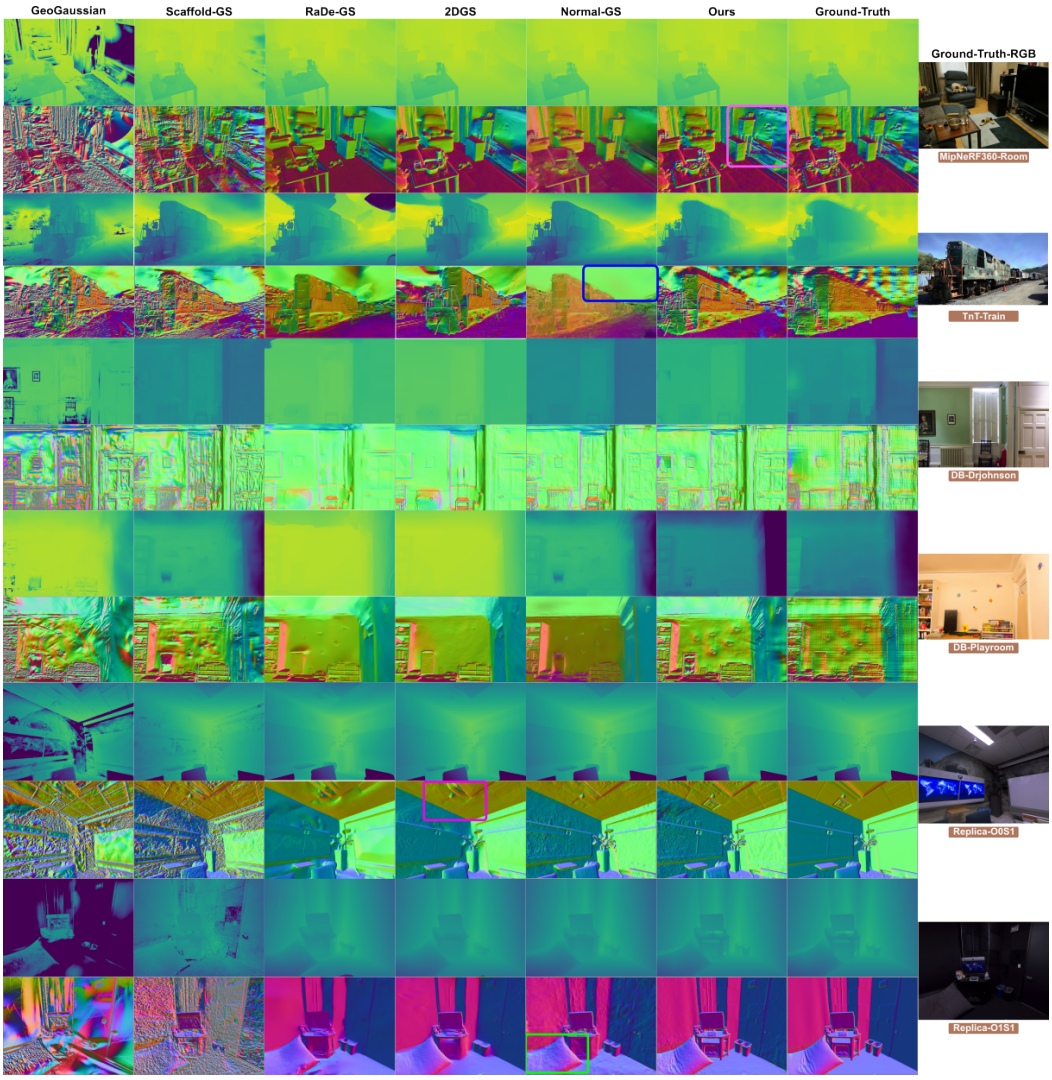


Figure 12: Qualitative comparison of depth and normal estimation accuracy across different methods on several datasets.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our main contribution.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Due to space constraints, the main paper presents the forward process, while the complete proof of our core closed-form solution (backward process) is provided in Section A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide: (1) detailed algorithm description in Section 3 and Section A; (2) hyperparameter settings and training protocols in Section 4; (3) instructions for dataset construction and model training in Section 4. Full code and reproduce guideline will be released upon acceptance.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The project involves substantial core code with significant engineering complexity, which cannot be anonymously released during the double-blind review phase due to intellectual property considerations. However, we commit to releasing the full source code and reproduce guideline upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Standard dataset interfaces, hyperparameters and optimizer are used. Full training details are provided in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our closed-form solution is deterministic under fixed inputs/hyperparameters, and 3DGS evaluation follows unified protocols with fixed test sets. Error bars are thus not applicable.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report computer resources and per-scene training time in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work focuses on deterministic 3D reconstruction algorithms and complies with all NeurIPS ethical guidelines.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: This is a foundational 3D vision method with no direct path to harmful applications.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our work does not release high-risk models or scraped datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets are cited with original licenses. Code dependencies follow their respective open-source licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release our codebase along with detailed README files. The release does not include source code copied from external libraries; all code provided is our own.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects research.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used in the core methodology development.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.