GSDF: 3DGS Meets SDF for Improved Neural Rendering and Reconstruction

Mulin Yu [1*] **Tao Lu**[1*] **Linning Xu**[2] **Lihan Jiang**[4,1] **Yuanbo Xiangli**[3†] **Bo Dai**[5,1]
[1]Shanghai Artificial Intelligence Laboratory, [2]The Chinese University of Hong Kong,
[3]Cornell University, [4] University of Science and Technology of China, [5] The University of Hong Kong

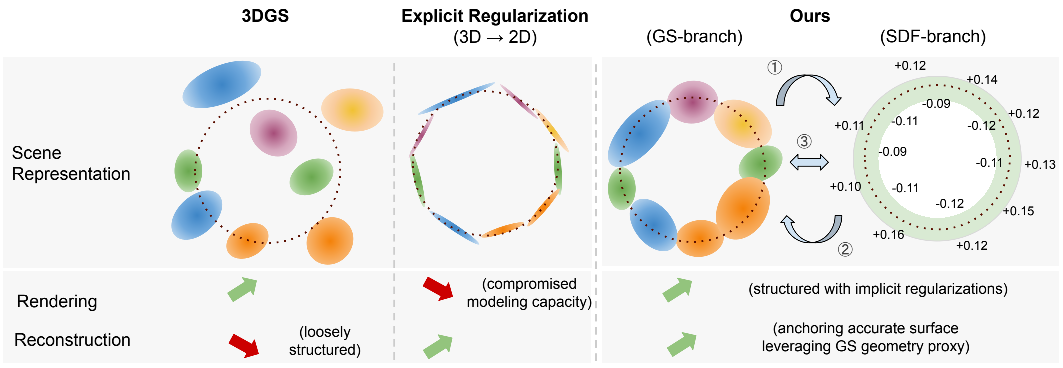

Figure 1: **Conceptual Illustration of GSDF.** Rendering and reconstruction tasks have traditionally involved trade-offs in neural representation methods. While 3D-GS achieves high-fidelity view-dependent rendering, it often compromises on geometric accuracy. Recent approaches [13, 10] use explicit regularization to align Gaussian primitives near surfaces, but this can reduce model capacity for high-fidelity visuals. Our GSDF introduces a dual-branch framework with specialized GS- and SDF-branches for rendering and geometry tasks. We propose three mutual guidances (detailed in Sec. 3.2) to enhance the quality of both tasks.

Abstract

Representing 3D scenes from multiview images remains a core challenge in computer vision and graphics, requiring both reliable rendering and reconstruction, which often conflicts due to the mismatched prioritization of image quality over precise underlying scene geometry. Although both neural implicit surfaces and explicit Gaussian primitives have advanced with neural rendering techniques, current methods impose strict constraints on density fields or primitive shapes, which enhances the affinity for geometric reconstruction at the sacrifice of rendering quality. To address this dilemma, we introduce GSDF, a dual-branch architecture combining 3D Gaussian Splatting (3DGS) and neural Signed Distance Fields (SDF). Our approach leverages mutual guidance and joint supervision *during the training process* to mutually enhance reconstruction and rendering. Specifically, our method guides the Gaussian primitives to locate near potential surfaces and accelerates the SDF convergence. This implicit mutual guidance ensures robustness and accuracy in both synthetic and real-world scenarios. Experimental results demonstrate that our method boosts the SDF optimization process to reconstruct more detailed geometry, while reducing floaters and blurry edge artifacts in rendering by aligning Gaussian primitives with the underlying geometry.

[*]Equal contribution
[†]Corresponding author

1 Introduction

Recent advancements in neural scene representations have showcased superior rendering capabilities [23, 24, 17], these advancements have sparked significant interest in neural surface reconstruction [39, 31, 19, 32, 10, 13]. They seek to develop unified representations, which simultaneously support high-fidelity rendering and accurate geometric reconstruction, to better support downstream applications such as robotics [16, 27], physical simulations [35, 8], and XR applications [36, 15].

Although both neural implicit surfaces and explicit Gaussian primitives have advanced with neural rendering techniques, current methods often suffer from a mismatched prioritization between appearance and geometry. Imposing regularization or constraints on the unified representation may boost one task, while unavoidably deteriorating the performance of the other. Recent approaches [10, 5], have explored using flat Gaussian primitives for surface modeling. Enforcing binary opacity [10] and jointly learned NeuS models [5] to regularize attributes have led to degraded rendering quality due to primitive constraints. However, works such as Adaptive Shell [34], Binary Occupancy Field [26], and Scaffold-GS [21] have demonstrated that incorporating geometry guidance significantly enhances rendering quality by producing well-regularized spatial structures with hybrid representations.

Building on these insights, we propose a synchronously optimized *dual-branch* system, addressing rendering and reconstruction with hybrid representations, to bypass the conflicts and further achieve mutual enhancements, as illustrated in Fig. 1. Our method leverages mutual guidance and joint supervision to balance rendering and reconstruction without compromising their intrinsic advantages. Specifically, our system features a 3D Gaussian Splatting (3DGS) branch for rendering and a Signed Distance Field (SDF) branch for surface reconstruction. The key innovations of our approach include: (1) Utilizing rasterized depth from the GS-branch to guide ray sampling in the SDF-branch, enhancing volume rendering efficiency and avoiding local minima. (2) Applying SDF-guidance for density control in 3DGS, directing the growth of 3D Gaussians in near-surface regions and pruning elsewhere. (3) Aligning geometry properties (depth and normal) estimated from both branches. This unified system overcomes the limitations inherent in each method due to differences in rendering techniques (rasterization vs. dense ray sampling) and scene representation (discrete primitives vs. continuous fields). Moreover, our framework is designed to accommodate future advancements in each branch.

Extensive experiments demonstrate that our dual-branch design allows: 1) The GS-branch to generate structured primitives closely aligned with the surface, reducing floaters and improving detail and edge quality in view synthesis. 2) Accelerated convergence in the SDF-branch, resulting in superior geometric accuracy and enhanced surface details. Our results confirm that an integrated blend of both reconstruction and rendering is achievable, enhancing overall robustness and performance.

2 Related work

Advanced Neural Rendering Techniques. Neural Radiance Fields (NeRFs) [23] have achieved remarkable photorealistic rendering with view-dependent effects. They use Multi-Layer Perceptrons (MLPs) to map 3D spatial locations to color and density, which are then aggregated into pixel colors through neural volumetric rendering. This approach excels in novel view synthesis but is slow due to the need for extensive point sampling along each ray and the global MLP architecture's scalability limitations. Recent research [24, 37, 4, 9] has shifted the learning burden to locally optimized spatial features, enhancing scalability. Alternative methods like rasterizing geometric primitives (e.g., point clouds) [2, 41] offer efficiency and flexibility but struggle with discontinuities and outliers. Augmenting points with neural features and incorporating volumetric rendering [38] improves quality but adds computational overhead. Recently, 3D Gaussian Splatting (3DGS) [17] revolutionized neural rendering by using anisotropic 3D Gaussians as primitives, sorted by depth and rasterized onto a 2D screen with α-blending. This method achieves high-quality, detailed results at real-time frame rates. Scaffold-GS [21] enhanced 3DGS by introducing a hierarchical structure that aligns anchors with scene geometry, improving rendering quality and memory efficiency. However, these methods prioritize view synthesis over accurate scene geometry, often resulting in fuzzy volumetric density fields that hinder the extraction of high-quality surfaces

Neural Surface Reconstruction. The success of neural rendering has sparked significant interest in neural surface reconstruction [25, 39, 31, 19, 32, 40]. These methods typically use coordinate-based networks to encode scene geometry through occupancy fields or Signed Distance Field (SDF) values.

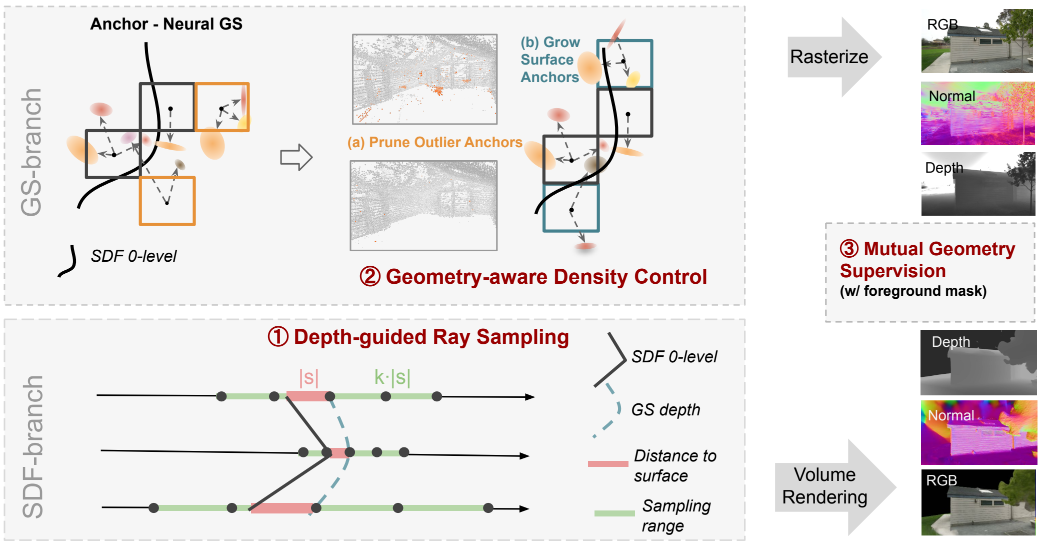

Figure 2: **Overview of Dual-branch Guidance**. Our dual-branch framework includes a GS-branch for rendering and an SDF-branch for learning neural surfaces. This design preserves the efficiency and fidelity of Gaussian primitive for rendering [17, 21] while accurately approximating scene surfaces from an SDF field adapted from NeuS [31]. Specifically: (1) The GS-branch renders depth maps to guide SDF-branch ray sampling, querying absolute SDF values $|s|$ and sampling points within $2k|s|$ (e.g., $k = 4$). (2) Predicted SDF values guide GS-branch density control, growing Gaussians near surfaces and pruning deviated ones. (3) Mutual geometry consistency is enforced by comparing depth and normal maps from both branches, ensuring coherent alignment between Gaussians and surfaces.

While MLPs with volume rendering produce smooth and complete surfaces, they often lack high-fidelity details and are slow to optimize. Recent works [19, 34, 26] have leveraged multi-resolution hashed feature grids from iNGP [24] to enhance representation power, achieving state-of-the-art results. Hybrid approaches combining surface and volume rendering [30, 34, 26] have also emerged to maintain rendering speed and quality. More recently, methods have explored integrating 3D Gaussian Splatting for surface learning. For example, SuGaR [10] aligns 3D Gaussians with potential surfaces by approximating them to 2D planar primitives with binary opacity. Similarly, 2D Gaussian Splatting [13] replaces 3D Gaussians with 2D Gaussians for more accurate ray-splat intersection and regularizes depth maps to enforce a more condensed Gaussian primitive distribution. Another approach [7] optimizes Gaussian surfels constrained by a normal prior from a pre-trained monocular estimator, enhancing surface representation. The concurrent work NeuSG [5] jointly optimizes 3D Gaussian Splatting with NeuS [31] by encouraging flat 3D Gaussians as well, with normals aligned with NeuS predictions. 3DGSR [22] also aligns the geometry from 3DGS with a SDF field. Despite advances in neural surface reconstruction, a fidelity gap persists due to the explicit regularization of Gaussian primitives, which hampers rendering quality compared to 3DGS. Our method optimizes primitives aligned with surfaces using geometric clues, enhancing structural integrity without losing representational power.

3 Method

We present a dual-branch framework with GS- and SDF-branches, jointly optimized to provide mutual guidance, as shown in Fig. 2. Our approach aligns Gaussian primitives with surfaces using geometric clues rather than constraining their shapes, enhancing both structure and representational power. These improved primitives yield more accurate and detailed scene surfaces.

In Sec.3.1, we briefly review neural rendering methods[17, 21] and neural implicit surface representations [31, 19]. Sec.3.2 details our framework and the three proposed mutual guidance techniques. Sec.3.3 outlines our training strategy and loss design.

3.1 Preliminary

3D Gaussian Splatting. 3D Gaussian Splatting (3DGS) [17] represents scenes using 3D Gaussian primitives, achieving state-of-the-art rendering quality and speed with a tile-based rasterizer. Each Gaussian is defined by its mean $\mu \in \mathbb{R}^3$ and covariance $\Sigma \in \mathbb{R}^{3\times 3}$ with $G(x) = e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$, where x is a 3D position in the scene. The rasterizer efficiently sorts and α-blends Gaussians onto the 2D image plane [43]. The adaptive control of Gaussians based on gradients is also critical for improving scene representation accuracy, reducing redundancy, and refining Gaussians to better match the underlying geometry and appearance.

Scaffold-GS [21] improves 3DGS by enhancing scene structure fidelity and robustness to view-dependent effects. It uses a hierarchical 3D Gaussian representation, with anchor points encoding local scene information and generating local neural Gaussians. Each anchor, optimized with a feature vector, predicts the color, center, variance, and opacity of the neural Gaussians.

Neural Implicit SDFs. NeuS [31] integrates signed distance functions (SDFs) with NeRF's volumetric rendering, converting SDF values into volume densities to maintain geometric accuracy. SDF values are converted to opacities using a logistic function $\alpha_i = \max\left(\frac{\Phi_s(f(\mathbf{x}_i))-\Phi_s(f(\mathbf{x}_{i+1}))}{\Phi_s(f(\mathbf{x}_i))}, 0\right)$. The color of a ray r is also calculated by accumulating weighted colors of the sample points: $\mathbf{C}(\mathbf{r}) = \sum_{i=1}^{P} T_i\alpha_i\mathbf{c}_i$, and $T_i = \exp\left(-\sum_{j=1}^{i-1}\alpha_j\delta_j\right)$, where δ_j is the interval between sampled points. Inspired by [24], [11] introduces a multi-resolution hash grid to enhance representation power and accelerate rendering and training.

3.2 GSDF: Dual-branch for Rendering and Reconstruction

To bridge the gap between rendering and geometric accuracy, we propose a novel approach that combines the strengths of Gaussian-based and SDF-based methods. By leveraging the high-quality rendering capabilities of 3DGS and the precise geometric representation of SDFs, our method aims to achieve superior results in both tasks.

As illustrated in Fig. 2, our dual-branch design integrates a GS-branch and an SDF-branch. We use Scaffold-GS [21] and NeuS [31] with adapted hash-encoding implementation [11] as the backbones, chosen for their effectiveness and simplicity. Importantly, our framework is versatile and can be easily adapted to incorporate future advanced methods for each branch.

3.2.1 GS $\rightarrow$ SDF: Depth Guided Ray Sampling

To address the computational expenses of ray-sampling, techniques such as hierarchical sampling [23], occupancy grids [20, 4, 24], early stopping [24], and proposal networks [3, 26] are widely used with CUDA accelerations. When depth maps are available, either from sensors or monocular estimation, samples can be strategically placed around surface regions, which is crucial for effective optimization of the Signed Distance Field (SDF). Unlike neural implicit SDF-based methods that rely on their own predicted SDF values for ray sampling [31, 19, 28], we employ the GS-branch to provide surface proximity, avoiding the chicken-and-egg dilemma. Inspired by [37], where a more efficient branch provides coarse geometry guidance, we leverage depth maps from the GS-branch to refine the ray sampling range for the SDF-branch. While Gaussian primitives may be less precise, they are efficient and flexible, offering sufficient geometric clues without significant overhead.

Concretely, for a ray emitted from a camera center $\vec{o}$ in the direction $\vec{v}$, the depth value D from the GS-branch is:

$$D = \sum_{i\in N} d_i\sigma_i \prod_{j=1}^{i-1}(1-\sigma_j), \tag{1}$$

where N is the number of 3D Gaussians encountered, σ_i is the opacity, and d_i is the distance from the i-th Gaussian to the camera. For SDF-branch optimization, points are sampled around $\vec{o} + D\cdot\vec{v}$. The sampling range adapts based on predicted SDF values s at various depths:

$$s = \mathcal{F}_s(\vec{o} + D\cdot\vec{v}), \tag{2}$$

where $\mathcal{F}_s$ is a two-layer MLP predicting the SDF value at a given position. The sampling range is defined as $r = [\vec{o} + (D - k|s|)\cdot\vec{v}, \vec{o} + (D + k|s|)\cdot\vec{v}]$. As shown in Fig. 2, inspired from NeRF's hierarchical sampling strategy [23], we use coarse and fine ranges with $k = 3$ and $k = 1$, respectively, we uniformly sample M points along the ray within each range.

3.2.2 SDF $\rightarrow$ GS: Geometry-aware Gaussian Density Control

Many previous and concurrent methods [13, 10, 7] have attempted to learn scene surfaces from 3DGS by flattening 3D Gaussians along the normal direction and enforcing nearly binary opacity, treating them like surface primitives similar to mesh triangles. However, this approach often leads to degraded rendering quality and incomplete surfaces. Our approach, instead of putting additional regularization on the 3D Gaussian primitives, enhances the distribution of Gaussian primitives using a geometry-aware density control strategy. Building on the original gradient-based density control, we leverage the zero-level set of the SDF-branch to determine the proximity of Gaussian primitives to the surface. By querying the SDF-branch with the positions of Gaussian primitives, we are able to identify close-to-surface Gaussian primitives (i.e. with smaller absolute SDF values), allowing for more precise control over the placement and density of Gaussians.

Growing Operator. For each Gaussian primitive at position c, we obtain its SDF value $s = \mathcal{F}_s(c)$ from the SDF-branch. The growth criteria for Gaussians are then defined as:

$$\epsilon_g = \nabla_g + \omega_g \mu(s), \tag{3}$$

where ∇_g is the averaged gradient of Gaussian primitives accumulated over K training iterations, as in Scaffold-GS [21]. $\mu(s) = \exp\left(-s^2/(2\sigma^2)\right)$ is a Gaussian function converting the SDF value to a positive factor, decreasing monotonically with distance from the zero level set. ω_g controls the influence of geometric guidance. New Gaussian primitives are added if ϵ_g are greater than a predefined τ_g.

Pruning Operator. Beyond the original opacity-based criteria, we prune Gaussian primitives far from the surface, indicated by large SDF values. The pruning criteria are:

$$\epsilon_p = \sigma_a - \omega_p(1 - \mu(s)), \tag{4}$$

where σ_a is the aggregated opacity over K iterations. The weight ω_p balances the contributions of transparency and SDF. Primitives with ϵ_p less than a predefined τ_p are pruned.

3.2.3 GS $\leftrightarrow$ SDF: Mutual Geometry Supervision

To enhance both rendering and reconstruction outcomes, our framework incorporates mutual supervision between the two branches, using depth and normal maps as key geometric features to facilitate this interconnection.

For the SDF-branch, a per-view depth map D_s is rendered using volumetric rendering principles, and a normal map N_s is derived by rendering the gradients of the SDF volumetrically. For the GS-branch, we compute a per-view depth map D_g following Eq. 1. The normal of each 3D Gaussian is determined by the direction of its smallest scaling factor, as in [10, 5, 6]. To render the normal map for each camera view, we accumulate the normals of the 3D Gaussians using α-blending $N_g = \sum_{i \in N} n_i \sigma_i \prod_{j=1}^{i-1}(1 - \sigma_j)$, where n_i is the estimated normal of the i-th 3D Gaussian.

This mutual geometry supervision ensures that the depth and normal maps from both branches are consistently aligned. This consistent alignment is crucial for maintaining coherence between the rendering and reconstruction processes, as it helps to prevent discrepancies that could lead to artifacts and inaccuracies when performing mutual guidance. This consistency is especially important for achieving high-quality results in complex scenes, where precise geometric alignment can significantly enhance the visual and structural integrity of the output.

3.3 Training Strategy and Loss Design

The GS-branch is supervised by rendering losses $\mathcal{L}_1$ and $\mathcal{L}_{\text{SSIM}}$, which measure the difference between the rendered RGB images and ground truth images, and a volume regularization term $\mathcal{L}_{vol}$ as in [21]. The complete loss function for GS-branch is:

$$\mathcal{L}_{\text{g}} = \lambda_1 \mathcal{L}_1 + (1 - \lambda_1)\mathcal{L}_{\text{SSIM}} + \lambda_{vol}\mathcal{L}_{vol}, \tag{5}$$

where λ_1 and λ_{vol} are weighting coefficients. Similarly, the SDF-branch is supervised by the $\mathcal{L}_1$ rendering loss, supplemented with Eikonal and curvature penalties to ensure accurate geometry reconstruction. The loss function for the SDF-branch is:

$$\mathcal{L}_{\text{s}} = \mathcal{L}_1 + \lambda_{\text{eik}}\mathcal{L}_{\text{eik}} + \lambda_{\text{curv}}\mathcal{L}_{\text{curv}}, \tag{6}$$

where $\mathcal{L}_{\text{eik}}$ represents the Eikonal loss, ensuring that the gradients of the predicted SDF field are normalized, and $\mathcal{L}_{\text{curv}}$ denotes the curvature loss, promoting surface smoothness as described in [19, 28]. The coefficients λ_{eik} and λ_{curv} balance the influence of these loss terms.

The mutual geometry supervision includes depth and normal consistency losses applied to both branches. For unbounded scenes with distant backgrounds, we only apply the mutual geometric supervision to foreground regions, as these geometries are more reliable and of primary interest [11]. The mutual loss is formulated as:

$$\mathcal{L}_{\text{mutual}} = \lambda_d\mathcal{L}_d + \lambda_n\mathcal{L}_n = \lambda_d\|D_{gs} - D_s\| + \lambda_n\left(1 - \frac{|N_{gs} \cdot N_s|}{\|N_{gs}\|\,\|N_s\|}\right), \tag{7}$$

where $\mathcal{L}_d$ and $\mathcal{L}_n$ represent the depth and normal discrepancies between the two branches, and λ_d and λ_n balance their importance. Finally, the total loss for joint learning is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{g}} + \mathcal{L}_{\text{s}} + \mathcal{L}_{\text{mutual}}. \tag{8}$$

The hyper-parameter settings are detailed in the supplementary material.

4 Experiment

4.1 Experimental Setup

Datasets. We evaluated results using 26 real-world and synthetic scenes from various datasets: 7 from Mip-NeRF360 [3], 2 from DeepBlending [12], 2 from Tanks&Temples [18], and 15 from DTU [14], featuring a wide range of indoor, outdoor, and object-centric scenarios.

Implementation Details. We implemented our dual-branch model based on 1) Scaffold-GS [21] and 2) an enhanced version of NeuS [31] with a hash-grid variant [11], following the practice of [19]. The hash grid resolution spans from 2^5 to 2^{11} with 16 levels, each entry having a feature dimension of 4 and a maximum of 2^{21} entries per level. The coarsest 4 layers were activated initially for the DTU [1], and 8 layers for other datasets, with finer levels added every $2k$ iterations. We trained the GS-branch for 15k iterations, followed by joint training of both branches for 30k iterations. The SDF-branch was warmed up for 2k iterations on the DTU and 5k on other datasets without depth-guided ray sampling. Note that our system is adaptable with various existing or future rendering and reconstruction models.

Evaluations. We evaluated our method against state-of-the-art rendering and reconstruction approaches. For rendering comparison, we compared with Scaffold-GS [21], 3D-GS [17], NeuS [31], 2D-GS [13], and SuGaR [10] using PSNR, SSIM [33], and LPIPS [42] for quantitative comparisons. We trained 3D-GS and Scaffold-GS for $45k$ iterations to align with our configurations. For reconstruction, we compared with NeuS [31], Instant-NSR [11] (our representative SDF-branch), SuGaR [10] and 2D-GS [13]. We use Chamfer distance for quantitative evaluation. For all datasets, we used $1/8$ of the images as test sets and the other $7/8$ as training sets. All experiments were conducted on a single NVIDIA A100 GPU with 80G memory.

4.2 Results Analysis

4.2.1 Rendering Comparisons

Our GSDF retained high rendering quality compared to state-of-the-art 3D-GS and SDF-based methods, as shown in Table 1. Quantitative metrics against 3D-GS [17], 2D-GS [13], and Scaffold-GS [21] show that GSDF consistently outperformed these methods across all four datasets, showcasing improvements in all metrics. Notably, the prominent improvements in the LPIPS metric indicate that our method effectively captures high-frequency scene details and renders perceptually superior results compared to baselines.

Table 1: **Rendering and reconstruction comparisons against baselines** over four benchmark scenes. 3D-GS [17], Scaffold-GS [21], 2D-GS [13], SuGaR [10] and GSDF initialized the Gaussian primitives with COLMAP [29] sparse points.

Dataset		DTU [1]				Tanks&Temples [18]			Mip-NeRF360 [3]			Deep Blending [12]		
Method & Metrics		reconstruction CD ↓	rendering PSNR ↑	SSIM ↑	LPIPS ↓	rendering PSNR ↑	SSIM ↑	LPIPS ↓	rendering PSNR ↑	SSIM ↑	LPIPS ↓	rendering PSNR ↑	SSIM ↑	LPIPS ↓
SDF-Based	NeuS [31]	0.823	31.93	0.912	0.165	-	-	-	-	-	-	-	-	-
	Instant-NSR [11]	1.809	26.34	0.846	0.230	19.36	0.641	0.495	21.87	0.612	0.509	24.00	0.795	0.462
GS-Based	3D-GS [17]	4.034	32.91	0.943	0.092	26.73	0.858	0.210	28.89	0.857	0.209	29.44	0.899	0.248
	Scaffold-GS [21]	5.988	33.05	0.946	0.100	27.29	0.869	0.182	29.34	0.863	0.200	30.37	0.908	0.238
	2D-GS [13]	0.928	33.39	0.947	0.105	25.75	0.840	0.246	28.08	0.843	0.240	29.27	0.897	0.261
	SuGaR [10]	1.239	32.76	0.942	0.094	24.68	0.827	0.230	27.40	0.817	0.260	29.53	0.895	0.265
Hybrid	**GSDF**	0.802	33.65	0.948	0.092	27.37	0.875	0.156	29.38	0.865	0.185	30.38	0.909	0.223

Table 2: **Rendering comparisons with random initialization against baselines** over three benchmark scenes. Scaffold-GS (rand) [21], 2D-GS (rand) [13] and GSDF (rand) initialized the Gaussian primitives with random points.

Dataset	Mip-NeRF360 [3]			Tanks&Temples [18]			Deep Blending [12]		
Method & Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
2D-GS (rand)	26.94	0.793	0.300	25.16	0.817	0.275	28.76	0.892	0.274
Scaffold-GS (rand)	27.84	0.817	0.259	26.37	0.838	0.230	29.12	0.895	0.260
GSDF (rand)	**28.05**	**0.830**	**0.229**	**26.55**	**0.852**	**0.197**	**29.57**	**0.899**	**0.244**

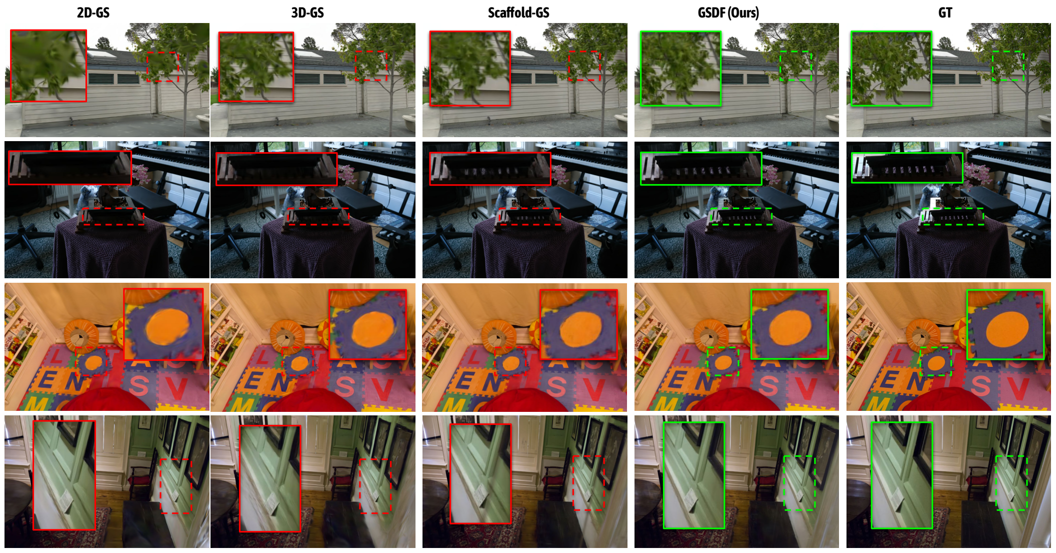

Figure 3: **Qualitative comparisons** of GSDF against popular Gaussian-based baselines [17, 21, 13] across diverse 3D scene datasets [18, 3, 12]. As highlighted, GSDF excels in modeling delicate geometries (1st & 2nd rows) and handling texture-less and sparsely observed regions (3rd & 4th rows), which are commonly presented in larger scenes where baseline approaches struggle to address.

Enchanced Detail and Fidelity. Notably, our method excelled in achieving *high rendering quality* in texture-less areas, as showcased in Fig.3, which is aligned with our design motivation for geometry-aware density control. Specifically, in texture-less areas where vanilla 3D Gaussians struggled due to small accumulated gradients, our method overcame this limitation by growing anchors in surface regions which brings enhanced accuracy and scene details.

Robustness with Noisy Gaussians. To test the robustness, we experimented with randomly initialized Gaussian primitives on 2D-GS, Scaffold-GS, and our method. Quantitative results in Table 2 highlight the advantages of our geometric guidance, demonstrating superior performance and stability even with random input. Further visual results are provided in the supplementary material.

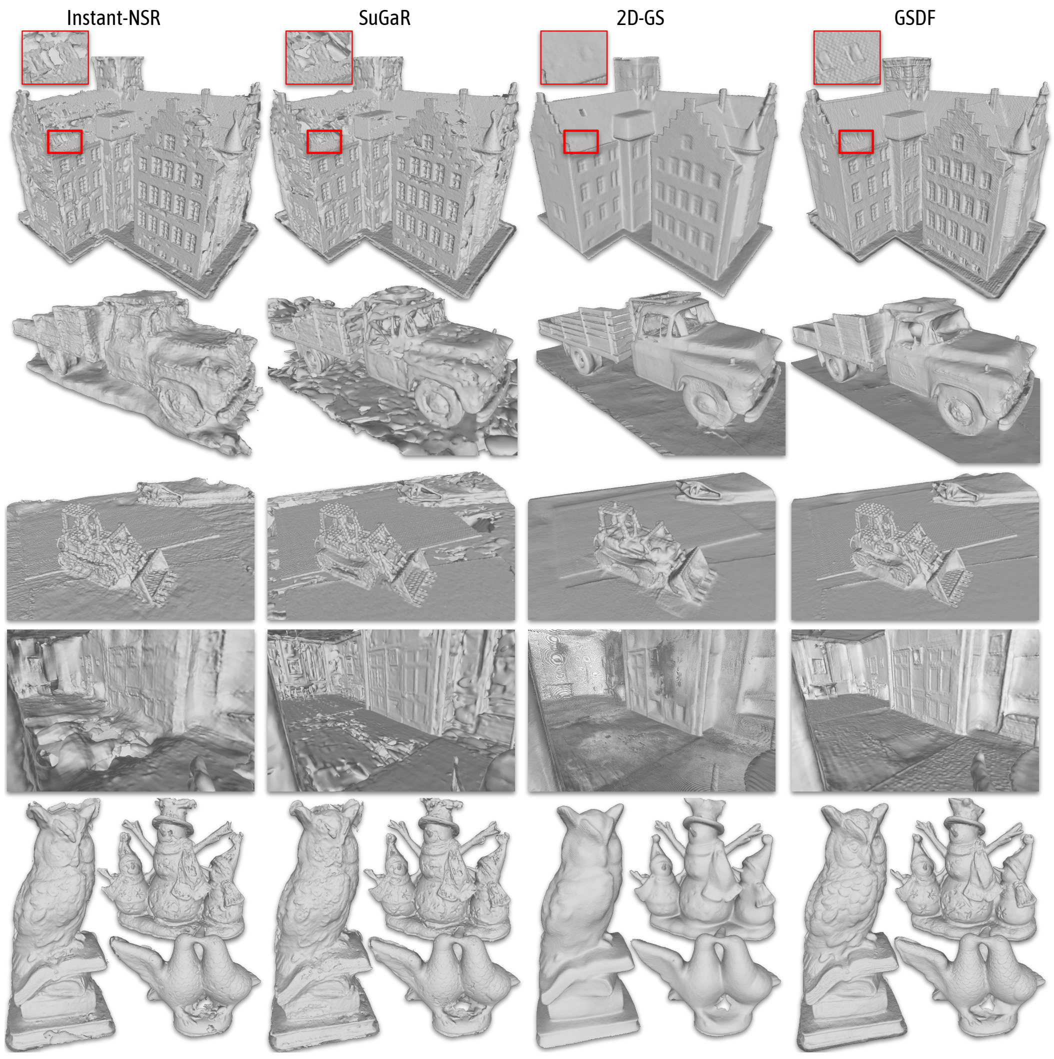

Figure 4: **Reconstruction Comparison**. We visualize the reconstructed meshes from Instant-NSR [11] (our SDF-branch), SuGaR [10], 2D-GS [13], and Ours.

4.2.2 Reconstruction Comparisons

We quantitatively evaluated the reconstruction ability on the DTU dataset, where our method achieved the best Chamfer distance as shown in Table 1.

Enhanced Geometry Accuracy and Completeness. As illustrated in Fig.4, our method reconstructed more complete and detailed meshes compared to baseline methods. Our approach effectively bypassed local minima, preventing holes in the meshes. Notably, our method outperformed Instant-NSR, delivering smooth, continuous meshes with well-preserved high-frequency details. In contrast, meshes extracted from SuGaR [10] were non-manifold with broken topological relationships. 2D-GS [13] extracts mesh from fused depth, which leads to over-smooth geometry.

Accerelated Convergence and Rendering. Benefited from the GS-branch, our method optimized the SDF field significantly faster in terms of training iterations than previous methods with accurate sample placements, which often required slow and exhaustive training and rendering. For instance, NeuS [31] required about 8 hours of optimization on a single GPU for the DTU dataset, whereas our method achieved comparable or better results within just 2 hours on a single GPU.

Table 3: **Quantitative Results on Ablation Studies.** We separately listed the rendering metrics for each ablation described in Sec. 4.3.

Dataset	Mip-NeRF360[3]			Tanks&Temples[18]			Deep Blending [12]		
Method | Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
GSDF (Full)	**29.38**	**0.865**	**0.185**	**27.37**	**0.875**	**0.156**	**30.38**	**0.909**	**0.223**
w/o geometric supervision	29.27	0.862	0.196	27.18	0.865	0.191	30.21	0.907	0.234
w/o depth-guided sampling	29.26	0.863	0.196	27.30	0.873	0.159	30.17	0.908	0.228
w/o geometry-aware densification	29.29	0.863	0.196	27.29	0.870	0.179	30.25	0.908	0.236

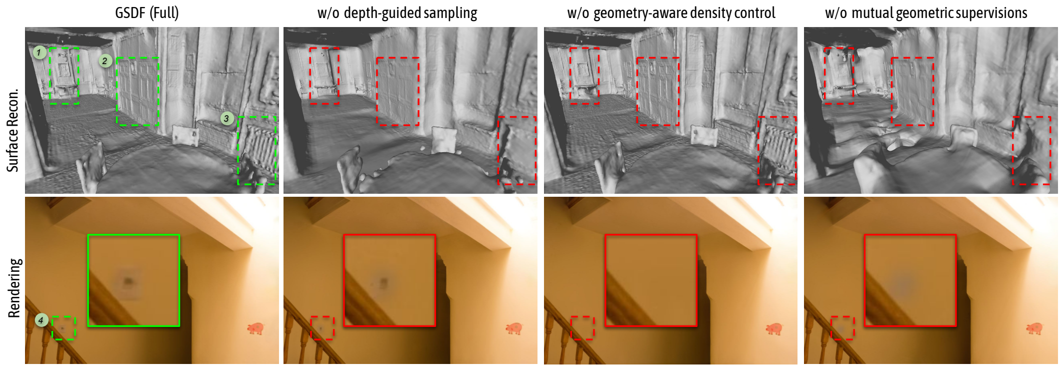
GSDF (Full)
w/o depth-guided sampling
w/o geometry-aware density control
w/o mutual geometric supervisions
Surface Recon.
Rendering
1
2
3
4

Figure 5: **Ablation results**. Visualizations of reconstructed meshes and rendered images from 1) our full method, 2) ours w/o depth-guided ray sampling, 3) ours w/o geometry-aware density control, and 4) ours w/o geometric supervision. We highlight the degradation of quality using numbered patches.

4.3 Ablation Studies

In this section, we examine the effectiveness of each individual module. Quantitative metrics and qualitative visualizations are provided in Table 3 and Fig. 5.

Depth-Guided Ray Sampling. To evaluate the effectiveness of our depth-guided ray sampling detailed in Sec 3.2, we conducted an ablation on depth-guided ray sampling using the original stratified ray sampling approach [23]. The results show that this ablated setting produced overly smoothed surfaces, failing to capture finer geometry details such as the heater and doors (Fig. 5, column 2, patches 2 & 3). The resulting less accurate SDF also impacted rendering quality, making details like the sticker on the wall appear more blurred compared to our full model.

Geometry-aware Gaussian Density Control. We replaced our geometry-aware Gaussian density control with the pruning and growing strategy from Scaffold-GS [21] to evaluate its efficacy. As shown in Fig. 5 (3rd column), this change led to missed details like the sticker on the wall (patch 4) due to small accumulated gradients on the texture-less surface. The reconstruction results also showed the absence of thin objects such as table legs and the chandelier (patch 1), highlighting the limitations of the vanilla strategy.

Mutual Geometric Supervision. Lastly, we ablated the proposed mutual geometric supervision by setting both λ_d and λ_n to 0 (Eq. 7). This resulted in a significant decay in surface quality and omission of details in rendering. These findings indicate that neural rendering is more tolerant of deviations than neural surface reconstruction. Without explicitly aligning Gaussians with SDF-derived geometry during optimization, the two branches can diverge, leading to sub-optimal results for both.

4.4 Limitations and Future Works

Currently, our method is not tailored to handle challenging scenes with reflections and intense lighting variations, such as those found in indoor environments. However, we have observed that employing more structured and surface-aligned Gaussian primitives holds promise in capturing such view-dependent appearance changes with improved scene geometry. Our framework requires more memory usage because it simultaneously considers two representations. Furthermore, the performance of our SDF-branch significantly lags behind the GS-branch, leading to extended training durations compared to Scaffold-GS [21] only. Hence, improving the efficiency of the MLP-based

SDF-branch is a crucial direction for future research. Still, training time is not the highest priority compared to inference, where primitives are stored and can be accessed efficiently.

5 Conclusion

In this work, we introduced a dual-branch framework that leverages the strengths of both 3D-GS and SDF, showcasing its potential to achieve both enhanced rendering and reconstruction quality. The inherent differences in two representations, rendering approaches, and supervision loss pose a challenge to the seamless integration. We therefore integrate a bidirectional mutual guidance approach during the training to circumvent these restrictions. Three types of guidance have been introduced and validated in our framework, namely: 1) depth guided sampling (GS$\rightarrow$SDF), 2) geometry-aware Gaussian density control (SDF$\rightarrow$GS); and 3) mutual geometry supervision (GS$\leftrightarrow$SDF). Our extensive results demonstrate the efficiency and joint performance improvement on both tasks. As the two branches maintain their original architectures, we keep their efficiency during inference, allowing room for potential enhancements by substituting each branch with more advanced models in the future. We envision our model benefiting applications demanding high-quality rendering and geometry, including embodied environments, physical simulation, and immersive VR experiences.

6 Acknowledgements

This work is funded in part by the National Key R&D Program of China (2022ZD0160201), and Shanghai Artificial Intelligence Laboratory.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, pages 1–16, 2016.

[2] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 696–712. Springer, 2020.

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. 2022.

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

[6] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. *arXiv preprint arXiv:*, 2024.

[7] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. 2024.

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

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

[10] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023.

[11] Yuan-Chen Guo. Instant neural surface reconstruction, 2022. https://github.com/bennyguo/instant-nsr-pl.

[12] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. 37(6):257:1–257:15, 2018.

[13] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. *ArXiv*, abs/2403.17888, 2024.

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

[15] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. *arXiv preprint arXiv:2401.16663*, 2024.

[16] Nikhil 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. *arXiv preprint arXiv:2312.02126*, 2023.

[17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

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

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

[20] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[21] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering, 2023.

[22] Xiaoyang Lyu, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi. 3dgsr: Implicit surface reconstruction with 3d gaussian splatting. *arXiv preprint arXiv:2404.00409*, 2024.

[23] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

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

[25] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599, 2021.

[26] Christian Reiser, Stephan Garbin, Pratul P Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan T Barron, Peter Hedman, and Andreas Geiger. Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. *arXiv preprint arXiv:2402.12377*, 2024.

[27] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3437–3444. IEEE, 2023.

[28] Radu Alexandru Rosu and Sven Behnke. Permutosdf: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8466–8475, 2023.

[29] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[30] Haithem Turki, Vasu Agrawal, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Deva Ramanan, Michael Zollhöfer, and Christian Richardt. Hybridnerf: Efficient neural rendering via adaptive volumetric surfaces. *arXiv preprint arXiv:2312.03160*, 2023.

[31] 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.

[32] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3272–3283, 2022.

[33] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[34] Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic. Adaptive shells for efficient neural radiance field rendering. *arXiv preprint arXiv:2311.10091*, 2023.

[35] Tianyi Xie, Zeshun Zong, Yuxin Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*, 2023.

[36] Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, et al. Vr-nerf: High-fidelity virtualized walkable spaces. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12, 2023.

[37] Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. 2023.

[38] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022.

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

[40] 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. *arXiv preprint arXiv:2302.14859*, 2023.

[41] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.

[42] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[43] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001.

A Supplementary Material

The following sections are organized as follows: 1) The first section elaborates on the implementation details of *GSDF*, covering hyper-parameters and curvature loss. 2) We then show additional experimental results. 3) Lastly, we also delve into limitations and future directions. More results and video demos can be found in supplementary webpage.

A.1 Implementation details

A.1.1 Configurations.

Below, we enumerate the hyper-parameters used in our experiments:

- The variance σ^2 for the Gaussian function in Eq. 3 and 4 is set to 0.005.
- For the rendering loss discussed in Sec. 3.3, we set $\lambda_1 = 0.2$ and $\lambda_{vol} = 0.01$, in consistency with the configurations specified in Scaffold-GS [21].
- For the SDF-branch, we set $\lambda_{eik} = 0.1$ and implement an adaptive scheme for λ_{curv}. Specifically, λ_{curv} increases linearly from 0 to 1 (0.5 on DTU) over the first 2000 iterations, after which it remains at 0.05 (0.01 on DTU) for subsequent iterations. This strategy is based on our observation that increasing the weight of the curvature loss significantly encourages the convergence of the SDF to a geometric outline.
- For the mutual geometry loss, we typically assign $\lambda_d = 0.5$ and $\lambda_n = 0.01$.

A.1.2 Curvature Loss.

While Instant-NSR [11] derives the curvature loss from a discrete Laplacian, we follow a more robust and explicit method as described in PermutoSDF [28]. For any given point, we randomly perturb it within the tangent plane orthogonal to its normal. The curvature loss is then measured by the cosine similarity of normals between the original point and its perturbed counterpart. Note that, we applied the same curvature loss for both Instant-NSR and GSDF.

A.2 More experiments

A.2.1 Reconstruction with 500k iterations.

As discussed in Sec. 4, optimizing the SDF is a time-intensive process. While our GS-branch assists in the convergence of the SDF-branch, we conjecture that further improvements could be made through additional iterations. Hence, we train our method for 500k iterations. As shown in Fig.6, our method achieves superior reconstruction quality compared to Instant-NSR [11] (our SDF-branch) with the same training iterations.

A.2.2 Rendering comparison with random initialization.

In Sec. 3.2, we discussed how the SDF-branch can improve the rendering quality of the GS-branch. This enhancement is particularly noticeable when Gaussian primitives are randomly initialized, as shown in Fig. 7 and Fig. 8.

A.2.3 Per-scene Results.

We list the per-scene quality metrics (PSNR, SSIM, LPIPS, and Chamfer Distance) used in our rendering and reconstruction evaluation in Sec. 4 for all considered methods, as shown from Tab. 4 to Tab. 6.

Table 4: Quantitative results for DTU scenes [1]. Note that we tried our best, but NeuS [31] failed to reconstruct geometries in scenes 69 and 105 with $\frac{7}{8}$ training images.

CD ↓	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
NeuS [31]	0.828	0.939	1.120	0.421	1.061	0.623	-	1.474	1.180	-	0.555	1.218	0.340	0.462	0.480
Instant-NSR [11]	1.296	1.107	1.284	6.582	1.459	0.710	2.361	1.535	1.664	0.897	0.736	2.580	0.552	2.803	1.565
3D-GS [17]	4.877	5.720	4.738	3.133	7.119	3.713	3.486	3.830	4.294	4.323	3.218	3.418	3.648	2.445	2.550
Scaffold-GS [21]	7.234	6.234	6.483	7.436	8.173	4.266	5.779	5.451	6.565	6.358	5.049	5.953	6.317	5.615	2.899
2D-GS [13]	0.662	1.108	0.707	0.430	1.307	1.095	0.901	1.385	1.257	0.823	0.754	1.419	0.484	0.974	0.606
SuGaR [10]	1.166	1.075	0.958	0.471	2.187	1.638	1.022	1.812	1.505	0.930	0.672	2.686	0.701	0.943	0.819
GSDF	0.588	0.936	0.460	0.376	1.300	0.771	0.734	1.589	1.287	0.760	0.588	1.222	0.379	0.518	0.513

PSNR ↑	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
NeuS [31]	26.96	26.08	26.86	27.89	33.76	32.90	-	38.13	28.95	-	33.92	32.09	29.62	35.77	37.19
Instant-NSR [11]	21.64	21.24	23.27	19.10	30.40	27.25	25.07	32.87	24.32	29.71	28.90	27.39	24.96	29.57	29.44
3D-GS [17]	29.98	27.33	28.87	31.64	35.68	32.76	28.75	38.05	31.78	36.63	34.66	33.08	29.94	36.83	37.78
Scaffold-GS [21]	30.59	27.85	29.65	32.33	29.27	33.22	29.43	38.96	32.04	36.60	36.38	34.05	31.07	36.67	37.73
2D-GS [13]	30.31	27.90	29.02	32.79	35.52	33.32	29.58	37.01	32.32	36.50	36.25	33.47	30.87	37.30	38.76
SuGaR [10]	28.41	26.94	28.10	32.12	34.72	32.82	28.82	38.10	31.01	35.95	36.21	32.22	30.31	37.12	38.66
GSDF	30.16	27.65	28.92	32.90	35.46	33.73	29.46	39.32	32.28	36.81	36.47	34.00	30.99	37.57	39.17

SSIM ↑	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
NeuS [31]	0.780	0.816	0.749	0.884	0.950	0.962	-	0.972	0.922	-	0.924	0.939	0.909	0.953	0.964
Instant-NSR [11]	0.689	0.754	0.670	0.725	0.938	0.928	0.861	0.957	0.857	0.898	0.871	0.912	0.833	0.897	0.913
3D-GS [17]	0.915	0.897	0.887	0.951	0.964	0.959	0.933	0.976	0.943	0.957	0.952	0.952	0.934	0.962	0.971
Scaffold-GS [21]	0.914	0.906	0.902	0.956	0.948	0.962	0.936	0.976	0.945	0.957	0.957	0.958	0.938	0.965	0.972
2D-GS [13]	0.920	0.911	0.903	0.958	0.964	0.962	0.938	0.970	0.947	0.954	0.957	0.954	0.939	0.964	0.973
SuGaR [10]	0.901	0.903	0.889	0.955	0.956	0.955	0.932	0.974	0.939	0.952	0.958	0.947	0.935	0.965	0.973
GSDF	0.916	0.905	0.902	0.959	0.965	0.963	0.938	0.978	0.947	0.957	0.959	0.960	0.939	0.967	0.975

LPIPS ↓	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
NeuS [31]	0.780	0.816	0.749	0.884	0.950	0.962	-	0.972	0.922	-	0.924	0.939	0.909	0.953	0.964
Instant-NSR [11]	0.689	0.754	0.670	0.725	0.938	0.928	0.861	0.957	0.857	0.898	0.871	0.912	0.833	0.897	0.913
3D-GS [17]	0.915	0.897	0.887	0.951	0.964	0.959	0.933	0.976	0.943	0.957	0.952	0.952	0.934	0.962	0.971
Scaffold-GS [21]	0.914	0.906	0.902	0.956	0.948	0.962	0.936	0.976	0.945	0.957	0.957	0.958	0.938	0.965	0.972
2D-GS [13]	0.920	0.911	0.903	0.958	0.964	0.962	0.938	0.970	0.947	0.954	0.957	0.954	0.939	0.964	0.973
SuGaR [10]	0.901	0.903	0.889	0.955	0.956	0.955	0.932	0.974	0.939	0.952	0.958	0.947	0.935	0.965	0.973
GSDF	0.916	0.905	0.902	0.959	0.965	0.963	0.938	0.978	0.947	0.957	0.959	0.960	0.939	0.967	0.975

Table 5: Quantitative results for Mip-NeRF 360 scenes [3].

Outdoor	bicycle			garden			stump		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Instant-NSR [11]	19.68	0.449	0.615	19.74	0.422	0.627	18.77	0.486	0.592
3D-GS [17]	24.46	0.707	0.313	26.63	0.819	0.175	26.31	0.758	0.309
Scaffold-GS [21]	24.63	0.721	0.289	26.59	0.814	0.182	26.58	0.765	0.302
Scaffold-GS(rand)	23.48	0.633	0.397	26.09	0.776	0.239	22.88	0.652	0.428
2D-GS [13]	23.96	0.683	0.358	26.07	0.803	0.211	25.69	0.746	0.349
2D-GS(rand)	22.65	0.578	0.455	24.98	0.734	0.298	22.51	0.637	0.471
SuGaR [10]	23.26	0.630	0.356	25.44	0.763	0.239	25.12	0.705	0.324
GSDF	24.61	0.724	0.261	26.91	0.824	0.158	26.04	0.757	0.290
GSDF (rand)	24.09	0.682	0.328	26.25	0.798	0.196	21.92	0.625	0.416

Indoor	room			counter			kitchen			bonsai		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Instant-NSR	23.18	0.769	0.439	23.00	0.723	0.459	23.87	0.685	0.400	24.84	0.753	0.430
3D-GS [17]	31.68	0.925	0.194	29.25	0.914	0.180	31.69	0.933	0.113	32.19	0.945	0.176
Scaffold-GS [21]	32.45	0.933	0.178	29.93	0.920	0.173	31.99	0.934	0.112	33.24	0.951	0.165
Scaffold-GS(rand)	31.84	0.920	0.205	29.00	0.898	0.206	30.50	0.914	0.144	31.08	0.928	0.196
2D-GS [13]	30.77	0.913	0.219	28.19	0.898	0.212	30.50	0.922	0.133	31.39	0.934	0.201
2D-GS(rand)	30.01	0.895	0.256	27.78	0.874	0.248	30.10	0.914	0.147	30.56	0.922	0.222
SuGaR [10]	30.08	0.904	0.259	27.55	0.884	0.244	29.51	0.901	0.179	30.81	0.933	0.219
GSDF	32.46	0.937	0.165	30.11	0.923	0.159	31.93	0.936	0.106	33.60	0.954	0.155
GSDF (rand)	32.01	0.929	0.181	29.17	0.909	0.183	31.45	0.929	0.118	31.43	0.936	0.182

Table 6: Quantitative results for Tanks&Temples [18] and Deep Blending [12] scenes.

Dataset	Tanks&Temple [18]						Deep Blending [12]					
	Barn			Truck			Dr Johnson			Playroom		
Method & Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Instant-NSR [11]	20.86	0.660	0.475	17.87	0.622	0.514	24.59	0.791	0.476	23.42	0.799	0.448
3D-GS [17]	28.21	0.855	0.223	25.24	0.861	0.197	29.04	0.897	0.248	29.84	0.900	0.248
Scaffold-GS [21]	28.77	0.869	0.192	25.80	0.869	0.172	29.86	0.908	0.236	30.89	0.908	0.239
Scaffold-GS (rand)	27.82	0.832	0.243	24.93	0.845	0.217	28.58	0.892	0.262	29.66	0.898	0.258
2D-GS [13]	26.74	0.828	0.267	24.75	0.852	0.224	28.65	0.895	0.262	29.88	0.899	0.260
2D-GS (rand)	26.48	0.810	0.288	23.84	0.824	0.262	28.13	0.889	0.276	29.40	0.895	0.273
SuGaR [10]	26.67	0.837	0.233	22.69	0.817	0.227	28.77	0.889	0.271	30.30	0.900	0.258
GSDF	28.93	0.877	0.176	25.81	0.873	0.135	29.87	0.909	0.225	30.89	0.909	0.222
GSDF (rand)	28.11	0.851	0.210	24.99	0.853	0.184	29.09	0.898	0.247	30.04	0.899	0.241

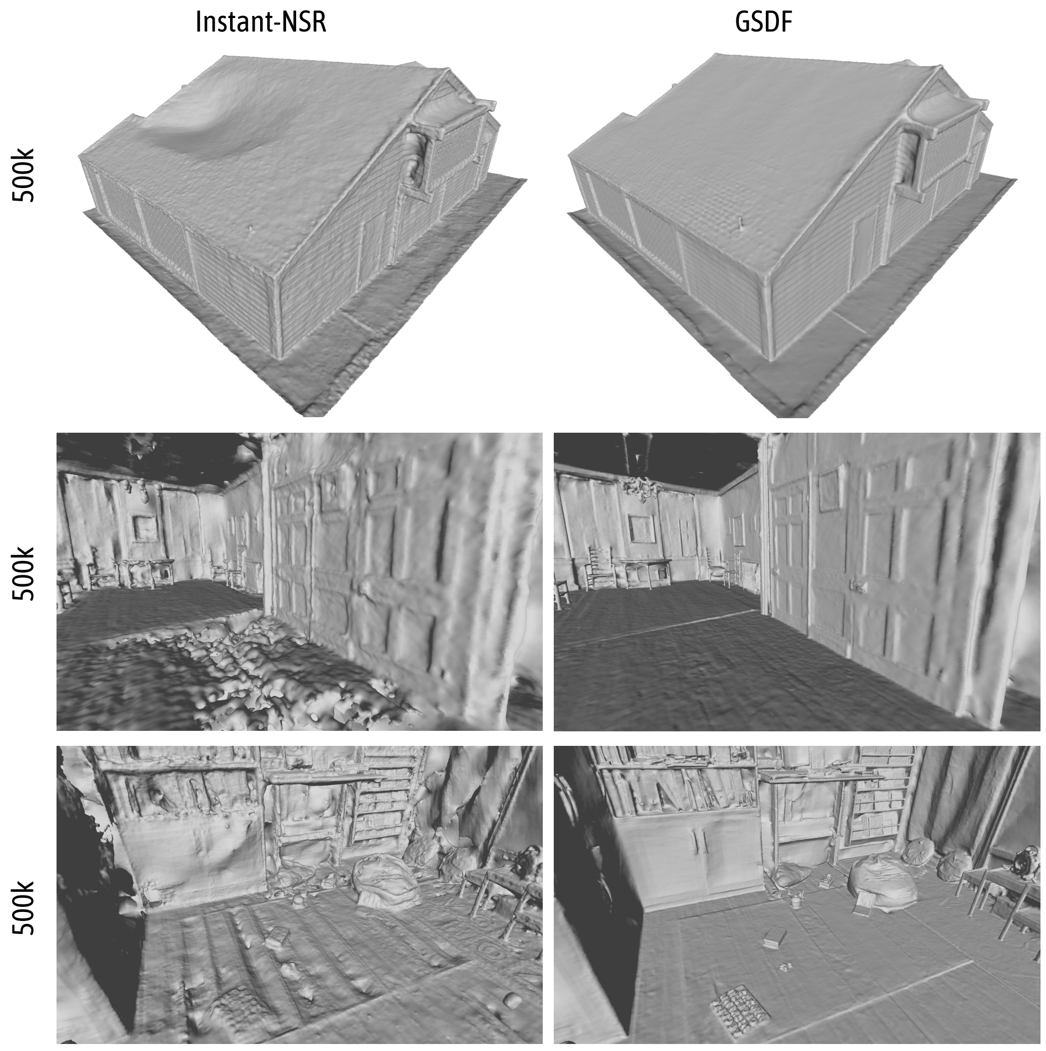

Figure 6: **Reconstruction comparison with** 500**k training iterations**.

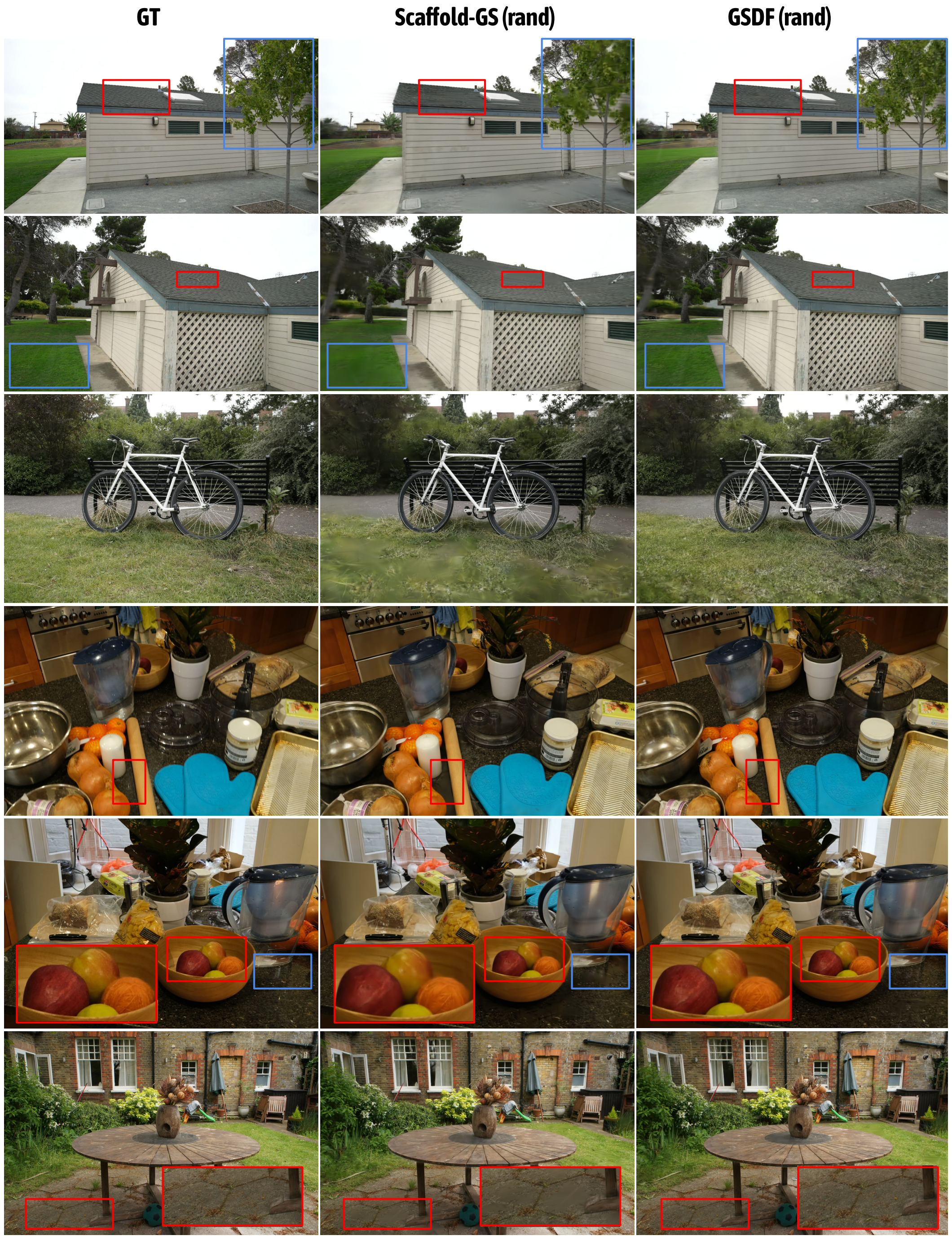

Figure 7: **Rendering Comparison with random initialization (Part 1)**. We compare rendering results between Scaffold-GS [21] and our method. The highlighted patches indicate that our method is superior in expressing finer details in both geometry and appearance.

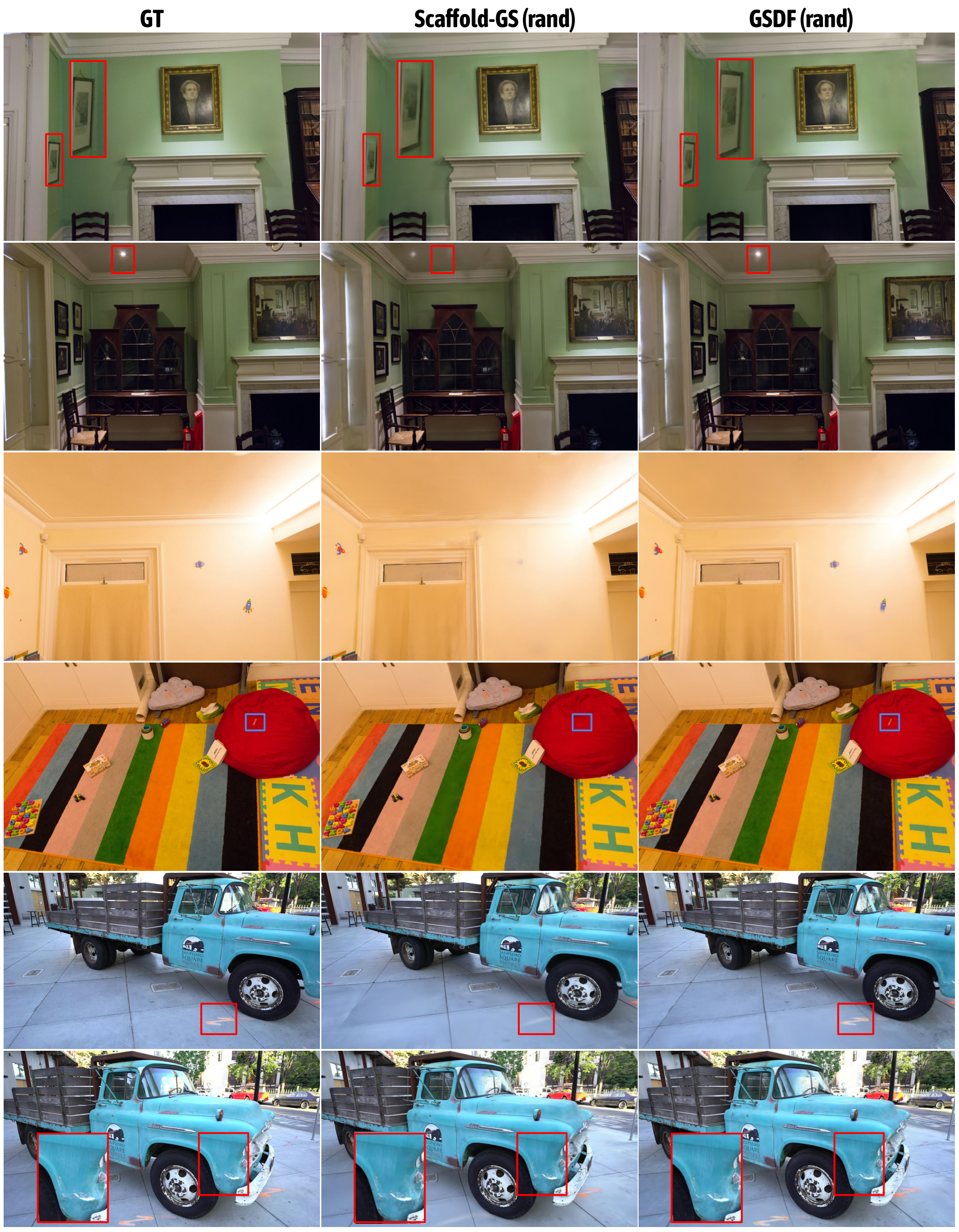

Figure 8: **Rendering Comparison with random initialization (Part 2) - Cont**.

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: We introduce GSDF, a dual-branch structure combining the strengths of 3D-GS and SDF, for improving both rendering and reconstruction. Please see our abstract and introduction for further details.

   Guidelines:

   - The answer NA 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 NA 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: Our framework still struggles with the dilemma like most SDF-based reconstruction methods. Please see the limitation section (Sec. 4.4) for further details.

   Guidelines:

   - The answer NA 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: In the method and experiment sections, we provide a detailed discussion and substantial evidence to demonstrate each result.

Guidelines:

- The answer NA 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: GSDF is a robust framework that comprises a GS-branch dedicated to rendering and an SDF-branch focusing on learning neural surfaces. By leveraging the mutual guidance between these two branches, our method achieves state-of-the-art performance on both rendering and reconstruction tasks.

Guidelines:

- The answer NA 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: [No]

   Justification: We will release the code after acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In the Experimental Setup section (Sec. 4.1) and Implementation details section (Sec. A.1), we provide comprehensive details, including the data splits and hyperparameters used in our experiments.

   Guidelines:

   - The answer NA 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: The experimental result for redering and reconstruction tasks from dense views are very stable, the standard deviation is negligible, so we follow the previous work's practice and not report the error bar.

   Guidelines:

   - The answer NA 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: In Experiment section (Sec. 4), We provide sufficient information on the computer resources, including the type of compute workers GPU and training time.

   Guidelines:

   - The answer NA 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: We have carefully reviewed the NeurIPS Code of Ethics and are committed to strictly adhering to its guidelines.

   Guidelines:

   - The answer NA 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: [Yes]

    Justification: Neural rendering and surface reconstruction are crucial tasks with important downstream applications in areas like robotics, physical simulations, and augmented reality. We believe our GSDF system has the potential to advance progress in this domain and support the development of these applications.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: There is no such risks in the rendering and reconstruction tasks.

    Guidelines:

    - The answer NA 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: In the Experimental Setup section (Sec. 4.1), we cite each used code package and public datasets.

    Guidelines:

    - The answer NA 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: [No]

    Justification: We plan to release the code upon acceptance of the paper.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: This paper does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: This paper does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA 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.