G2SDF: Surface Reconstruction from Explicit Gaussians with Implicit SDFs

Kunyi Li¹ * Michael Niemeyer² Zeyu Chen³ Nassir Navab^{1,4} Federico Tombari^{1,2}

¹Technical University of Munich ²Google ³Tsinghua University ⁴Johns Hopkins University

Abstract

State-of-the-art novel view synthesis methods such as 3D Gaussian Splatting (3DGS) achieve remarkable visual quality. While 3DGS and its variants can be rendered efficiently using rasterization, many tasks require access to the underlying 3D surface, which remains challenging to extract due to the sparse and explicit nature of this representation. In this paper, we introduce G2SDF, a novel approach that addresses this limitation by integrating a neural implicit Signed Distance Field (SDF) into the Gaussian Splatting framework. Our method links the opacity values of Gaussians with their distances to the surface, ensuring a closer alignment of Gaussians with the scene surface. To extend this approach to unbounded scenes at varying scales, we propose a normalization function that maps any range to a fixed interval. To further enhance reconstruction quality, we leverage an off-the-shelf depth estimator as pseudo ground truth during Gaussian Splatting optimization. By establishing a differentiable connection between the explicit Gaussians and the implicit SDF, our approach enables high-quality surface reconstruction and rendering. Experimental results on several real-world datasets demonstrate that G2SDF achieves superior reconstruction quality than prior works while maintaining the efficiency of 3DGS.

1. Introduction

3D Gaussian Splatting (3DGS) [3] has emerged as a state-of-the-art method for high-quality novel view synthesis by leveraging the rasterization pipeline and representing 3D scenes using points characterized by Gaussian functions. However, 3DGS remains an explicit and discrete representation, posing challenges for accurate 3D surface reconstruction. This limitation is in particular important for applications such as geometry editing [4–6], 3D animation [7–9], and robotics [10–12]. The primary challenge lies in

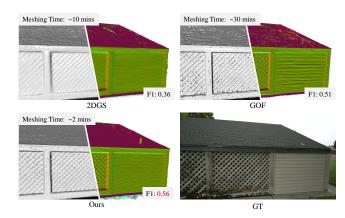


Figure 1. **G2SDF.** Our method establishes a connection between the implicit SDFs and explicit Gaussians, linking the opacity of each Gaussian to its distance from the surface. Designed to handle unbounded scenes, our approach achieves superior surface reconstruction, particularly in areas with limited view coverage. Compared to 2DGS [1] and GOF [2], our method achieves higher F1 scores (↑) and reconstructs smooth surfaces with fine details.

3DGS being a discrete and unstructured point-based geometry representation. Given that the optimization process is only guided by the rendering quality, the Gaussian representation tends to exhibit noisy geometry of the scene which complicates the extraction of accurate 3D surfaces through post-processing methods like Poisson surface reconstruction [13].

These challenges have motivated recent investigations to explore how 3DGS can be used for high-quality surface reconstruction while maintaining its rendering speed and efficiency. 2DGS [1] employs 2D instead of 3D Gaussians, forcing Gaussians to align closer to surface. They extract meshes through Truncated Signed Distance Field (TSDF) fusion [14]. However, TSDF fusion struggles with accurately modeling thin structures and reconstructing unbounded scenes, leading to rendered depth maps which can be inaccurate. SuGaR [15] addresses these issues by coarse-to-fine reconstruction. It regulates 3D Gaussians to align with coarse mesh faces, then uses Poisson surface recon-

^{*}kunyi.li@tum.de

struction [13] in second step to extract a mesh, although Poisson reconstruction ignores the Gaussians' opacities and scales. Overall, the primary challenge remains in the inconsistency between mesh extraction and Gaussian rasterization during training. Gaussian Opacity Fields (GOF) [2] attempts to address this by generating an opacity field from Gaussians and extracting surfaces using Marching Tetrahedra [16]. Nevertheless, the opacity field is created via post-processing through ray tracing, which requires a specific Gaussian rasterizer. This approach demands extra time and resources, and still suffers from inconsistency between the underlying geometry and Gaussians.

To achieve high-quality reconstructions while preserving the efficency of 3DGS, we introduce G2SDF, a novel approach that seamlessly integrates an implicit SDF into the Gaussian Splatting optimization. Our **contributions** are:

- A hash-based SDF which provides spatial continuity to the originally unstructured Gaussians without additional volume rendering training. Rather than solely optimizing Gaussian positions through appearance supervision, we use the SDF to effectively constrain their spatial distribution, creating a more cohesive 3D representation.
- A differentiable SDF-to-Opacity function that links the implicit SDF with Gaussian opacities, ensuring Gaussians closer to the surface exhibit higher opacity values and enforcing the distribution of Gaussians to be more surface aligned.
- A novel coordinate normalization function for our hybrid representation that maps unbounded 3D coordinates to a 0-1 range, enabling efficient positional embedding.

We find that the joint hybrid optimization of an implicit SDF and the explicit Gaussian Splatting representation leads to better reconstructions while maintaining 3DGS's efficiency. Experiments on real-world unbounded datasets demonstrate that our method delivers high-quality reconstructions while preserving the view synthesis quality. Note that our method is also easily adaptable to any existing Gaussian rasterizer and can be used with different methods for mesh extraction such as Marching Cubes [17], Marching Tetrahedra [16] and TSDF fusion [14].

2. Related Works

2.1. View Synthesis and Gaussian Splatting

Neural Radiance Fields (NeRF) [18] utilize multi-layer perceptions (MLP) as scene representation to predict geometry and view-dependent appearance. The MLP is optimized via a photometric loss through volume rendering. Subsequent methods have focused on optimizing NeRF's training and expressiveness using grid representations [19], improving rendering speed [20, 21] and scaling to unbounded scenes [22]. However, volume rendering typically requires substantial computational resources and extensive training

durations. Recently, 3D Gaussian Splatting (3DGS) [3] has emerged as an efficient alternative for view synthesis. By leveraging the differentiability of Gaussian functions, 3DGS achieves high-quality rendering while enabling real-time and efficient novel view image synthesis. Subsequent works have enhanced rendering quality through antialiasing techniques [23–25] or improved rendering speed by either controlling the density of Gaussians or using radiance field as a prior [26–28]. DNGaussian [29] introduces a depth-regularized approach to address the challenge of geometry degradation in sparse input views. GeoGaussian [30] focuses on preserving the geometry of non-textured regions like walls and furniture, which tend to degrade over time. Instantsplat [31] uses Dust3r [32] to initialize the point cloud and achieves fast training on sparse view case. Instead of representing scenes with explicit Gaussians, Scaffold-GS [33] utilizes a combination of MLPs and feature grids to infer Gaussians for rasterization, merging implicit and explicit representations with a focus on view synthesis. However, these methods primarily focus on enhancing appearance quality while neglecting underlying geometry, limiting their application to view synthesis only.

2.2. Surface Reconstruction from Gaussians

Due to the discrete and unstructured nature of 3DGS, along with supervision being limited to RGB images during training, previous methods often struggle to accurately capture scene geometry, making surface reconstruction a challenging task. SuGaR [15] addresses this by building a density field from Gaussians aggregating the opacity values of nearby Gaussians for any given 3D point. It extracts meshes from this density field using level-set searching and employs a two-stage training process where meshes from the coarse stage are used to guide the alignment of Gaussians. However, SuGaR is time-consuming and struggles with reconstructing large, smooth areas such as floors or walls. Other methods [34–38] employ either depth or normal estimators as priors to supervise Gaussian optimization. They then use TSDF fusion or Poisson reconstruction to extract meshes from multi-view depth maps or Gaussians. However, TSDF fusion faces challenges in handling unbounded scenes, whereas Poisson reconstruction ignores all Gaussian attributes. 2DGS [1] partly addresses this by flattening 3D Gaussians into 2D disks which better align with the underlying surfaces, improving geometric accuracy. Gaussian Opacity Fields (GOF) [2] constructs an opacity field by approximating the minimum accumulated alpha values from all views using ray tracing and employs a depth distortion loss from 2DGS [1] to improve smooth surface reconstruction. Surface extraction is done using Marching Tetrahedra [16], but the distribution of Gaussians relative to scene surfaces remains inconsistent and unstructured. 3DGSR [39] improves upon this by linking each Gaussian's opac-

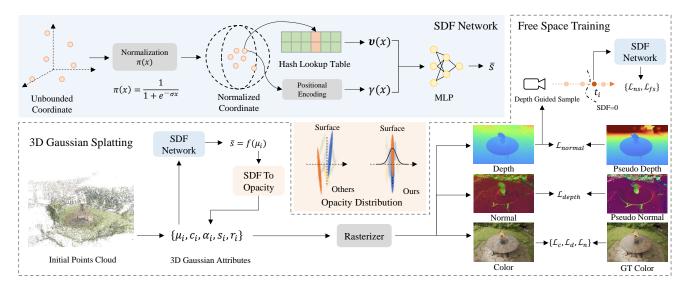


Figure 2. **Overview.** Our hybrid representation consists of the Gaussian Splatting pipeline and an SDF network, integrating implicit SDFs into the explicit Gaussian representation. Our SDF network first normalizes any 3D input to (0,1) range and predicts an SDF value. Then using a differentiable SDF-to-Opacity function, we connect the SDF value \bar{s} with the opacity value α and force Gaussians to follow a certain distribution near the surface. Rendered depth is used as a pseudo-ground truth signal to supervise the SDF directly in 3D. We further apply an off-the-shelf depth estimator to supervise rendered depth and normal.

ity value with its distance to the surface through a hash-grid based SDF. However, to effectively train the SDF, it incorporates an additional branch to volume render additional depth and normal maps supervised by depth and normal from Gaussian rasterization, which is in-efficient and the inconsistent rendering may degrade both branches. Our hybrid representation integrates a hash-grid into the 3DGS representation and is trained jointly, preserving both the speed and quality of training and rendering, while enabling detailed reconstruction.

3. Method

3.1. 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [3] employs a set of 3D points to effectively render images from given viewpoints, each characterized by a Gaussian function with 3D mean $\mu_i \in \mathbb{R}^3$, covariance matrix $\Sigma_i \in \mathbb{R}^{3 \times 3}$, opacity value $\alpha_i \in \mathbb{R}$, RGB color values $\mathbf{c}_i \in \mathbb{R}^3$:

$$o_i(\mathbf{x}) = \alpha_i * \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1}(\mathbf{x} - \mu_i)\right).$$
 (1)

Given a 3D position \mathbf{x} , $o_i(\mathbf{x})$ represents current opacity value contributed by the *i*-th Gaussian. To facilitate optimization, Σ_i is factorized into the product of a scaling matrix S_i , represented by scale factors $\mathbf{s}_i \in \mathbb{R}^3$, and a rotation matrix R_i encoded by a quaternion $\mathbf{r}_i \in \mathbb{R}^4$:

$$\Sigma_i = R_i S_i S_i^T R_i^T. \tag{2}$$

3D Gaussians are then projected onto a 2D image plane according to elliptical weighted average (EWA) [40] to render images for given views. Color $\bar{\mathbf{C}}(\mathbf{u})$, depth $\bar{D}(\mathbf{u})$, and normal $\bar{\mathbf{N}}(\mathbf{u})$ at pixel \mathbf{u} is rendered by N projected and ordered Gaussians using point-based α -blending:

$$\{\bar{\mathbf{C}}, \bar{D}, \bar{\mathbf{N}}\}(\mathbf{u}) = \sum_{i \in N} T_i o_i \{\mathbf{c}_i, d_i, \mathbf{n}_i\},$$
 (3)

where $T_i = \prod_{j=1}^{i-1} (1 - o_j)$, depth d_i is the distance between camera center and the ray-Gaussian intersection plane, and Gaussian's normal \mathbf{n}_i is approximated the as the normal of the ray-Gaussian intersection plane.

3.2. Neural Signed Distance Field

In our hybrid representation, we optimize a neural signed distance field (SDF) next to the 3DGS representation. To scale to unbounded scenes and achieve high fidelity, we combine multi-layer perceptrons (MLPs) with grid-based backbones [41] as outlined in the following.

Unbounded Normalization. Grid-based representations require defining a specific scene bounding box to initialize the grid's volume size. To adapt to unbounded real-world scenes at various scales, we first apply a sigmoid-like normalization function on any 3D point $\mathbf{x} \in \mathbb{R}^3$, mapping any scope to a fixed (0,1) range:

$$h(\mathbf{x}) = 1/(1 + exp(-\sigma \mathbf{x})),\tag{4}$$

where $\sigma=2/B, B\in\mathbb{R}^3$ is the bounding box size of initial Gaussians. This term ensures that the volume containing

the majority of Gaussians occupies most of the grid. During Gaussian densification, some new Gaussians may extend beyond this primary bounding box but remain within the defined scope (0,1) after normalization.

Parameterization. To better encode the scene geometry, we choose One-blob $\gamma(\mathbf{x})$ [42] as positional embedding and a multi-resolution hash-based feature grid $\mathcal{V} = \{\mathcal{V}^l\}_{l=1}^L$ [19]. Feature $\mathcal{V}(h(\mathbf{x}))$ at any 3D point \mathbf{x} are queried via trilinear interpolation. The geometry decoder f is a MLP which maps the 3D coordinate to a SDF value \bar{s} :

$$f(\mathbf{x}) = f(\gamma(h(\mathbf{x})), \mathcal{V}(h(\mathbf{x}))) \mapsto \bar{s}.$$
 (5)

Therefore, for each 3D Gaussian located at μ_i , its corresponding SDF value can be written as $\bar{s}_i = f(\mu_i)$.

3.3. Scene Representation and Optimization

While 3DGS is an explicit and discrete representation, SDFs provide an implicit representation that facilitates precise surface extraction. In this section, we discuss how to leverage the advantages of both and optimize them jointly. **Hybrid Scene Representation.** We propose a hybrid scene representation that combines the strengths of neural implicit SDF with 3DGS, as shown in Fig. 2. First, an SDF value is predicted for each Gaussian using Eq. 5, and it is then converted to opacity value which is needed by the Gaussian rasterizer. We use a differentiable Gaussian-shaped transformation $g(\bar{s})$:

$$g(\bar{s}_i) = \exp(-\beta * \bar{s}_i^2) \mapsto \alpha_i,$$
 (6)

where β is a learnable scale factor. This transformation has two main properties that ensures Gaussians to follow a structured distribution. First, it establishes a connection between the opacity and the position of a Gaussian Second, it enforces Gaussians to remain close to the surface, as their opacity values drop to zero with increasing distance.

Joint Optimization. To train such a hybrid model, a straightforward approach is to enforce all Gaussians to meet the condition $\bar{s}_i = 0$, resulting in $\alpha_i = 1$. But experimentally we found that this negatively impacts rendering quality as achieving high-quality renderings requires variation in Gaussian opacity and providing SDF supervision solely at the position of Gaussians is too sparse. On the other hand, supervising the SDF through volume rendering like 3DGSR [39] introduces substantial computational overhead, with supervision signals limited to 2D, which is insufficient for accurate SDF representation.

To address this issue, we back-project the rendered Gaussian depth \bar{D} into 3D to generate dense supervision signals. Further, we also supervise free space to mitigate unexpected noise. More specifically, we first sample M pixels $\{\mathbf{u}=\mathbf{u}_i\mid \mathbf{u}_i\in\mathbb{R}^2, \text{for }i=1,2,...,M\}$ from the rendered Gaussian depth map \bar{D} . The ray cast from a pixel

u is denoted as $\mathbf{p}(\mathbf{u}) = \mathbf{r}_o + t \cdot \mathbf{r}_d(\mathbf{u})$, where $\mathbf{r}_o \in \mathbb{R}^3$ is the camera center and $\mathbf{r}_d \in \mathbb{R}^3$ is the unit direction vector of the ray. For each ray, we sample $K = K_n + K_f$ points $\{t = t_k \mid t_k \in \mathbb{R}, \text{for } k = 1, 2, ..., K\}$ from the camera center to the surface, where K_n means near surface samples and K_f means free space samples. Using Eq. 5, the SDF values of these points can be written as \bar{s}_k . For near surface samples within a truncation region, i.e. samples where $S_{tr} = \{|\bar{D}(\mathbf{u}) - t_k| \leq tr\}$, we use the distance between the sampled point t_k and its surface $\bar{D}(\mathbf{u}_j)$ as an approximation of ground truth SDF value for supervision:

$$\mathcal{L}_{ns} = \sum_{\mathbf{u}_j \in \mathbf{u}} \sum_{t_k \in S_{tr}} ||\bar{s}_k - (\bar{D}(\mathbf{u}_j) - t_k)||_2.$$
 (7)

For points that are far from the surface $S_{fs} = \{\bar{D}(\mathbf{u}) - t_k > tr\}$, we apply a free-space loss:

$$\mathcal{L}_{fs} = \sum_{\mathbf{u}_j \in \mathbf{u}} \sum_{t_k \in S_{fs}} ||\bar{s}_k - 1||_2.$$
 (8)

During training, we jointly optimize explicit Gaussians and implicit SDF. The overall loss function for our hybrid scene representation is:

$$\mathcal{L}_{scene} = \mathcal{L}_c + \lambda_d \mathcal{L}_d + \lambda_n \mathcal{L}_n + \lambda_{ns} \mathcal{L}_{ns} + \lambda_{fs} \mathcal{L}_{fs}, \quad (9)$$

where we apply the color loss \mathcal{L}_c combining L1 with the D-SSIM term, distortion loss \mathcal{L}_d and depth-normal loss \mathcal{L}_n as defined in GOF [2]. In our approach, Gaussian opacity is directly linked to its position such that we can simplify the 3DGS densification process by removing its opacity reset step and corresponding hyperparameters.

3.4. Geometry Regularization

To ensure high-quality reconstructions on real-world scenes, we further employ an off-the-shelf monocular depth estimator* to obtain dense per-pixel depth priors. To address the scale ambiguity between the estimated depths and the scene, we apply linear regression regularization between rendered Gaussian depth \bar{D} and estimated monocular depth \hat{D} :

$$\mathcal{L}_{depth} = \sum log(1 + |(a * \hat{D} + b) - \bar{D}|).$$
 (10)

where a, b are learnable parameters for scale and shift. We also compute the gradient of monocular depth \hat{D} as a normal estimation $\hat{\mathbf{N}}$ to supervise rendered Gaussian normal $\bar{\mathbf{N}}$:

$$\mathcal{L}_{normal} = \sum (1 - \hat{\mathbf{N}} \cdot \bar{\mathbf{N}}). \tag{11}$$

A Total Variation (TV) regularization [45] is applied to the rendered depth maps \bar{D} to encourage surface smoothness:

$$\mathcal{L}_{tv} = \sum_{i,j} g_c \cdot (|\bar{D}_{i+1,j} - \bar{D}_{i,j}| + |\bar{D}_{i,j+1} - \bar{D}_{i,j}|), (12)$$

^{*}We use DepthAnything V2 [44] in this work.

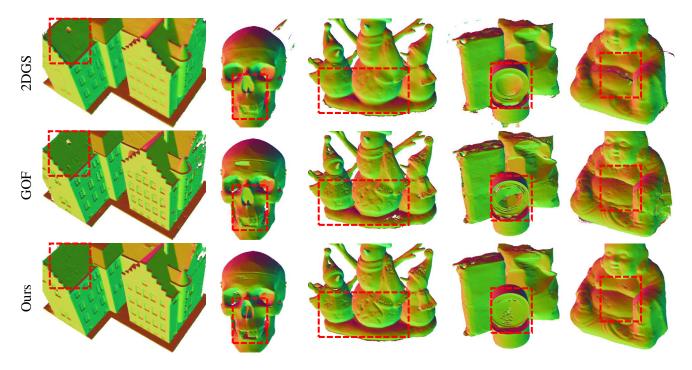


Figure 3. **Surface Reconstruction on the DTU Dataset [43].** We show normal maps rendered from the reconstructed meshes. 2DGS [1] produces overly smooth surfaces with floating artifacts and outliers, while GOF [2] struggles to reconstruct flat surfaces and performs poorly in regions with limited view coverage. In contrast, our method generates smooth yet detailed surfaces without floating artifacts, demonstrating its superior ability to represent geometry accurately.

where i, j are the indices of pixels, $g_c = \exp(-\nabla \mathbf{C})$ is the gradient of RGB image. Therefore, our geometry regularization is:

$$\mathcal{L}_{qeo} = \lambda_{depth} \mathcal{L}_{depth} + \lambda_{normal} \mathcal{L}_{normal} + \lambda_{tv} \mathcal{L}_{tv}.$$
 (13)

The overall training loss is defined as a combination of scene representation loss and geometry regularization:

$$\mathcal{L} = \mathcal{L}_{scene} + \mathcal{L}_{geo}. \tag{14}$$

4. Experiments

4.1. Experimental Settings

Datasets. We comprehensively evaluate our method on three public datasets: DTU dataset [46] which consists of indoor object scans; *Tanks and Temples* [47] which features six unbounded outdoor scenes; and Mip-NeRF 360 [22] for measuring novel view synthesis quality.

Baselines. We compare our proposal against several state-of-the-art methods. Among implicit methods, we compare against NeRF [18], VolSDF [48], NeuS [49], N-angelo [50], GeoNeus [51], Deep Blending [52], Instant NGP [53] and MipNeRF 360 [22]. As for explicit Gaussian methods, we compare against 3DGS [3], SuGaR [15], Mip-Splatting [23], 2DGS [1], GSurfel [54], 3DGSR [39] and GOF [2].

Metrics. To evaluate our method, we follow common practice and report surface accuracy as Chamfer Distance and

F1-score on DTU and Tanks and Temples, respectively. We measure the visual fidelity of the synthesized novel views with PSNR, SSIM and LPIPS [55] on Mip-NeRF 360.

Implementation. We perform single GPU training (NVIDIA 3090) and use by default $\lambda_d = 100, \lambda_n =$ $0.05, \lambda_{ns} = 1000, \lambda_{fs} = 10, \lambda_{depth} = 0.1, \lambda_{normal} =$ $0.1, \lambda_{tv} = 1$. Regarding the multi-resolution hash grid settings, we set the number of hash-grid level as 20 and voxel size as 0.01. We set truncation during SDF training as 0.02. For more implementation details, please refer to supp. mat.. Mesh Extraction. Our method supports different methods for mesh extraction, e.g., TSDF fusion [14], Marching Cubes [56] and Marching Tetrahedral [16]. For evaluation, we adopt Marching Tetrahedral [16] approach for fast mesh extraction based on SDF. In contrast to GOF [2] which requires ray tracing to build the opacity field during extraction, ours does not require any additional step and can be performed within two minutes instead of hours.

4.2. Geometry Evaluation

We begin by comparing our method with state-of-the-art surface reconstruction techniques on the DTU dataset [46]. As shown in Table 1, we report the Chamfer Distance, where our approach outperforms all existing 3DGS-based and implicit methods, achieving superior reconstruction quality and efficiency. For N-angelo [50], we follow prior

	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean
NeRF [18]	1.90	1.60	1.85	0.58	2.28	1.27	1.47	1.67	2.05	1.07	0.88	2.53	1.06	1.15	0.96	1.49
VolSDF [48]	1.14	1.26	0.81	0.49	1.25	0.70	0.72	1.29	1.18	0.70	0.66	1.08	0.42	0.61	0.55	0.86
NeuS [49]	1.00	1.37	0.93	0.43	1.10	0.65	0.57	1.48	1.09	0.83	0.52	1.20	0.35	0.49	0.54	0.84
N-angelo [50]	0.49	1.05	0.95	0.38	1.22	1.10	2.16	1.68	1.78	0.93	0.44	1.46	0.41	1.13	0.97	1.07
SuGaR [15]	1.47	1.33	1.13	0.61	2.25	1.71	1.15	1.63	1.62	1.07	0.79	2.45	0.98	0.88	0.79	1.33
2DGS [1]	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	1.27	0.76	0.70	1.40	0.40	0.76	0.52	0.80
GSurfel [54]	0.66	0.93	0.52	0.41	1.06	1.14	0.85	1.29	1.53	0.79	0.82	1.58	0.45	0.66	0.53	0.88
3DGSR [39]	0.68	0.84	0.70	0.39	1.16	0.87	0.77	1.48	1.26	0.87	0.69	0.80	0.42	0.64	0.60	0.81
GOF [2]	0.50	0.82	0.37	0.37	1.12	0.74	0.73	1.18	1.29	0.68	0.77	0.90	0.42	0.66	0.49	0.74
Ours	0.45	0.64	0.30	0.36	0.94	0.70	0.67	1.27	0.96	0.63	0.49	0.84	0.39	0.54	0.47	0.64

Table 1. Quantitative Comparison on the DTU Dataset [46]. We show the Chamfer distance. Our method achieves the highest reconstruction accuracy among all methods. For N-angelo [50], we report the results from UniSDF [57] reproduction and we show the vanilla results in supp. mat..

works and report the reproduced results from [57] as the published results cannot be reproduced. We visualize the qualitative results on DTU in Figure 3. Rendered normal maps from reconstructed meshes highlight that 2DGS [1] and GOF [2] often produce noisy, incomplete surfaces with numerous outliers and struggle with flat surface reconstruction. Our method addresses these issues by constraining Gaussians to the surface with SDF, resulting in smoother, more accurate reconstructions. It demonstrates that our method significantly improves surface reconstruction, particularly in challenging areas like flat surfaces or areas covered by sparse camera views.

We then compare our method on the Tanks and Temples dataset [47]. The evaluation is focused exclusively on foreground objects, as the ground truth point cloud does not cover background regions. As shown in Table 2, our method achieves results comparable to leading implicit approaches [50], while offering significantly faster optimization, training large unbounded scenes within 4 hours, compared to over 12 hours required by implicit methods. Additionally, our method achieves slightly improves results compared to GOF [2] with similar rendering efficiency and faster mesh extraction speed as shown in Table 4. Figure 4 provides a qualitative comparison, displaying rendered normal maps alongside ground truth color images. In the Barn scene, our method excels in reconstructing the flat rooftop where 2DGS [1] and GOF [2] struggle due to limited top-down camera views. The underlying continuity of the SDF field predicted by an MLP in our method enables accurate inference in this area. In the Caterpillar and Truck scenes, our approach successfully reconstructs reflective and transparent windows, a challenging task where both 2DGS [1] and GOF [2] fall short.

4.3. Novel View Synthesis

We further compare our method with state-of-the-art novel view synthesis (NVS) methods on the Mip-NeRF 360 dataset [22]. Quantitative results in Table 3 indicate that

		Implicit					
	NeuS [49]	GeoNeus [51]	N-angelo [50]	SuGaR [15]	2DGS [1]	GOF [2]	Ours
Barn	0.29	0.33	0.70	0.14	0.36	0.51	0.56
Caterpillar	0.29	0.26	0.36	0.16	0.23	0.41	0.39
Courthouse	0.17	0.12	0.28	0.08	0.13	0.28	0.30
Ignatius	0.83	0.72	0.89	0.33	0.44	0.68	0.72
Meetingroom	0.24	0.20	0.32	0.15	0.16	0.28	0.24
Truck	0.45	0.45	0.48	0.26	0.26	0.59	0.62
Average	0.38	0.35	0.50	0.19	0.30	0.46	0.47

Table 2. Quantitative Results on the Tanks and Temples Dataset [47]. Reconstructions are evaluated with the official evaluation scripts and we report F1-score. Our method outperforms all 3DGS-based surface reconstruction methods and performs comparably with the SOTA neural implicit methods while optimizing significantly faster.

	0	utdoor Sco	ene	Indoor Scene				
	PSNR ↑	SSIM ↑	LPIPS \downarrow	PSNR ↑	SSIM ↑	LPIPS \downarrow		
NeRF [18]	21.46	0.458	0.515	26.84	0.790	0.370		
Deep Blending [52]	21.54	0.524	0.364	26.40	0.844	0.261		
Instant NGP [53]	22.90	0.566	0.371	29.15	0.880	0.216		
MipNeRF360 [22]	24.47	0.691	0.283	31.72	0.917	0.180		
3DGS [3]	24.64	0.731	0.234	30.41	0.920	0.189		
SuGaR [15]	22.93	0.629	0.356	29.43	0.906	0.225		
Mip-Splatting [23]	24.65	0.729	0.245	30.90	0.921	0.194		
2DGS [1]	24.21	0.709	0.276	30.10	0.913	0.211		
GOF [2]	24.82	0.750	0.202	30.79	0.924	0.184		
Ours	24.77	0.742	0.210	30.83	0.928	0.167		

Table 3. Quantitative Results on Mip-NeRF 360 [22] Dataset. Our method achieves comparable NVS results, especially in the indoor scenes in terms of LPIPS.

our approach surpasses 2DGS [1] across all metrics in indoor scenes, while for outdoor scenes, the performance gap remains relatively small. However, our method significantly improves surface reconstruction quality. This suggests that our enhanced geometry representation leads to higher visual quality. Figure 5 presents a qualitative comparison of the extracted meshes. Consistent with our observations on the Tanks and Temples dataset [47], our method reconstructs smooth, detailed meshes. In contrast, 2DGS [1] yields overly smooth meshes, and GOF [2] produces noisy artifacts on flat surfaces.

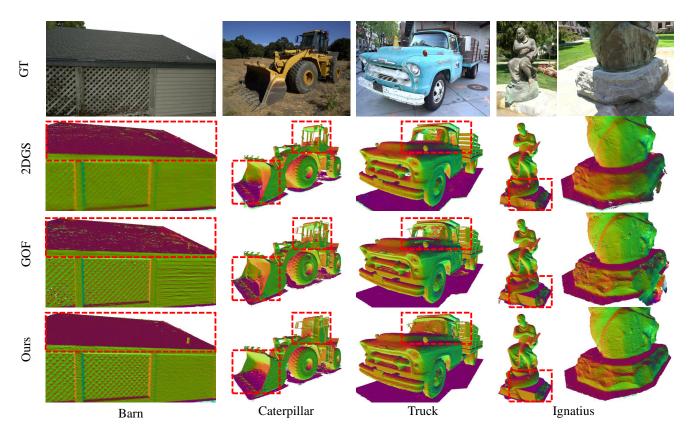


Figure 4. Surface Reconstruction on the Tanks and Temples Dataset [47]. We show the rendered normal map from reconstructed meshes. Resuls of 2DGS [1] and GOF [2] struggles to reconstruct smooth surfaces around transparent windows and produces noisy results in regions with sparse views. In contrast, our approach effectively handles transparent areas and achieves smooth reconstruction even in sparsely viewed regions.

4.4. Ablation Study

4.4.1 Mesh Extraction Comparison

Our method supports mesh extraction from 3D Gaussians using a variety of techniques. All mesh extractions are performed on a single NVIDIA 3090 GPU with 24GB of memory. To apply the Marching Cubes algorithm in unbounded scenes, we follow the 2DGS approach from [1], compressing the unbounded volume into a bounded range. However, the Marching Cubes algorithm is memory-intensive; on a 24GB GPU, it supports only low-resolution reconstructions, limiting its feasibility in unbounded scenes. For visualization purposes, we crop the Barn scene from the Tanks and Temples dataset [47] to a smaller range before running Marching Cubes. As shown in Figure 6, our method is still able to generate fine-detailed meshes with Marching Cubes.

In addition to Marching Cubes, we use Marching Tetrahedra for mesh extraction, which is not limited by resolution. In Table 4, we report the mesh extraction time of different methods, demonstrating the effectiveness of our Marching Tetrahedra approach. 2DGS uses TSDF fusion, reconstructing meshes in 10-30 minutes depending on reso-

Meshing Time (min)	TSDF	M.Cube	M.Tetrahedral
2DGS	~10-30	×	×
GOF	~10-30	×	~30
Ours	~10-30	~10-30	\sim 2
F1 Score	w/o Geo.		w/ Geo.
Ours	0.56		0.62

Table 4. **Quantitative Ablation Study.** We show the mesh extraction methods and time. We denote Marching Cube [56] as M.Cube and Marching Tetrahedral [16] as M.Tetrahedral. We also report the F1 score of Truck reconstruction with and without proposed geometry regularization.

lution, and GOF requires ray tracing to distill opacity fields from the 3DGS model before applying their own version of Marching Tetrahedra. In contrast, our approach leverages the SDF and 3DGS to initialize tetrahedral points, achieving high-detail mesh reconstruction within 2 minutes.

4.4.2 Geometry Regularization

We perform an ablation study of the proposed geometry regularization in the Truck scene from the Tanks and Temples dataset [47] which has transparent windows that are normally hard to reconstruct. Figure 6 shows visual compar-

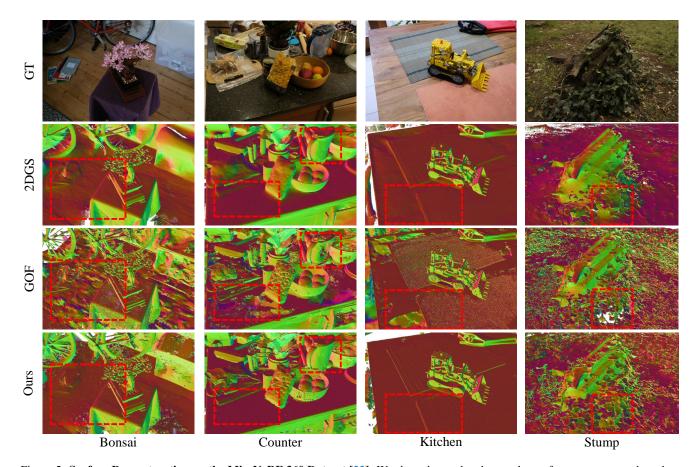


Figure 5. Surface Reconstruction on the Mip-NeRF 360 Dataset [22]. We show the rendered normal map from reconstructed meshes. GOF's [2] mesh is noisy and it fails to reconstruct smooth surface where there are reflection and curvature surface. In contrast, our method can generate smoother and detailed surface without any outlier points.

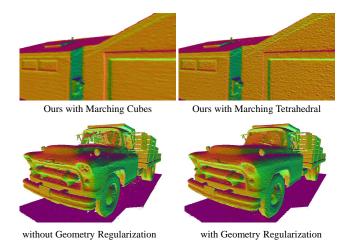


Figure 6. Ablation Study.

isons with and without the use of a depth estimator as a pseudo-ground-truth depth map, highlighting that pseudo geometry regularization significantly improves the reconstruction of flat and transparent surfaces. Quantitative results in Table 4 further demonstrate the effectiveness of this geometry regularization technique.

5. Conclusion

We present G2SDF, a novel method for efficient, highquality surface reconstruction that combines explicit Gaussian representations with implicit SDFs. By normalizing unbounded coordinates into a fixed range, we enable efficient positional embedding. Using a hash-grid-based SDF, we predict SDF values for each Gaussian and convert them to opacity values for rasterization, effectively aligning Gaussians to the surface for accurate geometry representation. G2SDF enables fast and detailed mesh extraction. Experimental results show that G2SDF matches state-of-theart methods in both surface reconstruction and novel view synthesis, producing high-quality meshes with fine details.

References

- [1] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 1, 2, 5, 6, 7
- [2] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024. 1, 2, 4, 5, 6, 7, 8
- [3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 2, 3, 5, 6
- [4] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024.
- [5] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xi-aofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21476–21485, 2024.
- [6] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20902–20911, 2024. 1
- [7] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. arXiv preprint arXiv:2309.16653, 2023.
- [8] Ye Yuan, Xueting Li, Yangyi Huang, Shalini De Mello, Koki Nagano, Jan Kautz, and Umar Iqbal. Gavatar: Animatable 3d gaussian avatars with implicit mesh learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 896–905, 2024.
- [9] Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. In *Proceedings of* the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5020–5030, 2024. 1
- [10] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 21634–21643, 2024. 1
- [11] 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. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 21357–21366, 2024.
- [12] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Os-

- wald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023. 1
- [13] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 1, 2
- [14] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings* of the 23rd annual conference on Computer graphics and interactive techniques, pages 303–312, 1996. 1, 2, 5
- [15] Antoine Guédon and Vincent Lepetit. Sugar: Surfacealigned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of* the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5354–5363, 2024. 1, 2, 5, 6
- [16] Jonas Kulhanek and Torsten Sattler. Tetra-nerf: Representing neural radiance fields using tetrahedra. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18458–18469, 2023. 2, 5, 7
- [17] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In Seminal graphics: pioneering efforts that shaped the field, pages 347–353. 1998.
- [18] 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. 2, 5, 6
- [19] 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. 2, 4
- [20] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 5459– 5469, 2022.
- [21] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492, 2020. 2
- [22] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 2, 5, 6, 8, 1, 3
- [23] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19447–19456, 2024. 2, 5, 6
- [24] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20923–20931, 2024.
- [25] Xiaowei Song, Jv Zheng, Shiran Yuan, Huan-ang Gao, Jing-wei Zhao, Xiang He, Weihao Gu, and Hao Zhao. Sags: Scale-adaptive gaussian splatting for training-free anti-aliasing. arXiv preprint arXiv:2403.19615, 2024. 2

- [26] 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. 2
- [27] Tianqi Liu, Guangcong Wang, Shoukang Hu, Liao Shen, Xinyi Ye, Yuhang Zang, Zhiguo Cao, Wei Li, and Ziwei Liu. Mvsgaussian: Fast generalizable gaussian splatting reconstruction from multi-view stereo. In European Conference on Computer Vision, pages 37–53. Springer, 2025.
- [28] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. arXiv preprint arXiv:2403.13806, 2024. 2
- [29] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20775–20785, 2024. 2
- [30] Yanyan Li, Chenyu Lyu, Yan Di, Guangyao Zhai, Gim Hee Lee, and Federico Tombari. Geogaussian: Geometry-aware gaussian splatting for scene rendering. In *European Confer*ence on Computer Vision, pages 441–457. Springer, 2025.
- [31] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, et al. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds. arXiv preprint arXiv:2403.20309, 2, 2024. 2
- [32] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20697– 20709, 2024. 2
- [33] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20654–20664, 2024. 2
- [34] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *arXiv* preprint arXiv:2406.06521, 2024. 2
- [35] Hanlin Chen, Fangyin Wei, Chen Li, Tianxin Huang, Yunsong Wang, and Gim Hee Lee. Vcr-gaus: View consistent depth-normal regularizer for gaussian surface reconstruction. arXiv preprint arXiv:2406.05774, 2024.
- [36] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. arXiv preprint arXiv:2406.01467, 2024.
- [37] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. Dn-splatter: Depth and normal priors for gaussian splatting and meshing. *arXiv* preprint arXiv:2403.17822, 2024.

- [38] Yaniv Wolf, Amit Bracha, and Ron Kimmel. Surface reconstruction from gaussian splatting via novel stereo views. *arXiv preprint arXiv:2404.01810*, 2024. 2
- [39] 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. 2, 4, 5, 6
- [40] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. IEEE Transactions on Visualization and Computer Graphics, 8(3):223–238, 2002. 3
- [41] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Coslam: Joint coordinate and sparse parametric encodings for neural real-time slam. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13293–13302, 2023. 3
- [42] Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. ACM Transactions on Graphics (ToG), 38(5):1–19, 2019. 4
- [43] 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. 5
- [44] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiao-gang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. arXiv preprint arXiv:2406.09414, 2024. 4
- [45] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In CVPR, 2022. 4
- [46] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 406–413. IEEE, 2014. 5, 6, 1, 2
- [47] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 5, 6, 7
- [48] 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. 5,
- [49] 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. 5, 6
- [50] 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. 5, 6, 1, 2
- [51] Qiancheng Fu, Qingshan Xu, Yew Soon Ong, and Wenbing Tao. Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 35:3403–3416, 2022. 5, 6
- [52] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for

- free-viewpoint image-based rendering. ACM Transactions on Graphics (ToG), 37(6):1–15, 2018. 5, 6
- [53] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 5, 6
- [54] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In ACM SIGGRAPH 2024 Conference Papers, pages 1–11, 2024. 5, 6
- [55] 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*, pages 586–595, 2018. 5
- [56] LORENSEN WE. Marching cubes: A high resolution 3d surface construction algorithm. *Computer graphics*, 21(1):7–12, 1987. 5, 7
- [57] Fangjinhua Wang, Marie-Julie Rakotosaona, Michael Niemeyer, Richard Szeliski, Marc Pollefeys, and Federico Tombari. Unisdf: Unifying neural representations for high-fidelity 3d reconstruction of complex scenes with reflections. arXiv preprint arXiv:2312.13285, 2023. 6, 2

G2SDF: Surface Reconstruction from Explicit Gaussians with Implicit SDFs

Supplementary Material

A. Implementation Details

A.1. Hyperparameters

In the following, we report implementation details and hyperparameters used for our method.

Gaussian Splatting. For hyperparameters used in the Gaussian Rasterization, we follow previsous works [1–3]. We train our Gaussian model and SDF network jointly with 30000 iterations. We set the initial learning rate for Gaussians' position as 0.00016 and the final initial learning rate as 0.0000016. And we set learning rates for opacity, scales and rotation as 0.05, 0.005, 0.001, respectively. We start the Gaussian densification after 500 iterations and until 15000 iterations. We densify Gaussians every 100 iterations and the gradient threshold for densification is 0.0002. We start the distortion loss \mathcal{L}_d and the depth-normal loss \mathcal{L}_n after 3000 iterations and 7000 iterations, respectively.

SDF Network. We use two-layer MLPs and the hidden dimension is 32. We initialize β from Eq. 6 as 100 and set learning rate as 0.005. We start train our SDF from 5000 iterations and until 30000 iterations. The learning rate for our SDF network is 0.002. For each iteration, we sample M=10000 pixels and $K_n=11, K_f=64$ points.

Geometry Regularization. We set the learning rate for a,b in Eq. 10 as 0.1. and for Tanks and Temples, we set $\lambda_{depth}=0.1, \lambda_{normal}=0.1$, for DTU and Mip-NeRF 360, we set $\lambda_{depth}=0.01, \lambda_{normal}=0.01$.

B. Distortion and Depth-Normal Loss

We apply a distortion loss [1] and depth-normal loss [2] as discussed in Section 3.3. The distortion loss concentrates the weight distribution along the rays by minimizing the distance between the ray-splat intersections:

$$\mathcal{L}_d = \sum_{i,j} \omega_i \omega_j |z_i - z_j|,\tag{15}$$

where $\omega_i = o_i(x) \prod_{j=1}^{i-1} (1 - o_j(x))$ is the blending weight of the *i*-th intersection and z_i is the depth of the intersection points. The depth-normal loss is defined as:

$$\mathcal{L}_n = \sum (1 - \bar{\mathbf{N}} \cdot \nabla(\bar{D})), \tag{16}$$

where $\nabla(\bar{D})$ is the gradient of rendered Gaussian Depth.

C. Additional Experimental Results

C.1. Results on DTU and Mip-NeRF 360

In Table 5, we present the results reported in the Neuralangelo publication [50] on the DTU dataset. It is important to note that the quantitative results from the Neuralangelo paper have been flagged by other studies and discussions on the official GitHub repository as non-reproducible[†].

To provide a more comprehensive evaluation, we include additional qualitative comparisons on the DTU dataset [46] in Figure 7 and further demonstrate results on the Mip-NeRF 360 dataset [22] in Figure 8. Our method excels in handling reflective surfaces and produces reconstructions that are not only smoother but also exhibit finer details, highlighting its capability for more accurate and visually appealing scene representations.

C.2. Mesh Extraction with Different Rasterizers

In Figure 9, we present meshes extracted using various Gaussian rasterizers, showcasing the versatility and adaptability of our method. These results demonstrate that our approach can be seamlessly integrated into existing Gaussian methods, no matter it's 2D Gaussians [1] or 3D Gaussians [2], enhancing their performance without requiring significant modifications. Furthermore, our method consistently delivers high-quality reconstructions with fine details, highlighting its effectiveness in capturing intricate scene structures such as transparent and reflection areas.

D. Limitation Discussion

The pseudo depth maps exhibit high uncertainty in the background regions and inconsistency across multiple views, which limits our method's ability to reconstruct distant background areas, such as the sky. However, since foreground objects are the primary focus in reconstruction tasks, our method demonstrates superior performance in these regions. To address this limitation, we plan to introduce an uncertainty-based weighting mechanism for the geometry regularization.

[†]Refer to this GitHub issue for details.

	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean
N-angelo [50]	0.49	1.05	0.95	0.38	1.22	1.10	2.16	1.68	1.78	0.93	0.44	1.46	0.41	1.13	0.97	1.07
N-angelo* [50]	0.37	0.72	0.35	0.35	0.87	0.54	0.53	1.29	0.97	0.73	0.47	0.74	0.32	0.41	0.43	0.61
Ours	0.45	0.64	0.30	0.36	0.94	0.70	0.67	1.27	0.96	0.63	0.49	0.84	0.39	0.54	0.47	0.64

Table 5. **Quantitative Comparison on the DTU Dataset [46]**. We show the Chamfer distance. For Neuralangelo [50], we report the results from UniSDF [57] reproduction as N-angelo, and the results from Neuralangelo publication as N-anglo*, which is not reproducible.

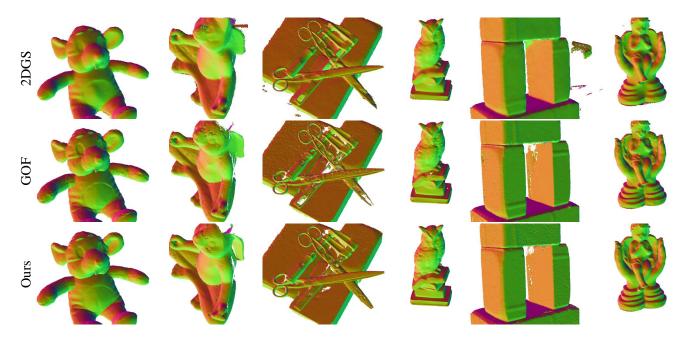


Figure 7. Additional Results on DTU [46].

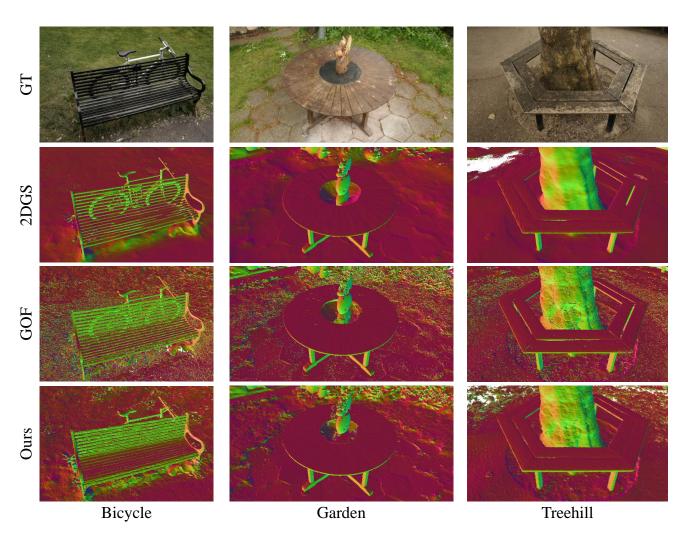


Figure 8. Additional Results on Mip-NeRF 360 [22].

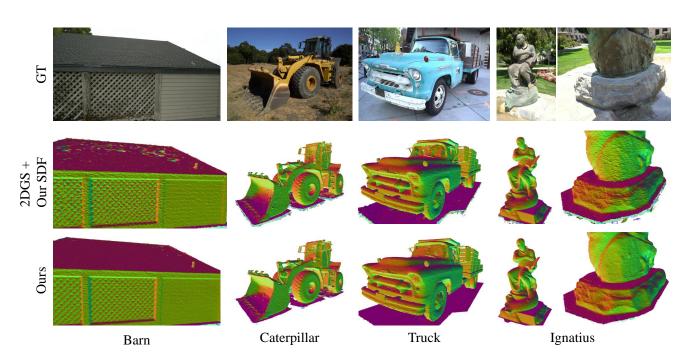


Figure 9. Mesh Extraction with Different Gaussian Rasterizer.