

Surface Reconstruction from 3D Gaussian Splatting via Local Structural Hints

Qianyi Wu¹, Jianmin Zheng², and Jianfei Cai^{1,2}

¹ Department of Data Science and AI, Monash University

² College of Computing and Data Science, Nanyang Technological University
{qianyi.wu,jianfei.cai}@monash.edu, asjmzheng@ntu.edu.sg

Abstract. This paper presents a novel approach for surface mesh reconstruction from 3D Gaussian Splatting (3DGS) [20], a technique renowned for its efficiency in novel view synthesis but challenged for surface reconstruction. The key obstacle is the lack of geometry hints to regulate the optimization of millions of unorganized Gaussian blobs to align to the true surface. This paper introduces local structural hints during training to address the challenge. We first leverage the prior knowledge from monocular normal and depth estimations to refine the covariance and mean of Gaussian primitives, enhancing their organization and providing crucial normal information for surface extraction. However, due to the highly discrete nature of Gaussian primitives, such geometry guidance remains insufficient for the alignment with the true surface. We then propose to construct a signed distance field by a moving least square (MLS) function over the Gaussians in each local region. More importantly, we further propose to jointly learn a neural implicit network to mimic and regularize the MLS function. The joint optimization helps the optimization of Gaussian Splatting towards accurate surface alignment. Extensive experimental results demonstrate the effectiveness of our method in achieving superior mesh quality compared with the SoTA surface reconstruction for 3DGS. More resources can be found on our project page: <https://qianyiwu.github.io/gsrc>.

Keywords: 3D Gaussian Splatting · Surface Reconstruction

1 Introduction

3D Gaussian Splatting (3DGS) [20] has garnered significant attention in the realm of 3D computer vision for its exceptional efficiency in modeling 3D radiance fields. Given multi-view images with corresponding camera poses, 3DGS initializes Gaussian primitives from a sparse point cloud that comes from COLMAP [41] and renders a novel view with a dedicated tile-based rasterization technique. With the dynamic densification operation on Gaussians including splitting and cloning, the final scene will be represented by millions of tiny Gaussians with unparalleled rendering efficiency. Despite the superior rendering efficiency and quality achieved by 3DGS over its implicit counterparts, Neural Radiance Field (NeRF) [3, 31, 32], its surface reconstruction ability is largely lagging behind. The

main reason is that a large number of discrete tiny Gaussians are noisy, unorganized, and do not align well with the underlying geometry, making it challenging to reconstruct the surface from them.

Prior efforts to address this intricate challenge of extracting surface meshes from 3D Gaussian Splatting have been sparse. Notably, SuGaR [15] represents a pioneering endeavor in this domain. SuGaR tries to align Gaussian primitives more closely to the surface and encourages the Gaussians well distributed over the surface. The cornerstone of SuGaR’s approach lies in designing two distinct regularization losses. The first is a density regularization term, predicated on the principle that the density at each query point should predominantly be influenced by *only one* Gaussian primitive. The second, a signed distance function (SDF) regularization term, delves into the nuanced relationship between a point’s density and its SDF relative to the nearest Gaussian, aiming to foster a more precise alignment with the surface. SuGaR employs the Poisson surface reconstruction [18] to extract the mesh from the aligned Gaussians, which is efficient for obtaining a mesh in minutes. SuGaR also proposes an optional refinement stage that binds Gaussian into the mesh for detailed optimization.

Despite these innovations, the meshes produced by SuGaR exhibit certain imperfections, notably including undesirable bumps and holes in textureless areas due to inaccuracies in the learned Gaussian primitives. We also observe the blocking artifacts resulting from SuGaR’s emphasis on preventing Gaussian blobs from overlapping. These artifacts not only compromise the mesh’s visual fidelity but also underscore the limitations of the regularization strategies in fully capturing complex surface geometry in 3DGS. This triggers us to rethink the problem and propose our new solution that centers on more accurate regulations of Gaussian geometry during the training process.

The key insight of our approach is *to leverage the local structure hints to guide the optimization of Gaussians*. Our primary idea involves the incorporation of additional geometric guidance to refine the geometry attributes of Gaussian blobs. By leveraging monocular surface normal predictions [11] of multiview images, we guide the learning process of the covariance of each Gaussian by approximating the normal with the eigenvector corresponding to the smallest eigenvalue. Extra geometry cues like monocular depth, can also be used to guide the learning of Gaussian positions. With the carefully designed monocular-cue-based losses, this simple idea not only directly enhances the alignment of Gaussians with the actual surface, but also provides more guidance for the texture-less region learning. It particularly improves the alignment from Gaussians to the surface with a rough orientation that mirrors the real-world geometry.

Although the improvements are afforded by monocular geometry cues, precise surface alignment remains elusive. Despite this, we recognize the potential of Gaussian blobs to act as indicators for providing local geometry information of the true surface. Drawing inspiration from the classical moving least square (MLS) technique [21, 42], the Gaussians inside a local region could define a smooth interpolation function like signed distance function (SDF), which approximates the truth surface in the zero level-set. However, the local geometry

information of the 3D Gaussians might not be consistent over a wider region. To address this, we propose a novel regularizer that leverages a neural implicit network to approximate the signed distance values of the MLS function at sampling points and the normals at Gaussian means. Through the joint optimization between the neural implicit network prediction and the MLS function, we achieve a significantly better alignment of Gaussians with the actual surface. Notably, the neural implicit network only serves as a regularizer to propagate gradients to Gaussians but is not used for the final surface reconstruction. After the model convergence, we also take Poisson reconstruction with the optimized Gaussian means and oriental normals to obtain the final mesh.

In addition to these methodological advancements, our framework incorporates a lightweight Gaussian Splatting architecture, Scaffold-GS [25], to enable an improved surface reconstruction quality over prior art while reducing the storage burden. Overall, we propose a new framework named **GSrec** for surface reconstruction from 3D **G**aussian **S**platting, which leverages two types of local structure hints, monocular normal (plus depth) and structured neural implicit function, to improve Gaussian Splatting’s surface alignment and consistency. Our contributions are summarized as follows:

- We incorporate the monocular normal guidance to augment Gaussians with improved covariance attributes. Despite a simple idea, the corresponding loss design is nontrivial. The monocular depth is also employed to guide the Gaussian locations. These geometry guidances effectively improve the surface alignment of Gaussians and help the learning in texture-less regions.
- We propose to predict the SDF of a true surface by applying MLS on the obtained Gaussians. Moreover, to ensure geometry consistency, we propose regularizing the MLS-based function prediction with a jointly learned neural implicit field. This joint optimization results in a significant improvement in the final surface extracted by Poisson reconstruction [18] over the surface-aligned Gaussian means and normals.
- Extensive experiments demonstrate that our **GSrec** framework outperforms the previous 3DGS surface reconstruction method by a large margin. Ablation studies verify the effectiveness of individual components.

2 Related Work

Radiance Field Reconstruction from Multiview Images. 3D Radiance Field has become a mainstream 3D reconstruction representation under the paradigm of "analysis by synthesis" due to the innovative Neural Radiance Field (NeRF) [31], which effectively promotes many downstream applications like novel view synthesis [2], scene understanding [65] and SLAM [66]. Taking into multi-view images with given camera poses, NeRF can establish volume density and view-dependent radiance field using an implicit neural network [30, 35, 43]. The established radiance field can synthesize images from new views by α -composed volume rendering [29] along the casting rays. However, both training and rendering speeds of NeRF are hindered due to the heavy network computation

burden from intensive ray point sampling. To overcome this issue, several initiatives have been proposed to mitigate this burden for expedited rendering, which mainly involves the explicit data structure like tensor [6], voxel grid [12, 45], and hyperplanes [4, 5] in the data representation. Despite these improvements, rendering efficiency remains a challenge in radiance field reconstruction.

A recent breakthrough, 3DGS [20], has emerged to effectively resolve this problem and boost both rendering speed and fidelity. To avoid the intensive query to the 3D space, 3DGS employs a carefully designed rasterization [67] approach to enable efficient GPU computation [20] by representing the entire scene as substantial Gaussian primitives. Each Gaussian primitive is endowed with both appearance and shape parameters to facilitate backpropagation via the rasterization-based rendering. However, despite 3DGS’s success in novel view synthesis, their discrete, unorganized, geometrically unaligned massive Gaussians pose a significant challenge for its adoption in surface reconstruction.

Surface Reconstruction for 3D Radiance Field. With the development of radiance field reconstruction, the necessity to extract an explicit surface [40, 48, 60] from it has become increasingly important for downstream applications like editing [13, 47, 49, 58, 64]. There are many successful attempts to obtain a surface under the structure of neural implicit representation [37, 48]. One key design is to predict the signed-distance function (SDF) by the network [61]. Two seminal works, NeuS [51] and VolSDF [59], first proposed surface reconstruction under the NeRF structure by converting SDF to density to enable volume rendering. The subsequent attempts further incorporate extra supervision such as normal [50], depth [16, 63], semantic [23, 27, 55–57] or dedicated training process [22, 52] to improve the quality of the final reconstructed surface. However, limited by the slow rendering of the NeRF structure, these approaches usually suffer from long training time (*e.g.*, up to 12+ hours).

While 3DGS has greatly improved the radiance field efficiency, attempts for its surface reconstruction remain challenging and important. For instance, NeuSG [7] proposes to leverage the Gaussian Splatting to improve the surface reconstruction of neural implicit surface [22, 51]. However, as their surface is represented as NeuS [51], the training time remains as long as [22] for more than 15 hours. SuGaR [15] is a pioneer in working towards surface reconstruction from 3DGS. The idea of SuGaR is to regulate the location and orientation of a Gaussian to lie on the surface. They assume the Gaussians not only to stay on the surface but also well-distributed. Consequently, they enforce each sampled query point to be only dominated by a single Gaussian, where two types of regularization are applied for density and SDF, respectively. These designs can generate a coarse mesh with roughly correct geometry by performing Poisson surface reconstruction [18]. After that, they propose a refinement strategy to bind Gaussians into the coarse triangle mesh and directly optimize it to get a final output. Thanks to the efficiency of 3DGS, SuGaR can obtain a surface mesh in less than 1 hour.

While SuGaR proposes the first solution for surface reconstruction from Gaussian Splatting, the surface quality is often not satisfactory, containing block-

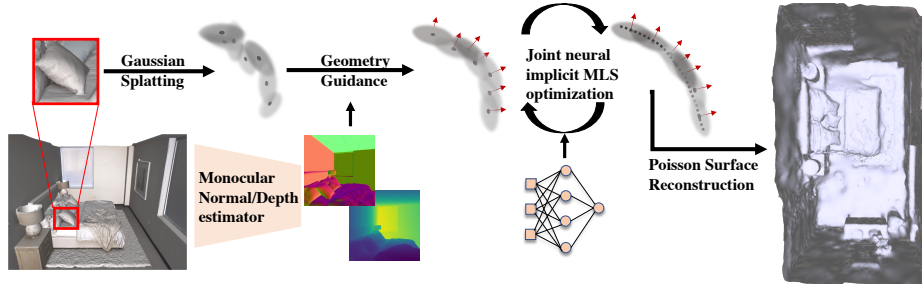


Fig. 1: Overview of GSrec. GSrec first leverages the monocular geometry cue as supervision to adjust the position and orientation of each 3D Gaussian primitive. After that, we jointly optimize a neural implicit representation to approximate local structural hints derived from Gaussian blobs in local regions. Finally, we use Poisson reconstruction [18] to extract the mesh from optimized Gaussians.

ing artifacts and unexpected bumps due to the single Gaussian assumption. Therefore, in this paper, we propose a new solution called GSrec. First, we incorporate the monocular normal and depth cues to better regulate the covariance and means of Gaussian primitives. Unlike SuGaR which clearly distinguishes respective Gaussians, we exploit the Gaussians in a local region to estimate an implicit moving least square (MLS) function for computing SDF. Second, we further propose to employ a neural implicit network to mimic and regularize the MLS function. Finally, we adopt Poisson reconstruction [18] as SuGaR for final mesh extraction. With the novel designs, our framework can well align the Gaussians to the real surface.

3 Method

Given a set of M posed RGB images $\mathcal{I} = \{I_1, \dots, I_M\}$ with corresponding camera parameters, 3DGS represents the scene as tons of Gaussian primitives. Our primary goal is to reconstruct the scene geometry from 3D Gaussians by aligning Gaussians with the real-world surface. This section introduces our novel framework called *GSrec*, which includes several modules as shown in Fig. 1. We will first introduce the preliminary knowledge of 3DGS in Sec. 3.1 and then elaborate on the technical details of each core module.

3.1 Preliminary: 3D Gaussian Splatting and Its Variants

3D Gaussian Splatting [20] represents a 3D scene as a bunch of 3D Gaussian blobs, each of which is defined by a set of attributes including location (mean) $\boldsymbol{\mu} \in \mathbb{R}^3$, covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3 \times 3}$, color $\boldsymbol{c} \in \mathbb{R}^3$, and opacity $o \in \mathbb{R}$. The covariance matrix is further decomposed as $\boldsymbol{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T$ with a diagonal scaling matrix $\mathbf{S} \in \mathbb{R}^{3 \times 3}$ and a rotation matrix $\mathbf{R} \in \mathbb{R}^{3 \times 3}$. With these attributes, a 3D Gaussian can be written as

$$G(\boldsymbol{x}) = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right), \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^3$ is a random 3D point. By splatting 3D Gaussians onto the 2D plane following Zwicker *et al.* [67], we obtain the projected 2D Gaussians $G'(\mathbf{x})$ and then utilize the α -composition [29, 46] to obtain the color in the image plane of an arbitrary viewpoint:

$$\hat{C} = \sum_i c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (2)$$

where i is the index of sorted Gaussians along the ray and α is the opacity after the 2D projection. Following [20], the means of 3D Gaussians are initialized from Structure-from-Motion (SfM) [41] points, and their attributes are optimized by the differentiable reconstruction loss on the given images.

As the sparsely initialized Gaussians may fail to represent scene details, 3DGS introduces a densification operation that performs splits and merges for Gaussians based on their training gradients. However, 3DGS usually consumes huge storage for real scenes [8, 25] due to the increased parameter number brought by such a spawning process. To alleviate the heavy storage burden, we adopt a light structural 3DGS framework Scaffold-GS [25] as our baseline model. Scaffold-GS proposes to use learnable anchor points as seeds and generate new Gaussians from these anchors along with their attributes (including color, opacity, and covariance) using tiny decoders. This design significantly reduces the storage requirement of 3DGS and more details can be found in [25].

3.2 Monocular Geometry Cue for 3DGS Optimization

While 3DGS mainly focuses on the image quality of view synthesis, it lags behind in scene surface reconstruction. The most significant hurdle lies in that the substantial Gaussians are unorganized, and their arbitrary orientations exhibit non-negligible influence on the final surface reconstruction due to their discrete property. Besides, sparsity and inaccuracy at their initialization from SfM add up to the difficulty of providing geometry information in texture-less regions.

To address the above issues, our first thought is to leverage the monocular geometry cues [11] to guide the 3DGS training, including both surface normal and depth. As a key indicator of the local 3D geometry, surface normal can provide a crucial hint to determine the orientation of a Gaussian blob. To be specific, the covariance of a Gaussian blob is controlled by a scaling matrix $\mathbf{S} = \text{diag}(s^1, s^2, s^3) \in \mathbb{R}^{3 \times 3}$ and a rotation matrix $\mathbf{R} = [\mathbf{r}^1, \mathbf{r}^2, \mathbf{r}^3]$ where $\mathbf{r}^i \in \mathbb{R}^3$ is the column vector of the rotation matrix. The normal can be approximated by the steepest ascent or descent direction of the density function in Eq. (1). In other words, we approximate the directional normal \mathbf{n} of each Gaussian as the column vector of \mathbf{R} corresponding to the minimum diagonal element of \mathbf{S} .

We adapt the rendering of tiled-based rasterization [20] to accumulate the normals of all Gaussians. The rendered normal $\hat{\mathbf{N}}$ can be calculated similarly to

Eq. (2) as follows:

$$\hat{\mathbf{N}} = \sum_i \mathbf{n}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad \mathbf{n}_i = \mathbf{r}_i^k, \quad k = \arg \min(s_i^1, s_i^2, s_i^3). \quad (3)$$

As $\hat{\mathbf{N}}$ is in the world coordinate system while the estimated monocular normal $\bar{\mathbf{N}}$ is defined in the camera coordinate system [11, 17], we normalize the rendered normal and apply the rotation transformation according to the given camera pose for the loss calculation. With a slight abuse of notation, we denote the adjusted rendered normal as $\hat{\mathbf{N}}'$ and apply the following normal consistency loss with monocular normal guidance $\bar{\mathbf{N}}$ during training:

$$\mathcal{L}_{\text{normal}} = \|\bar{\mathbf{N}} - \hat{\mathbf{N}}'\|_2 + \|1 - \bar{\mathbf{N}}^T \hat{\mathbf{N}}'\|_2. \quad (4)$$

This loss encourages the orientation of each Gaussian to align with the local surface normal. To further enhance the physical geometric property of the Gaussian normal \mathbf{n}_i , we add a regularization term for the scaling matrix \mathbf{S} to flatten the Gaussians. Without loss of the generality, we assume s^1 is the minimum scaling value, and define our regularization term using this minimum value and the harmonic mean as $\mathcal{L}_{\text{reg}} = \|s^1\|_1 + \|\frac{s^2}{s^3} + \frac{s^3}{s^2} - 2\|_1$. Herein, the second term discourages the Gaussian from degenerating into a needles-like shape, preventing normal ambiguity as more than one scaling value approaches zero. These carefully designed losses facilitate Gaussian primitives to converge to orientations aligned with local geometry characteristics. In addition, the estimated Gaussian normal can also serve as a point normal and benefit the subsequent surface reconstruction method like Poisson reconstruction [18].

In addition to the local geometry cue like surface normal to guide the covariance learning, we can also adopt the monocular depth to guide the training of the Gaussian mean. Inspired by the depth rendering from [15, 19, 26, 28], we also incorporate such a design in our framework by rendering the depth with the z-coordinate z_i of Gaussian mean $\boldsymbol{\mu}_i$ in the camera coordinate system:

$$\hat{D} = \sum_i z_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (5)$$

and optimizing it using a depth loss with monocular depth hint \bar{D} :

$$\mathcal{L}_{\text{depth}} = \|\bar{D} - (a\hat{D} + b)\|_2, \quad (6)$$

where a, b are scaling and shift parameters solved by given sampling rays to align non-metric depth \bar{D} with our metric rendering depth \hat{D} [38, 39, 63].

3.3 Neural Implicit Network for Regularization

With the guidance of monocular geometry cues, we have now augmented each Gaussian blob. However, we observe that the monocular geometry guidances

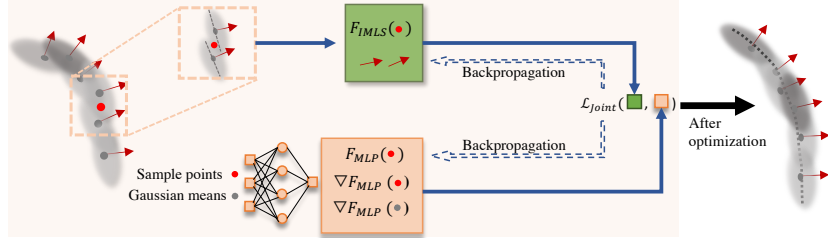


Fig. 2: Joint optimization of 3DGS and neural implicit representation. We propose a novel strategy to further align the Gaussians with the surface. We jointly train a neural implicit function approximating the MLS function and normal derived from Gaussians for regularization. The final Gaussians will align the surface more accurately to produce a better mesh.

cannot ensure a perfect alignment with the surface, due to the discrete and sparse nature of Gaussians. If the Poisson reconstruction is directly applied to the 3D Gaussians at this stage, it may lead to over-noisy geometry.

To address this, we need to regularize the scene SDF. Recall that SuGaR [15] introduces an SDF regularization term, which approximates the SDF of a point with its distance to the rendered depth. However, the regularization term is not accurate, since it is derived based on the assumption that the density of a point is determined by the nearest Gaussian, the approximated relationship between a point’s density and its SDF, and the approximation of the rendered depth. This motivates us to propose a better regularization design.

We take inspiration from a conventional method called *implicit moving least square (IMLS) function* [9, 21, 24, 42]. The definition of the IMLS function is given as: For a set of 3D points $\mathcal{P} = \{\mu_l \in \mathbb{R}^3\}$ with each point equipped with a unit normal vector \mathbf{n}_l , a signed distance function from a query point $\mathbf{x} \in \mathbb{R}^3$ to μ_l can be represented as $\langle \mathbf{x} - \mu_l, \mathbf{n}_l \rangle$, where $\langle \cdot, \cdot \rangle$ is the inner product. We can define an implicit function by weighted averaging of all point-wise signed distance functions as

$$F(x) := \frac{\sum_{\mu_l \in \mathcal{P}} \theta_l(\|\mathbf{x} - \mu_l\|) \langle \mathbf{x} - \mu_l, \mathbf{n}_l \rangle}{\sum_{\mu_l \in \mathcal{P}} \theta_l(\|\mathbf{x} - \mu_l\|)}, \quad (7)$$

where $\theta_l(\cdot)$ is a weight function, *e.g.* an RBF function [42]. As proved by Kooluri [21], *this IMLS function F is a tight approximation of the SDF of the original surface in a narrow band region.* As we have augmented the Gaussians with the normal attributes in Sec. 3.2, we can also define a similar function by treating all Gaussian means as \mathcal{P} and define the weight function $\theta_l(\mathbf{x})$ as the multiplication of opacity and the Gaussian density function, *i.e.* $\theta_l(\mathbf{x}) = o_l G_l(\mathbf{x})$:

$$F_{IMLS}(\mathbf{x}) = \frac{\sum_{l=1}^L o_l G_l(\mathbf{x}) \langle \mathbf{x} - \mu_l, \mathbf{n}_l \rangle}{\sum_{l=1}^L o_l G_l(\mathbf{x})}. \quad (8)$$

As the weight function $\theta_l(\cdot)$ decays with \mathbf{x} getting away from the mean $\boldsymbol{\mu}_l$, this function can be evaluated at a local region to accelerate computation. This local function tells where the surface is through the zero-level set [21].

With the IMLS function, we obtain an implicit function F_{IMLS} from explicit Gaussians. However, each place in F_{IMLS} is largely based on geometry hints derived from Gaussian blobs in a local region, which lacks structure and global consistency. This motivates us to propose to jointly learn a neural implicit SDF F_{MLP} based on an efficient backbone, *i.e.* Instant-NGP [32], which is good for modeling SDF efficiently and in a structured and global-consistent way. Our key idea is to use F_{MLP} to approximate F_{IMLS} while at the same time F_{MLP} is serving as a regularizer for F_{IMLS} (see Fig. 2). In particular, we define the joint optimization loss as

$$\mathcal{L}_{\text{joint}} = \sum_{\mathbf{x}, \boldsymbol{\mu}_l} (\|F_{\text{IMLS}}(\mathbf{x}) - F_{\text{MLP}}(\mathbf{x})\|_2 + \|\frac{\nabla F_{\text{MLP}}(\boldsymbol{\mu}_l)}{\|\nabla F_{\text{MLP}}(\boldsymbol{\mu}_l)\|} - \mathbf{n}_l\|_2 + (\|\nabla F_{\text{MLP}}(\mathbf{x})\|_2 - 1)^2), \quad (9)$$

where the first and second terms are the zero-order and first-order constraints, and the third term is the commonly used eikonal constraint [14] for F_{MLP} . We use the rendered depth \hat{D} in Eqn.(5) to back-project 2D pixels into 3D points, then conduct random jittering to construct the set of sampling points $\mathbf{x} \in \mathcal{X}$. The loss function will backpropagate the gradients to both the Gaussian Splatting field and the neural SDF. Our experiments will show that such a joint training scheme significantly improves the reconstructed surface, which encourages the Gaussian primitives close to the true surface, likely due to the inductive smoothness bias of implicit representation [1, 14, 36, 62].

In addition to the vanilla IMLS definition, we further introduce a Robust IMLS (RIMLS) by applying a 1-D Gaussian kernel inputted with the norm of the difference between the normalized gradient ∇F_{MLP} at query point \mathbf{x} and the Gaussian normal [34, 53]. Specifically, the weight function for θ_l is defined as $\theta_l(\mathbf{x}) = o_l G_l(\mathbf{x}) \phi(\|\mathbf{n}_l - \frac{\nabla F_{\text{MLP}}(\mathbf{x})}{\|\nabla F_{\text{MLP}}(\mathbf{x})\|}\|)$, where ϕ is defined the 1-D kernel with variance σ_n^2 , *i.e.* $\phi(x) = \exp(-x^2/\sigma_n^2)$. Consequently, the joint loss $\mathcal{L}_{\text{joint}}$ is defined by replacing F_{IMLS} with F_{RIMLS} . Details can be found in the supplementary.

3.4 Model Training

The primary goal of our model is to optimize the 3D Gaussian attributes for better surface reconstruction. At first, we train the model with the color reconstruction loss as in original 3DGS [20] together with the monocular cue related losses in Sec. 3.2. The overall loss function at this stage is

$$\mathcal{L}_{\text{stage1}} = \mathcal{L}_{\text{color}} + \lambda_d \mathcal{L}_{\text{depth}} + \lambda_n \mathcal{L}_{\text{normal}} + \lambda_r \mathcal{L}_{\text{reg}}, \quad (10)$$

where $\lambda_d, \lambda_n, \lambda_r$ are the loss tradeoff weights. Following the Scaffold-GS [25] design, the anchor points will stop growing or pruning after 15000 iterations. Once the number of anchors is fixed, we start the joint optimization by involving $\mathcal{L}_{\text{joint}}$ till the end. The total loss at this later stage is defined as

$$\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{color}} + \lambda_d \mathcal{L}_{\text{depth}} + \lambda_n \mathcal{L}_{\text{normal}} + \lambda_r \mathcal{L}_{\text{reg}} + \lambda_j \mathcal{L}_{\text{joint}}, \quad (11)$$

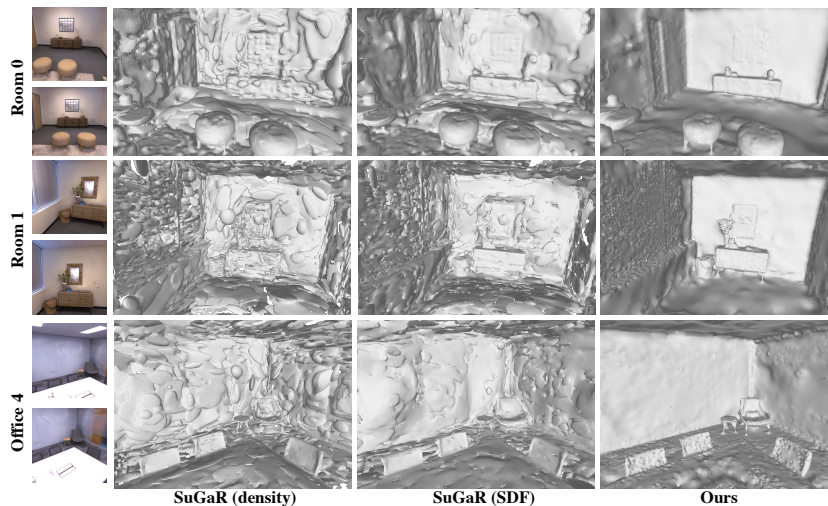


Fig. 3: Qualitative results on Replica [44]. The surface produced by our approach achieves better quality compared with SuGaR [15] owing to the structural hints.

where λ_j is the loss weight for $\mathcal{L}_{\text{joint}}$. After the optimization, we use 3D Gaussian means and normals for Poisson surface reconstruction [18] to extract the reconstructed meshes.

4 Experiments

Implementation details. We implement our GSrec based on Scaffold-GS [25] framework using PyTorch. Our experiments were run on a 24GB NVIDIA RTX 3090 GPU. We set the $\lambda_d, \lambda_n, \lambda_r, \lambda_j$ in Eqn.(11) as 0.1, 0.1, 0.01, 1 respectively. The total training iterations is set as 30000 following 3DGS [20]. The monocular geometry cues used in Sec. 3.2 are extracted by Omnidata [11]. The neural implicit network used in Sec. 3.3 adopts the multi-resolution hash grid [32] embedding with a level of 16 followed by a two-layer tiny MLP for fast convergence during training. We initialize the MLP weights following the geometric initialization [14]. We randomly sample 8192 pixels in the image to construct the sampling set \mathcal{X} . We set $L = 50$ in calculating the IMLS/RIMLS function. To facilitate the efficiency of MLS calculation, we constrain only utilizing these Gaussian locations within a radius of 0.01 of sampling points for computation.

Dataset and metrics. We conduct our experiments on 1) **Replica** [44] is a synthesized dataset with accurate camera poses and ground truth mesh for evaluation. we use 8 scenes from Replica for experiments. For quantitative evaluation of surface quality, we measure Chamfer Distance, Normal Consistency Score and F-score with a threshold of 5cm on Replica. 2) **ScanNet** [10] is a real-world dataset captured with challenging image quality. Following the setting [16, 63], 4 scenes are selected from ScanNet for experiments and compared ours with both the

Table 1: The quantitative results of the scene reconstruction on 8 Replica scenes. We compare our method against the SoTA surface reconstruction method for 3D Gaussian Splatting [15] in terms of Chamfer distance and F-score.

Chamfer distance↓	room0	room1	room2	office0	office1	office2	office3	office4	average
SuGaR (density) [15]	8.84	9.79	12.59	7.89	14.31	11.24	9.54	12.06	10.78
SuGaR (SDF) [15]	7.28	8.10	10.85	7.11	14.51	9.98	8.64	10.09	9.57
GSrec (Ours)	6.08	6.08	9.97	5.20	7.10	8.31	6.74	7.16	7.08
F-score ↑	room0	room1	room2	office0	office1	office2	office3	office4	average
SuGaR (density) [15]	52.93	46.66	39.68	54.50	34.63	45.64	52.48	41.97	46.06
SuGaR (SDF) [15]	60.93	55.45	47.88	61.39	32.71	49.84	57.48	51.11	52.10
GSrec (Ours)	76.02	70.47	61.70	77.60	58.31	60.95	63.03	69.66	67.22

state-of-the-art neural implicit surface methods and Gaussian-splatting-based methods. More details can be found in the Sec. A of supplementary.

Baseline Methods. We mainly compare our method with the recent representative works in the realm of 3D Gaussian Splatting in surface reconstruction.

-SuGaR [15] (SDF/density). SuGaR is the first attempt to reconstruct a surface mesh from Gaussian Splatting. The idea of SuGaR is to enforce the Gaussian primitive to align with the real surface by applying density or SDF regularization terms during training. We term these two variants as SuGaR (SDF)/(density). To further enhance the quality of the final reconstructed mesh, SuGaR proposes an extra refinement stage to adjust the new Gaussian attached to the coarse mesh from the first stage. We compared the final refinement mesh from SuGaR with our output. Note that we maintain the Poisson depth parameter the same for both our approach and SuGaR for a fair comparison.

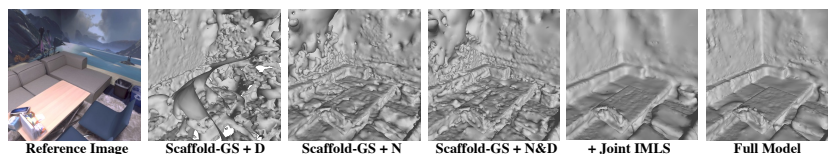
-Scaffold-GS [25] w monocular geometry cue (Scaffold-GS+N/D). As our base model is Scaffold-GS, we add several variants of it for comparison. We add additional monocular geometry guidance including depth and normal cues to regulate Scaffold-GS training. Since we have the Gaussian means and the normals for this baseline, we can also apply [18] to extract a mesh from it.

4.1 Experimental results on Replica

Comparison with the previous approach. SuGaR [15] is the most representative approach for surface reconstruction from Gaussian Splatting. We provide the quantitative results of all scene reconstruction in Replica in Tab. 1 where our method outperforms both of SuGaR’s variants with a clear margin. The octree depth for Poisson reconstruction is set as the same for both SuGaR and our approach for fair comparison. As seen from Fig. 3, the surface reconstruction of SuGaR still contains substantial undesired artifacts in the flat region. We found that SuGaR (density) is not as good as SuGaR (SDF). The main drawback of utilizing density regularization lies in that it forces the density of each query point to be dominated by one single Gaussian, which leads to significant blocking artifacts. Although SuGaR (SDF) achieves better quality, the inaccurate depth in the texture-less region such as walls in Fig. 3 still leads to the bumps of output mesh. The derivation of SDF regularization [15] is also based on the

Table 2: Ablation study on Replica. We compared the key components with the variants of [25] including the guidance and the joint optimization.

Method	Geometry	Guidance	MLS design		Reconstruction metric	
	Normal	Depth	IMLS	RIMLS	Normal-C \uparrow	F-score \uparrow
(a) Scaffold-GS [25]+D	\times	\checkmark	\times	\times	66.53 \pm 2.56	55.33 \pm 6.29
(b) Scaffold-GS [25]+N	\checkmark	\times	\times	\times	80.03 \pm 2.85	65.28 \pm 7.11
(c) Scaffold-GS [25]+N&D	\checkmark	\checkmark	\times	\times	80.32 \pm 2.42	64.00 \pm 7.29
(d) (b)+joint IMLS	\checkmark	\times	\checkmark	\times	83.96 \pm 3.17	65.46 \pm 8.67
(e) (b)+joint RIMLS	\checkmark	\times	\times	\checkmark	83.99 \pm 2.63	66.04 \pm 8.34
(f) Full model (Ours)	\checkmark	\checkmark	\times	\checkmark	85.23 \pm 2.38	67.22 \pm 7.26

**Fig. 4: Reconstructed surface by ablating proposed components on Replica [44].** Our approach significantly improves the quality of the final mesh.

single Gaussian assumption, which potentially makes the output mesh rough. Our approach takes inspiration from MLS for better SDF estimation near Gaussians and uses a novel implicit network as a regularizer. Our framework benefits from the geometry hints provided by monocular prediction, which gives more guidance in Gaussian location and orientation optimization. Together with the joint optimization design, the final output surface achieves the best quality on surface reconstruction. Notably, the average training time of our approach on this dataset is about 40 minutes, which is similar to SuGaR. In terms of storage, thanks to the Scaffold-GS baseline, our framework only takes about 45MB for each scene while the size of the SuGaR model is more than 150MB.

Ablation study. To clarify our framework’s key contributions, we showcase the effectiveness of its individual modules by answering these insightful questions.

-1) How does monocular geometry cue help reconstruction? Our first key design is to incorporate monocular normal guidance into the training. It provides very important hints about the Gaussian orientation, which also significantly improves the quality of Poisson reconstruction. As shown in Fig. 4 and Tab. 2, the inaccurate normal estimated by the density gradient will lead to a degraded iso-surface estimation compared with Scaffold-GS+D and Scaffold-GS+N. We discover the naive incorporation of depth guidance will slightly impair the F-score, which might be due to the inaccurate depth calculation in Eqn.(5). As a result, it remains an open problem to find an accurate depth rendering solution for the 3DGS structures. Notably, in our approach, we incorporate sampling points near depth to jointly optimize the neural implicit model. Although the depth rendering might be not accurate enough, our joint optimization could also benefit from it to construct the sampling set \mathcal{X} and lead to better reconstruction.

-2) What is the role of joint optimization with the neural network regularizer? A key novelty of our framework is to utilize the neural implicit

Table 3: Ablation study about the joint MLS optimization and the MLS computation. We provide an in-depth analysis by verifying the effectiveness of the joint optimization loss and the number of Gaussians used in the MLS computation.

	Normal-C \uparrow F-score \uparrow			Normal-C \uparrow F-score \uparrow	
w/o MLS term	84.15	64.55	L=1	84.49	64.52
w/o gradient term	81.38	63.56	L=10	84.63	65.25
w/o eikonal term	84.64	66.62	L=30	85.03	65.28
Full joint loss	85.23	67.22	Ours (L=50)	85.23	67.22

(a) Ablation study about the joint loss (b) The number of Gaussians for MLS

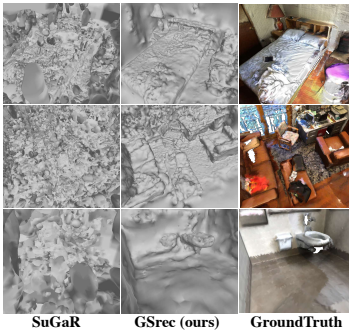
network as a regularizer. To delve into its functionality, we show the results by ablation study over the MLS-based joint optimization. By comparing the methods (b), (d) and (e) in Tab. 2, we find the normal consistency score (Normal-C) obtains significant improvement. This is because both variants of the MLS function in joint optimization can increase the alignment of Gaussians and true surface, which can be demonstrated in Fig. 4. From Tab. 2, we observe the normal consistency and F-score are better by using RIMLS. Compared with the vanilla MLS function which only depends on the Gaussian means and normal, the extra normal difference weights could somehow capture more details of the surface [34]. To further inspect the functionality of joint optimization, we conduct the following ablation studies to better understand the mechanism.

-3) Which part of $\mathcal{L}_{\text{joint}}$ is more important? The joint loss in Eqn.(11) uses a neural implicit network to approximate the local structural hints derived from Gaussians for joint optimization. There are three parts of joint loss: an MLS term that uses MLP to approximate MLS function, a gradient term by minimizing the gradient of MLP with Gaussian normal, and an eikonal term to regulate the MLP itself to predict valid SDF. As given in Tab. 3 (a), we found that with only first-order constraint (w/o MLS term), the method can also improve slightly in terms of F-score but benefit a lot in normal consistency (compared with method (c) in Tab. 3). The gradient term is crucial in joint optimization, which provides strong constraints for the MLP [1]. While keeping the MLS term with the gradient term in the joint loss (w/o eikonal term), the F-score can be significantly improved thanks to the zero-order approximation of the MLS value. The eikonal constraint on the MLP itself could further regulate the implicit part leading to a better final surface reconstruction.

-4) How does Gaussian number contribute? One key insight difference between our solution and SuGaR [15] lies in the treatment of different Gaussian numbers contributing to surface estimation. To demonstrate the superiority of considering multiple Gaussians rather than one, we ablate the choice of L when calculating RIMLS. As given in Tab. 3 (b), although $L = 1$ has achieved good results, the F-score keeps improving as L gets increased. As the Gaussian Splatting is a sparse and discrete representation, we argue that using a weighted average for multiple Gaussians could increase the accuracy of the zero level-set estimation. Consequently, the SDF estimation of our approach provides a more accurate structural hint compared to SuGaR.

Table 4: Scene-level 3D Reconstruction on ScanNet. (a) The quantitative results of several neural implicit surface reconstruction methods and the 3DGS-based approaches [15] (b) Reconstructed surface by SuGaR [15] and our approach.

	Chamfer-L1↓F-score ↑	
COLMAP [41]	0.141	0.537
UNISURF [33]	0.359	0.267
NeuS [51]	0.194	0.291
VolSDF [59]	0.267	0.346
Manhattan-SDF [16]	0.070	0.602
MonoSDF (Grid) [63]	0.064	0.626
MonoSDF (MLP) [63]	0.042	0.733
SuGaR (density)	0.284	0.217
SuGaR (SDF)	0.269	0.184
Ours	0.068	0.599



(a) Comparison with SOTA approaches (b) Reconstructed Mesh visualization

4.2 Experimental results on real-world dataset

To further investigate our method to the real-world dataset, we evaluate our approach on the challenging ScanNet [10] dataset, which contains motion blur and inaccurate camera poses. We compare with previous strong baselines of neural implicit surface [16, 33, 51, 59, 63] and the 3DGS-based approach SuGaR [15]. As reported in Tab. 4 (a), we find that our approach is comparable with Manhattan-SDF [16] and the MonoSDF [63] in which multi-resolution grid embedding [32] is utilized, while our training time is $12\times$ less (less than 1 hour *v.s.* more than 12 hours) compared over those neural implicit network-based ones thanks to structure of 3DGS. This shows the potential ability of 3DGS to achieve high-quality surface reconstruction for real-world capture. Although the MonoSDF (MLP) adopts pure MLP structure which shows robustness to the camera noise, the training time of such a variant gets much longer than others.

Regarding SuGaR [15], which is built over the same 3DGS structure, it is significantly impaired by the noisy camera poses and the inferior image quality and outputs unsatisfied reconstruction results, as shown in Tab. 4 (b). On the contrary, in our approach, the monocular geometry cues provide extra guidance to effectively prevent unexpected Gaussian blobs in the empty space, and the neural implicit network regularization also helps improve the smoothness.

5 Conclusion

We have presented a novel framework coined GSrec to address the challenge of misalignment between Gaussians and real-surface for surface reconstruction from 3DGS, leveraging monocular geometry cue and neural implicit networks as regularization to enhance mesh precision and quality. Our approach significantly outperforms existing methods, showcasing 3DGS’s potential for detailed and accurate surface reconstruction.

Acknowledgement

We sincerely thank Yihang Chen for proofreading. This work is partly supported by the Monash FIT Start-up Grant and MOE AcRF Tier 1 Grant of Singapore (RG12/22).

References

1. Atzmon, M., Lipman, Y.: Sald: Sign agnostic learning with derivatives. In: ICLR (2021) [9](#), [13](#)
2. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. ICCV (2021) [3](#)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. CVPR (2022) [1](#)
4. Cao, A., Johnson, J.: Hexplane: A fast representation for dynamic scenes. In: CVPR. pp. 130–141 (2023) [4](#)
5. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L.J., Tremblay, J., Khamis, S., et al.: Efficient geometry-aware 3d generative adversarial networks. In: CVPR. pp. 16123–16133 (2022) [4](#)
6. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: ECCV. pp. 333–350. Springer (2022) [4](#)
7. Chen, H., Li, C., Lee, G.H.: Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance. arXiv preprint arXiv:2312.00846 (2023) [4](#)
8. Chen, Y., Wu, Q., Lin, W., Harandi, M., Cai, J.: Hac: Hash-grid assisted context for 3d gaussian splatting compression. In: European Conference on Computer Vision (2024) [6](#), [2](#)
9. Cheng, Z.Q., Wang, Y., Li, B., Xu, K., Dang, G., Jin, S.: A survey of methods for moving least squares surfaces. In: VG/PBG@ SIGGRAPH. pp. 9–23 (2008) [8](#)
10. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In: CVPR (2017) [10](#), [14](#), [3](#)
11. Eftekhar, A., Sax, A., Malik, J., Zamir, A.: Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In: ICCV. pp. 10786–10796 (2021) [2](#), [6](#), [7](#), [10](#)
12. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR. pp. 5501–5510 (2022) [4](#)
13. Gao, L., Yang, J., Zhang, B.T., Sun, J.M., Yuan, Y.J., Fu, H., Lai, Y.K.: Mesh-based gaussian splatting for real-time large-scale deformation. arXiv preprint arXiv:2402.04796 (2024) [4](#)
14. Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes. In: ICML (2020) [9](#), [10](#)
15. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In: CVPR (2024) [2](#), [4](#), [7](#), [8](#), [10](#), [11](#), [13](#), [14](#)
16. Guo, H., Peng, S., Lin, H., Wang, Q., Zhang, G., Bao, H., Zhou, X.: Neural 3d scene reconstruction with the manhattan-world assumption. In: CVPR (2022) [4](#), [10](#), [14](#), [3](#)

17. Kar, O.F., Yeo, T., Atanov, A., Zamir, A.: 3d common corruptions and data augmentation. In: CVPR. pp. 18963–18974 (2022) [7](#)
18. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. vol. 7, p. 0 (2006) [2](#), [3](#), [4](#), [5](#), [7](#), [10](#), [11](#)
19. Keetha, N., Karhade, J., Jatavallabhula, K.M., Yang, G., Scherer, S., Ramanan, D., Luiten, J.: Splatam: Splat, track map 3d gaussians for dense rgb-d slam. arXiv (2023) [7](#)
20. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (July 2023) [1](#), [4](#), [5](#), [6](#), [9](#), [10](#)
21. Kolluri, R.: Provably good moving least squares. ACM Transactions on Algorithms (TALG) **4**(2), 1–25 (2008) [2](#), [8](#), [9](#)
22. Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: CVPR (2023) [4](#)
23. Li, Z., Lyu, X., Ding, Y., Wang, M., Liao, Y., Liu, Y.: Rico: Regularizing the unobservable for indoor compositional reconstruction. In: ICCV (2023) [4](#)
24. Liu, S.L., Guo, H.X., Pan, H., Wang, P.S., Tong, X., Liu, Y.: Deep implicit moving least-squares functions for 3d reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1788–1797 (2021) [8](#), [2](#)
25. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: CVPR (2024) [3](#), [6](#), [9](#), [10](#), [11](#), [12](#), [1](#)
26. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In: 3DV (2024) [7](#)
27. Lyu, X., Chang, C., Dai, P., Sun, Y.t., Qi, X.: Total-decom: Decomposed 3d scene reconstruction with minimal interaction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20860–20869 (2024) [4](#)
28. Matsuki, H., Murai, R., Kelly, P.H.J., Davison, A.J.: Gaussian Splatting SLAM. In: CVPR (2024) [7](#)
29. Max, N.: Optical models for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics **1**(2), 99–108 (1995) [3](#), [6](#)
30. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: CVPR (2019) [3](#)
31. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020) [1](#), [3](#)
32. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM TOG. **41**(4), 102:1–102:15 (Jul 2022) [1](#), [9](#), [10](#), [14](#), [2](#)
33. Oechsle, M., Peng, S., Geiger, A.: Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: ICCV (2021) [14](#), [4](#)
34. Öztireli, A.C., Guennebaud, G., Gross, M.: Feature preserving point set surfaces based on non-linear kernel regression. In: Computer graphics forum. vol. 28, pp. 493–501. Wiley Online Library (2009) [9](#), [13](#)
35. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019) [3](#)
36. Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., Courville, A.: On the spectral bias of neural networks. In: International Conference on Machine Learning. pp. 5301–5310. PMLR (2019) [9](#)

37. Rakotosaona, M.J., Manhardt, F., Arroyo, D.M., Niemeyer, M., Kundu, A., Tombari, F.: Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes. In: 3DV (2024) 4
38. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. ICCV (2021) 7
39. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence 44(3) (2022) 7
40. Reiser, C., Garbin, S., Srinivasan, P.P., Verbin, D., Szeliski, R., Mildenhall, B., Barron, J.T., Hedman, P., Geiger, A.: Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. arXiv preprint arXiv:2402.12377 (2024) 4
41. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR. pp. 4104–4113 (2016) 1, 6, 14, 2
42. Shen, C., O’Brien, J.F., Shewchuk, J.R.: Interpolating and approximating implicit surfaces from polygon soup. In: ACM SIGGRAPH 2004 Papers, pp. 896–904 (2004) 2, 8
43. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: NeurIPS (2019) 3
44. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S., et al.: The replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019) 10, 12
45. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In: CVPR. pp. 5459–5469 (2022) 4
46. Tagliasacchi, A., Mildenhall, B.: Volume rendering digest (for nerf) (2022) 6
47. Tang, J., Ren, J., Zhou, H., Liu, Z., Zeng, G.: Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. arXiv preprint arXiv:2309.16653 (2023) 4
48. Tang, J., Zhou, H., Chen, X., Hu, T., Ding, E., Wang, J., Zeng, G.: Delicate textured mesh recovery from nerf via adaptive surface refinement. In: ICCV (2023) 4
49. Waczyńska, J., Borycki, P., Tadeja, S., Tabor, J., Spurek, P.: Games: Mesh-based adapting and modification of gaussian splatting. arXiv preprint arXiv:2402.01459 (2024) 4
50. Wang, J., Wang, P., Long, X., Theobalt, C., Komura, T., Liu, L., Wang, W.: Neuris: Neural reconstruction of indoor scenes using normal priors. In: ECCV (2022) 4
51. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In: NeurIPS (2021) 4, 14
52. Wang, Y., Skorokhodov, I., Wonka, P.: Hf-neus: Improved surface reconstruction using high-frequency details. In: NeurIPS. vol. 35, pp. 1966–1978 (2022) 4
53. Wang, Z., Wang, P., Wang, P., Dong, Q., Gao, J., Chen, S., Xin, S., Tu, C., Wang, W.: Neural-impls: Self-supervised implicit moving least-squares network for surface reconstruction. arXiv preprint arXiv:2109.04398 (2021) 9
54. Wang, Z., Wang, P., Wang, P., Dong, Q., Gao, J., Chen, S., Xin, S., Tu, C., Wang, W.: Neural-impls: Self-supervised implicit moving least-squares network for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics pp. 1–16 (2023). <https://doi.org/10.1109/TVCG.2023.3284233> 2
55. Wu, Q., Liu, X., Chen, Y., Li, K., Zheng, C., Cai, J., Zheng, J.: Object-compositional neural implicit surfaces. In: ECCV (2022) 4

56. Wu, Q., Wang, K., Li, K., Zheng, J., Cai, J.: Objectsdf++: Improved object-compositional neural implicit surfaces. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2023) [4](#)
57. Wu, T., Zheng, C., Cham, T.J., Wu, Q.: Clusteringsdf: Self-organized neural implicit surfaces for 3d decomposition. arXiv preprint arXiv:2403.14619 (2024) [4](#)
58. Yan, H., Li, Y., Wu, Z., Chen, S., Sun, W., Shang, T., Liu, W., Chen, T., Dai, X., Ma, C., et al.: Frankenstein: Generating semantic-compositional 3d scenes in one tri-plane. arXiv preprint arXiv:2403.16210 (2024) [4](#)
59. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. In: NeurIPS (2021) [4](#), [14](#)
60. Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P.P., Szeliski, R., Barron, J.T., Mildenhall, B.: Baked sdf: Meshing neural sdfs for real-time view synthesis. In: SIGGRAPH (2023) [4](#)
61. Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., Lipman, Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. In: NeurIPS (2020) [4](#)
62. Yifan, W., Wu, S., Oztireli, C., Sorkine-Hornung, O.: Iso-points: Optimizing neural implicit surfaces with hybrid representations. In: CVPR (2020) [9](#)
63. Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A.: Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. In: NeurIPS (2022) [4](#), [7](#), [10](#), [14](#), [1](#), [3](#)
64. Yuan, Y., Li, X., Huang, Y., De Mello, S., Nagano, K., Kautz, J., Iqbal, U.: Gavatar: Animatable 3d gaussian avatars with implicit mesh learning. arXiv preprint arXiv:2312.11461 (2023) [4](#)
65. Zhi, S., Laidlow, T., Leutenegger, S., Davison, A.: In-place scene labelling and understanding with implicit scene representation. In: ICCV (2021) [3](#)
66. Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M.R., Pollefeys, M.: Nice-slam: Neural implicit scalable encoding for slam. In: CVPR (2022) [3](#)
67. Zwicker, M., Pfister, H., van Baar, J., Gross, M.: Ewa volume splatting. In: Proceedings Visualization, 2001. VIS '01. pp. 29–538 (2001) [4](#), [6](#)

–Appendix–

This document includes the supplementary information of our main document. We provide more details about our experiments in Sec. A, including implementation details, metric design, and dataset acquisition. In the following section, we provide more experimental results on Replica and Scannet in Sec. B and Sec. C respectively. Finally, we discuss the limitation of our framework in Sec. D.

A More details about experiments

We provide more details about the experiment, including the dataset preprocess, implementation details, and evaluation metric definition.

A.1 More Implementation details

Our implementation is based on Scaffold-GS [25]. Following [25], we set an opacity MLP, covariance MLP, and color MLP to decode the corresponding attributes for each spawn Gaussian. We set the anchor feature size as 32. As the main focus of our framework targets at surface reconstruction, we take the anchor feat as input for opacity MLP and covariance MLP to get the opacity, covariance, and scaling for each spawns Gaussian, while using the concatenated feature by anchor feature and view direction encoding for the color MLP to get the view-depend appearance, which is different with original design. We set the neural implicit network following the structure of Instant-NGP [32]. For the experiments conducted in Replica, we used the Poisson octree depth as 8 for surface extraction. We adopt the default train/test split for the Replica dataset. For ScanNet, we follow the experimental design of MonoSDF [63] which uses the entire dataset for training and evaluating the surface quality. We set the voxel size to 0.001 for the initial anchor point construction. We provide the training overview in Fig. A.1.

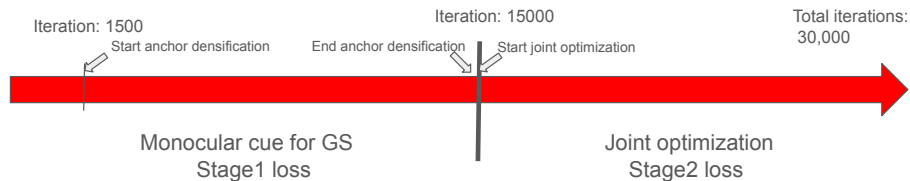


Fig. A.1: The training progress of our framework. We follow the Scaffold-GS to start anchor densification in 1500 iterations. Once the anchors stop growing or pruning, we add the neural implicit network for joint optimization to regulate the Gaussians attributes till the end of training.

We provide more details about the point sampling strategy used in our framework for the joint optimization of neural implicit representation. Firstly, we use the depth rendered by the Gaussians to un-project a 2D pixel into a 3D point. To guarantee the effectiveness of the point in the calculation of MLS, we propose to find the nearest Gaussian for this point and use the corresponding Gaussian to generate a new candidate with its mean and covariance. We will use this new point and its nearest L Gaussians to calculate the MLS function. As suggested in [24,54], we use the sampling points to construct a sphere with a radius of 0.01 and only use these Gaussians inside the sphere for the MLS function calculation.

A.2 The dataset for training

We use Replica and ScanNet for experiments. We use the ground camera poses for both datasets to train our model. To obtain the sparse point cloud used for 3D Gaussian Splatting, we use those ground truth camera poses and rerun COLMAP [41] to construct the initial point cloud. We also trained SuGaR with the ground truth camera pose and the initialized sparse point cloud for comparison.

We also normalized the camera pose to make sure the scene is located in a unit cube for the construction of instant-ngp [32] hash embedding. While this can also be implemented dynamically using the anchor point to set the normalized factor [8].

A.3 Metric for evaluating surface reconstruction

The definition of the evaluation metrics we used in the main document is given in Table. A.1.

Metric	Definition
Accuracy	$\text{mean}_{\mathbf{p} \in \mathbf{P}}(\min_{\mathbf{q} \in \mathbf{Q}} \ \mathbf{p} - \mathbf{q}\ _1)$
Completeness	$\text{mean}_{\mathbf{q} \in \mathbf{Q}}(\min_{\mathbf{p} \in \mathbf{P}} \ \mathbf{p} - \mathbf{q}\ _1)$
Chamfer-L1	$0.5 * (\text{Accuracy} + \text{Completeness})$
Precision	$\text{mean}_{\mathbf{p} \in \mathbf{P}}(\min_{\mathbf{q} \in \mathbf{Q}} \ \mathbf{p} - \mathbf{q}\ _1) < 0.05$
Recall	$\text{mean}_{\mathbf{q} \in \mathbf{Q}}(\min_{\mathbf{p} \in \mathbf{P}} \ \mathbf{p} - \mathbf{q}\ _1) < 0.05$
F-score	$2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall})$

Table A.1: Evaluation Metric Calculation. We provide the equation for computing the quantitative metric used in the experiment. Given the sampled point cloud from ground-truth \mathbf{P} and predicted result \mathbf{Q} , all the metrics can be calculated as shown above.

A.4 Details about RIMLS

The detailed equation of RIMLS is defined as follows:

$$F_{\text{RIMLS}}(\mathbf{x}) = \frac{\sum_{l=1}^L o_l G_l(\mathbf{x}) \phi(\|\mathbf{n}_l - \frac{\nabla F_{\text{MLP}}(\mathbf{x})}{\|\nabla F_{\text{MLP}}(\mathbf{x})\|}\|) \langle \mathbf{x} - \boldsymbol{\mu}_l, \mathbf{n}_l \rangle}{\sum_{l=1}^L o_l G_l(\mathbf{x}) \phi(\|\mathbf{n}_l - \frac{\nabla F_{\text{MLP}}(\mathbf{x})}{\|\nabla F_{\text{MLP}}(\mathbf{x})\|}\|)}. \quad (12)$$

As mentioned in the main document, the ϕ is defined as a 1-D Gaussian kernel with a variance of σ_n^2 . The σ_n is set as 0.05 in our implementation.

B More results on Replica

To further investigate the benefits of joint optimization, we conducted additional experiments focused on using an implicit network to fit normal-augmented Gaussians.

For this purpose, we targeted Gaussians derived from Scaffold+N&D [25] and employed an identical implicit network structure for fitting. The training process utilized the same loss functions as in our joint optimization scheme, $\mathcal{L}_{\text{joint}}$. The critical variable in our experiment was the application of joint optimization for updating the Gaussians. We visualized the resulting meshes by the implicit network through Marching Cubes, as depicted in Fig. B.1. Our findings reveal that without joint optimization, the implicit function tends to fit more high-frequency noise as the number of iterations increases, leading to a less smooth surface. Conversely, the use of joint optimization yields a smoother, more accurate surface by regulating the Gaussians’ positions and orientations. This regulation not only mitigates the tendency to fit low-frequency details early on but also refines the Gaussians’ attributes for better surface alignment. Notably, meshes generated through our method exhibited some floating elements in empty spaces, attributed to isolated Gaussians. However, by employing Poisson reconstruction for surface construction from the final Gaussians, our approach demonstrates robustness against such noisy and outlier Gaussians, ensuring cleaner, more coherent mesh outputs.

C More results on ScanNet

We provide more comprehensive results on ScanNet [10] in Tab. C.1. Our approach achieves a comparable performance with Manhattan-SDF [16] and MonoSDF (Grids) [63] in all metrics, which demonstrates a strong potential ability of 3DGS to produce a high-quality surface mesh.

D Discussion about Limitations

While our method achieves superior quality in many aspects, our analysis reveals a noticeable gap when compared to the current state-of-the-art in surface

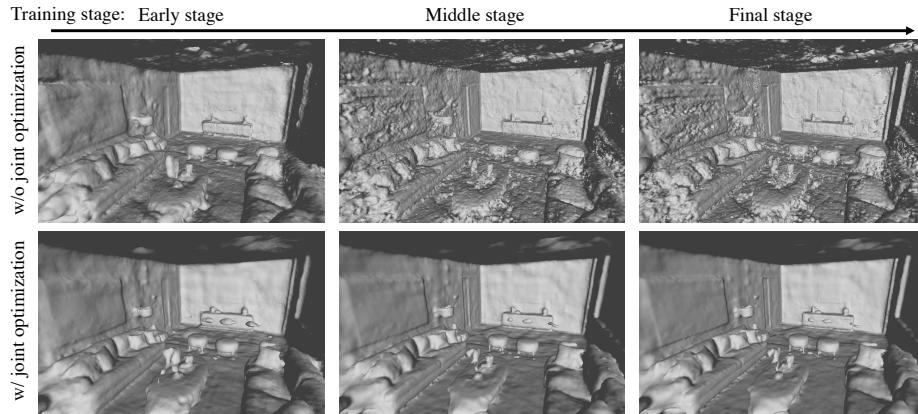


Fig. B.1: Comparison of the mesh produced by performing Marching Cube from the implicit network. The top row shows the mesh produced by the neural implicit network without joint optimization of the Gaussians and the bottom row depicts the mesh from the implicit network with joint optimization. We noticed that without joint optimization, the implicit function will fit the high-frequency noise as the training goes on. The joint optimization regulates the orientation and position of Gaussians to obtain a better surface alignment Gaussian Splatting field.

reconstruction. A promising direction for future work is the incorporation of appearance guidance, which can enable a more precise capture of detailed geometry, thereby lessening our method’s dependence on pre-trained models. Furthermore, revisiting and potentially revising the Gaussian assumption inherent in our 3D Gaussian Splatting (3DGS) approach could yield more accurate models for density estimation, enhancing surface reconstruction fidelity. These areas represent valuable opportunities for future research, signaling our commitment to pushing the boundaries of what is achievable in surface reconstruction quality.

	Accuracy ↓	Completeness ↓	Chamfer-L1 ↓	Precision ↑	Recall ↑	F-Score ↑
UNISURF [33]	0.554	0.164	0.359	0.212	0.362	0.267
Neus [51]	0.179	0.208	0.194	0.313	0.275	0.291
VolSDF [59]	0.414	0.120	0.267	0.321	0.394	0.346
Manhattan-SDF [16]	0.072	0.068	0.070	0.621	0.586	0.602
MonoSDF (Grids) [63]	0.072	0.057	0.064	0.660	0.601	0.626
MonoSDF (MLP) [63]	0.035	0.048	0.042	0.799	0.681	0.733
SuGaR [15](density)	0.398	0.170	0.284	0.189	0.255	0.217
SuGaR [15](SDF)	0.328	0.211	0.269	0.179	0.189	0.184
Ours	0.067	0.069	0.068	0.604	0.594	0.599

Table C.1: The quantitative results of the scene reconstruction on ScanNet.