# High-quality Surface Reconstruction using Gaussian Surfels


Pinxuan Dai*
daipinxuan@zju.edu.cn
State Key Lab of CAD&CG, Zhejiang University
China

Jiamin Xu*
superxjm@yeah.net
Hangzhou Dianzi University
China

Wenxiang Xie
zju_xwx@zju.edu.cn
State Key Lab of CAD&CG, Zhejiang University
China

Xinguo Liu
xgliu@cad.zju.edu.cn
State Key Lab of CAD&CG, Zhejiang University
China

Huamin Wang
wanghmin@gmail.com
Style3D Research
United States of America

Weiwei Xu†
xww@cad.zju.edu.cn
State Key Lab of CAD&CG, Zhejiang University
China


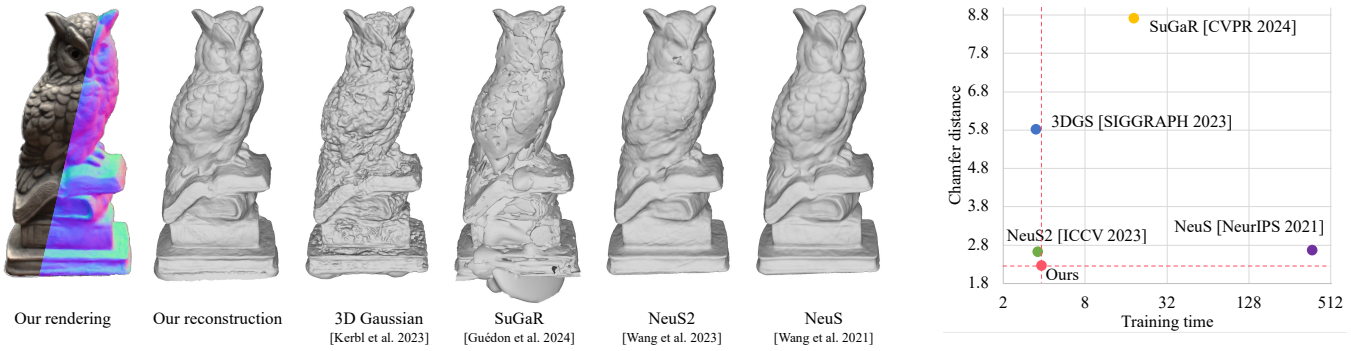


**Figure 1: Left: Our method achieves high-quality surface reconstruction result on par with NeuS [Wang et al. 2021] for this owl object (DTU-#122). Right: The 2D plot shows that our method achieves a good balance between training time and the quality of surface reconstructions measured in the Chamfer distance. The statistics are collected from the scenes selected from BlendedMVS [Yao et al. 2020] dataset.**


## ABSTRACT

We propose a novel point-based representation, Gaussian surfels, to combine the advantages of the flexible optimization procedure in 3D Gaussian points and the surface alignment property of surfels. This is achieved by directly setting the z-scale of 3D Gaussian points to 0, effectively flattening the original 3D ellipsoid into a 2D ellipse. Such a design provides clear guidance to the optimizer. By treating the local z-axis as the normal direction, it greatly improves optimization stability and surface alignment. While the derivatives to the local z-axis computed from the covariance matrix are zero in this setting, we design a self-supervised normal-depth consistency loss to remedy this issue. Monocular normal priors and foreground masks are incorporated to enhance the reconstruction quality, mitigating issues related to highlights and background. We propose a volumetric cutting method to aggregate the information of Gaussian surfels so as to remove erroneous points in depth maps generated by alpha blending. Finally, we apply screened Poisson reconstruction method to the fused depth maps to extract the surface mesh. Experimental results show that our method demonstrates superior performance in surface reconstruction compared to state-of-the-art neural volume rendering and point-based rendering methods.



## CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**.

## KEYWORDS

3D Surface Reconstruction, Gaussian Surfels, Depth-normal Consistency

**ACM Reference Format:**
Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. 2024. High-quality Surface Reconstruction using Gaussian Surfels. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657441


*Joint first authors
†Corresponding author



*SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA*



ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657441

# 1 INTRODUCTION

Recently, 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] has gained widespread popularity for reconstructing and rendering 3D scenes. Different from neural implicit representations [Mildenhall et al. 2021; Wang et al. 2021; Yariv et al. 2021], 3DGS represents the appearance and geometry through a set of explicit and topology-free Gaussian points. These Gaussian points can be dynamically added and removed during optimization, ensuring the convergence of the optimization process to cover the entire surface and represent high-frequency details. Moreover, 3DGS utilizes GPU/CUDA-based rasterization with closed-form integration [Zwicker et al. 2002] during rendering, eliminating the need for time-consuming ray-based point sampling in volume rendering. This results in a significant reduction in training and rendering time, enabling a training time of less than 10 minutes and facilitating real-time rendering.

Despite the advantages of the 3DGS representation, it struggles to generate high-quality geometric reconstructions. This limitation stems from three aspects: (1) *Non-zero thickness*: 3D Gaussian points, resembling ellipsoids, have non-zero thickness along each axis, hindering their close alignment with the actual surfaces. (2) *Ambiguity in normal direction*: there exists ambiguity in determining the normal for each 3D Gaussian, i.e. the normal axis can change among different scale directions during optimization. This ambiguity can result in inaccuracies when reconstructing geometries with fine details. (3) *Modeling sharp surface edges*: the alpha blending process may introduce bias to a reconstructed surface edge, which will happen when Gaussian points with extent beyond the surface edge or far from the edge are occasionally involved during the blending. Recent methods such as SuGaR [Guédon and Lepetit 2023] and NeuSG [Chen et al. 2023] introduce a regularizer to minimize the smallest component of the scaling factor along each axis, which can alleviate the first thickness problem. However, the quality of reconstructed surface is still unsatisfactory. Using Gaussian points to reconstruct surface with fine details remains challenging.

In this paper, we propose a novel representation, Gaussian surfels, to combine the advantages of the flexible optimization procedure in 3DGS and the surface alignment property of surfels [Pfister et al. 2000]. This significantly improves the quality of the reconstructed geometry. Gaussian surfels are achieved by directly setting the z-scale of the scale matrix in 3D Gaussian points to 0, effectively flattening the original 3D ellipsoid into a 2D ellipse. Compared with the regularization methods in [Chen et al. 2023; Guédon and Lepetit 2023], our representation avoids the need to determine a minimal scale that might change during optimization. It provides clear guidance to the optimizer to treat the local z-axis as the normal direction, greatly improving optimization stability and surface alignment.

The technical challenge arising from flattening 3D Gaussian points in this manner is that the derivatives that are computed from the covariance matrix with respect to the local z-axis will be zero. As a result, the photometric loss itself can not affect the local z-axis during optimization. Therefore, we design a self-supervised normal-depth consistency loss to remedy this problem. It requires local z-axis to be close to the normal computed from the depth map rendered using Gaussian splatting. Moreover, although the truncation threshold in 3D Gaussian points is carefully selected to guarantee high-quality rendering results ($\frac{1}{255}$ in 3DGS), we found it is too small to prevent the generation of blurred sharp edges or floating geometries in the surface reconstruction results. These artifacts often occur when a ray passes through a front surface near its edge, but actually, the ray's first intersection with scene geometry is at a surface behind the front surface. We thus propose a volumetric cutting method to determine whether a voxel should be cut off or not according to its distance from the Gaussian surfels, which further improves geometric quality.

Our high-quality surface reconstruction results using Gaussian surfels are made possible by the following technical contributions:

- Introducing a novel point-based representation, Gaussian surfels, to resolve the inherent normal ambiguity of 3DGS and achieve a close alignment with the actual surface. Combined with our volumetric cutting method, the quality of surface reconstruction is significantly enhanced.
- Proposing a self-supervised normal-depth consistency regularizer, along with the photometric loss, to guide the Gaussian surfels in moving and rotating in a manner that closely conforms to the surfaces of the object. We also incorporate monocular estimated normals as a prior to address shape-radiance ambiguity [Zhang et al. 2020] in regions with specular reflections.
- Compared with state-of-the-art neural volume and point-based rendering methods, our method achieves a good balance between reconstruction quality and training speed (Fig. 1).

# 2 RELATED WORK

The target of multi-view surface reconstruction methods is to create a geometric representation of an object or a scene using multi-view images. This is accomplished through classic multi-view stereo (MVS) techniques [Furukawa et al. 2015], which can be broadly classified as voxel grids optimization [Seitz and Dyer 1999; Sinha et al. 2007], feature point growing [Furukawa and Ponce 2009; Wu et al. 2010], or depth-map estimation and merging [Schönberger et al. 2016]. These methods rely on photometric consistency across views, making it challenging to accurately capture complete geometric representations due to the ambiguities in the correspondence.

Recently, neural rendering [Tewari et al. 2020] has demonstrated impressive capabilities in view synthesis and surface reconstruction. By directly minimizing the per-pixel difference between the image and rendering results, it achieves detailed surface reconstruction [Wang et al. 2021]. Additionally, it can adapt to complex materials through a sophisticated rendering process [Zhang et al. 2021, 2022a].

## 2.1 Neural volume rendering

The landmark work in neural volume rendering is NeRF [Mildenhall et al. 2021], which employed a differentiable volumetric rendering technique to reconstruct a neural scene representation, achieving impressive photorealistic view synthesis with view-dependent effects. To accelerate its optimization, subsequent research replaces the neural scene representation with explicit or hybrid scene representations, such as voxel grid [Fridovich-Keil et al. 2022; Sun et al. 2022b], low-rank tensors [Chen et al. 2022], tri-planes [Chan et al. 2022; Reiser et al. 2023], and multi-resolution hash maps [Müller et al. 2022].

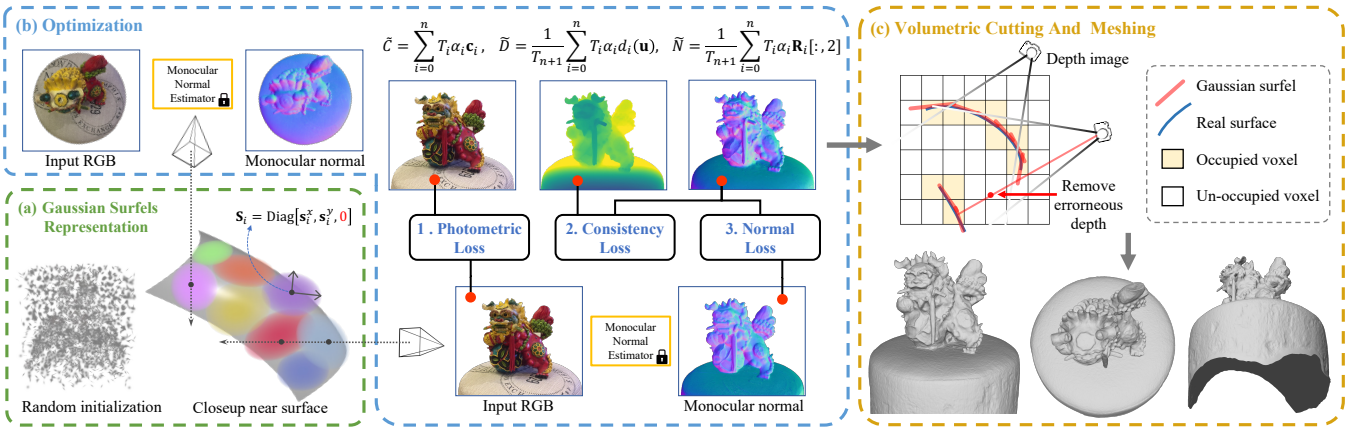


**Figure 2: The pipeline of our method. Our method involves the following steps: (a) Starting with random initialization, our method represents the surface as a set of Gaussian surfels, each with learnable position, rotation, color, opacity, and covariance; (b) Optimize the Gaussian surfels through multi-view photometric loss, depth-normal consistency loss, and normal prior loss; (c) Perform volumetric cutting on rendered depth maps, then apply Poisson meshing from rendered depth and normal to extract a high-quality mesh. Our method can automatically obtain an open surface reconstruction result.**

However, NeRF and its variants extract isosurfaces based on heuristic thresholding of density values, which can introduce high-frequency noise into the reconstructed surfaces. To enhance the quality of reconstruction, several studies have suggested using occupancy grids [Oechsle et al. 2021] or signed distance functions (SDFs) [Wang et al. 2021; Yariv et al. 2021, 2020] represented by coordinate-based multi-layer perceptron (MLP) networks. These methods provide better-defined 3D surfaces compared to volume density fields. Follow-up works have further improved reconstruction by utilizing auxiliary information. For example, NeuralWarp [Darmon et al. 2022] employs patch warping with co-visibility information for surface optimization. Additionally, monocular depth and normals [Yu et al. 2022], sparse point clouds [Fu et al. 2022; Zhang et al. 2022b], semantic segmentation [Guo et al. 2022; Sun et al. 2022a], and geometric priors [Long et al. 2022] have been used to guide and regularize 3D reconstruction.

Alternatively, some methods incorporate explicit voxel grids [Wu et al. 2022] or multi-resolution hash encodings with a CUDA-based MLP implementation to expedite surface reconstruction [Li et al. 2023; Wang et al. 2023; Zhao et al. 2022]. For calculating the derivatives of Eikonal loss, Instant-NSR [Zhao et al. 2022] approximates these derivatives using finite differences, while NeuS2 [Wang et al. 2023] proposes a precise and efficient formulation of second-order derivatives tailored to MLPs. Neuralangelo [Li et al. 2023] also utilizes numerical gradients for computing higher-order derivatives as a smoothing operation. Additionally, it involves coarse-to-fine optimization on the hash grids, controlling different levels of detail.

## 2.2 Neural point-based rendering.

Point-based rendering has also been employed for neural rendering. Unlike volumetric representation, point-based methods represent geometry through unstructured samples in a topology-free manner [Kobbelt and Botsch 2004]. As directly rendering point samples suffers from holes and discontinuities, surface splatting has been developed by splatting point primitives with an extent, such as ellipsoids or surfels with circular or elliptic discs [Botsch et al. 2005; Habbecke and Kobbelt 2007; Pfister et al. 2000; Ren et al. 2002; Zwicker et al. 2002], and the surfel representation has been applied to depth fusion in 3D reconstruction using depth cameras [Weise et al. 2009; Xu et al. 2018]. For point-based surface reconstruction, DSS [Yifan et al. 2019] employs opaque circles to represent surfaces, optimizing them through differentiable surface splatting. This approach confines its gradients to few closest ellipses. Similarly, PBNR [Kopanas et al. 2021] uses an anisotropic 2D Gaussian associated with each pixel to capture local geometry. Due to the absence of adjustable opacity, both of these methods rely on a relatively precise geometric initialization. Point-based rendering can also be combined with neural features and decoder networks to enhance rendering quality [Aliev et al. 2020; Rückert et al. 2022].

Alternatively, recent methods have been developed to approximate volume rendering via conventional alpha-blending on sorted splats. Pulsar [Lassner and Zollhofer 2021] and Neural Catacaustics [Kopanas et al. 2022] represent surfaces as a set of isotropic 3D spheres or 2D Gaussian discs. 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] introduces a promising approach to modeling 3D scenes, achieving fast reconstruction and real-time rendering speeds, as well as enhanced quality. It represents complex scenes as a combination of numerous 3D Gaussians with position, opacity, anisotropic covariance, and spherical harmonic coefficients. SuGaR [Guédon and Lepetit 2023] further proposes a method allowing precise and fast mesh extraction from 3D Gaussian Splatting by adding a regularization term that encourages the Gaussians to align well with the surface. Similarly, NeuSG [Chen et al. 2023] incorporates scale regularization to promote narrow 3D Gaussian ellipsoids. However, despite these efforts, achieving a close fit between anisotropic 3D Gaussians and surfaces remains a challenging task.

In contrast, our method takes a different approach by utilizing Gaussian surfels with opacity and anisotropic covariance to achieve a better alignment with the surface. Unlike 3DGS [Kerbl et al. 2023]

and SuGaR [Guédon and Lepetit 2023] which emphasize realistic rendering, our primary objective is to achieve super-fast surface reconstruction while preserving fine details.

# 3 METHOD

## 3.1 Overview

The goal of our method is to combine the flexible optimization and rendering scheme in 3DGS and the surface alignment property of surfels in a way such that surface meshes with fine details can be efficiently reconstructed. As outlined in Fig. 2, the proposed method takes a set of posed RGB images as input, where $\mathbf{I}_k$ denotes the captured $k$-th image. As to the output, the method produces a set of Gaussian surfels, represented as ellipses with anisotropic Gaussian kernels, opacities, and view-dependent colors represented using spherical harmonics (Sec. 3.2).

We optimize the Gaussian surfels to minimize the photometric difference between input RGB images and rendering results at captured views. To make the optimization controllable, we introduce a set of regularization losses to enforce surface smoothness and depth-normal consistency (Sec. 3.3). After completing the optimization, we render multi-view depth maps and normal maps and fuse them through screened Poison reconstruction [Kazhdan and Hoppe 2013] to extract a high-quality global mesh. Before meshing, a volumetric cutting procedure is performed to reduce erroneous depth values caused by alpha blending for pixels at surface boundaries (see 3.4).

## 3.2 Gaussian Surfels

In this section, we introduce the representation of our Gaussian surfels, which comprise a set of unstructured Gaussian kernels $\{\mathbf{x}_i, \mathbf{r}_i, \mathbf{s}_i, o_i, C_i\}_{i\in\mathcal{P}}$, where $i$ is the index of each Gaussian kernel, $\mathbf{x}_i \in \mathbb{R}^3$ denotes the position of Gaussian kernel's center, $\mathbf{r}_i \in \mathbb{R}^4$ is its rotation (orientation) represented by a quaternion, $o_i \in \mathbb{R}$ is the opacity, and $C_i \in \mathbb{R}^k$ is spherical harmonic coefficients of each Gaussian. The symbol $\mathbf{s}_i \in \mathbb{R}^3$ represents the scaling factors for each of the two local axes of a surfel. Thus, the 3D Gaussian distribution can be represented as:

$$G\left(\mathbf{x}; \mathbf{x}_i, \Sigma_i\right) = \exp\left\{-0.5\left(\mathbf{x}-\mathbf{x}_i\right)^\top \Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\right\}, \tag{1}$$

where $\Sigma_i$ is the covariance matrix expressed as the product of a scaling matrix $\mathbf{S}_i$ and a rotation matrix $\mathbf{R}(\mathbf{r}_i)$, which is a $3\times 3$ rotation matrix represented by a quaternion $\mathbf{r}_i$.

We set $\mathbf{s}_i = [\mathbf{s}_i^x, \mathbf{s}_i^y, 0]^\top$ to flatten the 3D Gaussians [Kerbl et al. 2023]. Therefore, the corresponding $\Sigma_i$ is changed to:

$$\Sigma_i = \mathbf{R}(\mathbf{r}_i)\mathbf{S}_i\mathbf{S}_i^\top\mathbf{R}(\mathbf{r}_i)^\top = \mathbf{R}(\mathbf{r}_i)\mathrm{Diag}\left[\left(\mathbf{s}_i^x\right)^2, \left(\mathbf{s}_i^y\right)^2, 0\right]\mathbf{R}(\mathbf{r}_i)^\top, \tag{2}$$

where Diag$[\cdot]$ indicates a diagonal matrix with diagonal entries in $[]$. In this Gaussian surfel representation, each Gaussian is truncated as a 2D ellipse and the normal for each Gaussian kernel can be directly computed as $\mathbf{n}_i = \mathbf{R}(\mathbf{r}_i)[:, 2]$, where the $[\cdot]$ operator follows the slicing syntax of a multi-dimensional array in NumPy [Kingma and Ba 2014]. Our method supports optimizing the normal direction of each ellipse to achieve the alignment of Gaussian surfels with the actual surface.

*Differentiable Gaussian splatting.* For novel view rendering, the Gaussian splatting procedure remains consistent with that of 3DGS. Specifically, during rendering, the color of each pixel $\mathbf{u}$ is calculated via alpha-blending of all the nearby Gaussian kernels:

$$\tilde{C} = \sum_{i=0}^{n} T_i\alpha_i\mathbf{c}_i, \quad T_i = \prod_{j=0}^{i-1}(1-\alpha_j), \quad \alpha_i = G'(\mathbf{u}; \mathbf{u}_i, \Sigma_i')o_i, \tag{3}$$

where $\alpha_i$ represents alpha-blending weight, which is the product of opacity and the Gaussian weight specified in Eq. 1. The view dependent color $\mathbf{c}_i$ is computed via spherical harmonics. To enhance the rendering efficiency, the 3D Gaussian in Eq. 2 are reparameterized in 2D ray space [Zwicker et al. 2002] as $G'$:

$$G'\left(\mathbf{u}; \mathbf{u}_i, \Sigma_i'\right) = \exp\left\{-0.5\left(\mathbf{u}-\mathbf{u}_i\right)^\top \Sigma_i'^{-1}\left(\mathbf{u}-\mathbf{u}_i\right)\right\}, \tag{4}$$

$$\Sigma_i' = \left(\mathbf{J}_k\mathbf{W}_k\Sigma_i\mathbf{W}_k^\top\mathbf{J}_k^\top\right)[:2, :2], \tag{5}$$

where $\mathbf{W}_k$ is a viewing transformation matrix for input image $k$, $\mathbf{J}_k$ is the affine approximation of the projective transformation. $\Sigma'$ represents the transformed covariance matrix in image coordinates.

Similarly, the depth $\tilde{D}$ and normal $\tilde{N}$ for each pixel can also be calculated via Gaussian splatting and alpha-blending:

$$\tilde{N} = \frac{1}{1-T_{n+1}}\sum_{i=0}^{n} T_i\alpha_i\mathbf{R}_i[:, 2], \quad \tilde{D} = \frac{1}{1-T_{n+1}}\sum_{i=0}^{n} T_i\alpha_i d_i(\mathbf{u}). \tag{6}$$

We utilize $1/(1-T_{n+1})$ to normalize the blending weight $T_i\alpha_i$. Unlike adding a background color during color rendering, we found this normalization approach to be more suitable for rendering depth and normal maps.

For depth rendering, directly blending the depth of the center position $d_i(\mathbf{u}_i)$ of each Gaussian kernel is inaccurate because it neglects the slope of the 2D ellipse, as illustrated in Fig. 3. For instance, when dealing with Gaussian surfels aligned with a slanted plane, such depth rendering results will not be consistent with surface normals. To this end, in our implementation, the depth of pixel $\mathbf{u}$ for each Gaussian kernel $i$ is computed by calculating the intersection of the ray cast through pixel $\mathbf{u}$ with the Gaussian ellipse during splatting. It can be simplified via local Taylor expansion as:

$$d_i(\mathbf{u}) = d_i(\mathbf{u}_i) + (\mathbf{W}_k\mathbf{R}_i)[2, :]\mathbf{J}_{pr}^{-1}(\mathbf{u}-\mathbf{u}_i), \tag{7}$$

where $\mathbf{J}_{pr}^{-1}$ is the Jacobian of inverse mapping a pixel in the image space onto the tangent plane of Gaussain surfel as in [Zwicker et al. 2001], $(\mathbf{W}_k\mathbf{R}_i)$ transforms the rotation matrix of a Gaussian surfel to the camera space.

## 3.3 Optimization

To guide the optimization of explicit Gaussian surfels, our total loss $\mathcal{L}$ consists of five components: photometric loss $\mathcal{L}_\mathrm{p}$, normal-prior loss $\mathcal{L}_\mathrm{n}$, opacity loss $\mathcal{L}_\mathrm{o}$, depth-normal consistency loss $\mathcal{L}_\mathrm{c}$. Similar to [Wang et al. 2021], we also introduce a mask loss $\mathcal{L}_\mathrm{m}$, which computes binary cross-entropy between $\sum_{i=0}^{n} T_i\alpha_i$ and the binary segmentation mask. Thus we have:

$$\mathcal{L} = \mathcal{L}_\mathrm{p} + \mathcal{L}_\mathrm{n} + \lambda_\mathrm{o}\mathcal{L}_\mathrm{o} + \lambda_\mathrm{c}\mathcal{L}_\mathrm{c} + \lambda_\mathrm{m}\mathcal{L}_\mathrm{m}, \tag{8}$$

where the trade-off weights $\lambda_\mathrm{o}$, $\lambda_\mathrm{c}$, and $\lambda_\mathrm{m}$ balance the loss terms.

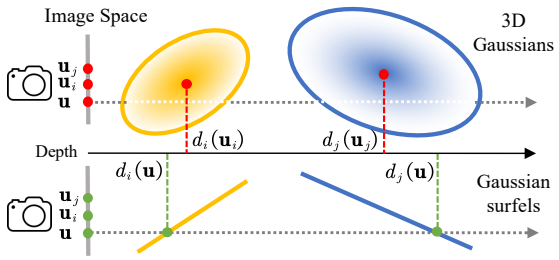


**Figure 3: *The first row*: the intersection of a ray with a 3D Gaussian point is challenging to calculate precisely. As a result, methods such as SuGaR [Guédon and Lepetit 2023] approximate the depth of intersection with the depth of the Gaussian's center point, which can introduce errors. *The second row*: the intersection of a ray with our Gaussian surfel can be calculated precisely, as well as its depth.**

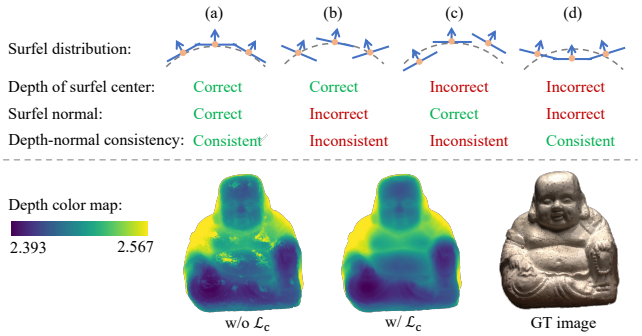


**Figure 4: Depth-normal consistency. *Top*: The consistency of surfel depth and normal during optimization. *Bottom*: Rendered depth maps with and without consistency loss, alongside the reference image.**

*Photometric loss* $\mathcal{L}_p$. The photometric loss is same as in 3DGS. It combines an $L_1$ term and a D-SSIM term to minimize the difference between a rendered image $\tilde{\mathbf{I}}$ and its matched input image $\mathbf{I}$:

$$\mathcal{L}_p = 0.8 \cdot L_1(\tilde{\mathbf{I}}, \mathbf{I}) + 0.2 \cdot L_{DSSIM} \cdot (\tilde{\mathbf{I}}, \mathbf{I}), \tag{9}$$

*Depth-normal consistency loss* $\mathcal{L}_c$. This term enforces consistency between the rendered depth $\tilde{\mathbf{D}}$ and the rendered normal $\tilde{\mathbf{N}}$:

$$\mathcal{L}_c = 1 - \tilde{\mathbf{N}} \cdot N(V(\tilde{\mathbf{D}})). \tag{10}$$

where $V(\cdot)$ transforms each pixel and its depth to a 3D point, and $N(\cdot)$ calculates the normal from neighboring points using the cross product. The depth-normal consistency loss plays a crucial role in our optimization process, particularly in resolving the gradient vanishing problem for each Gaussian surfel.

Moreover, we have found that it can also aid in resolving the ambiguity between rendered depth and normal, where Gaussian-splatted normals may exhibit accuracy while depths do not, and vice versa. As illustrated in Fig. 4: when the center depth is correct but the normal is not (top b), the rendered depth can help correct the direction of the Gaussian ellipse; when the normal is correct but the depth is not (top c), $\mathcal{L}_c$ can be interpreted as normal-aware depth smoothing. Enforcing bidirectional consistency between rendered depth and normal results in an improved quality of reconstruction in both cases. The self-supervised consistency loss alone does not guarantee correct surface (top d), while combining with other loss terms in this section achieves consistent and correct results (top a).

*Normal-prior loss* $\mathcal{L}_n$. This term acts as a prior-based regularizer that improve optimization stability, especially in areas with highlights, where photometric loss may result in wrong surface points. We use normal maps $\hat{N}$ from a pretrained monocular deep neural network from Omnidata [Eftekhar et al. 2021]. Additionally, we introduce an $L_1$ loss to minimize the gradient of the rendered normal, denoted as $\nabla\tilde{N}$, thereby regularizing the curvature of the surface:

$$\mathcal{L}_n = 0.04 \cdot (1 - \tilde{\mathbf{N}} \cdot \hat{\mathbf{N}}) + 0.005 \cdot L_1(\nabla\tilde{\mathbf{N}}, \mathbf{0}), \tag{11}$$

*Opacity loss* $\mathcal{L}_o$. This opacity loss promotes non-transparent surfaces by encouraging each Gaussian's opacity to be either near zero or near one, where $o_i$ is parameterized with a sigmoid function. This contributes to the overall quality of the reconstruction:

$$\mathcal{L}_o = \exp\left(-(o_i - 0.5)^2 / 0.05\right). \tag{12}$$

According to Equation 2, the gradient of the covariance matrix with respect to the 3rd column of $\mathbf{R}_i$ (normal), is equal to zero. Consequently, following the chain rule, the photometric loss $\mathcal{L}_p$ has no gradient with respect to the normal of each Gaussian surfel. As $\mathbf{R}_i$ is a rotation matrix in $\mathbb{SO}(3)$, the normal will still be modified according to the first two axes of $\mathbf{R}_i$. However, this kind of indirect modification can lead to errors during optimization. This observation inspired us to incorporate the depth-normal consistency loss $\mathcal{L}_c$, which rectifies the normal of each Gaussian surfel using the gradient obtained from the rendered depth.

## 3.4 Gaussian Point Cutting and Meshing

We fuse the rendered depth and normal maps at captured views and then apply screened Poisson reconstruction [Kazhdan and Hoppe 2013] with a tree depth of 10 to obtain the final surface mesh. Compared with directly applying Poisson reconstruction to Gaussian centers, this greatly increases the point density and improves the quality of surface details.

However, the rendered depth map still contains errors, especially near depth discontinuities. As depicted in Fig. 5, if ellipses extend beyond the true surface boundary and the associated weights cannot rapidly decay to zero, the rendered depth of the background surfaces will be influenced by the alpha of the foreground Gaussian points, resulting in a depth that is in front of the true background surface. Due to the complex distribution of Gaussian surfels, we found it was difficult to remove outliers along each ray by discarding surfels far from median or alpha-weighted mean (Fig. 6 (c)). Instead, we implement volumetric cutting according to the aggregated alpha values from each Gaussian surfel.

*Volumetric cutting.* The idea is to mark those voxels far from Gaussian surfels as un-occupied, i.e., cutting them from the grid. Thus, the wrong 3D point in red illustrated in Fig. 5 can be removed because it is inside a un-occupied voxel. More precisely, we first construct a $512^3$ voxel grid within the bounding box. Next,

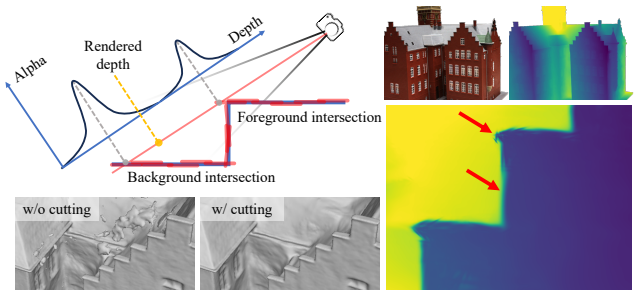


**Figure 5: An example of error in the rendered depth. The complex distribution of Gaussian surfels after optimization makes it difficult to remove outlier Gaussian points along each ray by discarding points far from median or alpha-weighted mean.**

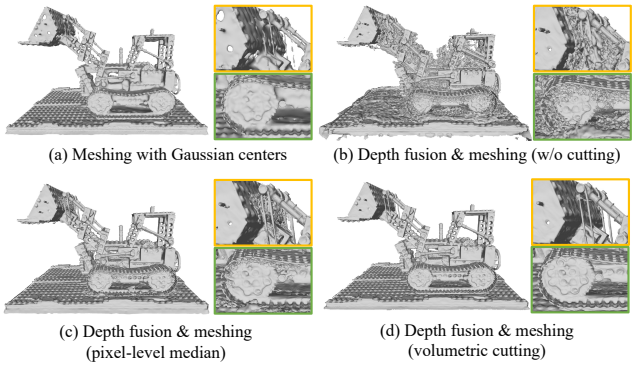


**Figure 6: Comparisons on cutting strategies. Meshing is achieved via screened Poisson reconstruction [Kazhdan and Hoppe 2013].**

we traverse all the Gaussian ellipses, calculate their intersection with the surrounding voxels, and accumulate the weighted opacity $G(\mathbf{x}; \mathbf{x}_i, \Sigma_i) \cdot o_i$ to the corresponding voxels. To reduce computational cost, we approximate the integration of Gaussian weights and opacities within the intersection area using the weighted opacity of the voxel center. If a voxel has a low accumulated weighted opacity, lower than $\lambda = 1$ in our experiments, indicating a large distance from the foreground or background surfaces, we prune these voxels as well as the 3D points in them computed from the depth. As shown in Fig. 6, compared with outlier removal using median depth at each pixel, our volumetric cutting exhibits better quality and computational efficiency.

## 3.5 Implementation details

*Initialization.* We support the use of sparse points computed via the Structure from Motion (SfM) as the initialization (Fig. 10). We found this could accelerate the decrease of loss function values in the first few steps but overall did not significantly improve the convergence rate. Thus, we initialize our Gaussian surfels with random positions and rotations inside the approximated bounding box of the target object in all our experiments.

*Optimization.* Our model is trained on a GPU server with an i9-14900K CPU and a RTX 4090 GPU, using the Adam optimizer [Kingma and Ba 2014], PyTorch 1.12.0 [Paszke et al. 2019], and CUDA 11.8. Each data batch contains all pixels of an image at every iteration. We do the ADMM training procedure for 15k iterations starting at lower resolutions for a warm-up. For the training loss, we assign $\lambda_o = 0.01$ and $\lambda_m = 1$. $\lambda_c$ is linearly increased from 0 to 0.1. The learning rate associated with the surfels position $\mathbf{x}_i$ is set to 1.6e-4, and decays exponentially to 1.6e-6. The learning rates for $\mathbf{r}_i, \mathbf{s}_i, o_i, C_i$ are set to 1.0e-3, 5.0e-3, 5.0e-2, 2.5e-3, respectively. As the photometric loss does not have a gradient with respect to $\mathbf{R}(\mathbf{r}_i)[:,2]$, we scale the gradient $\partial\tilde{\mathbf{N}}/\partial\mathbf{R}[:,2]$ by a factor of 10 in the derivative chain to balance the gradients along each axis in $\mathbf{R}(\mathbf{r}_i)$, and guide the Gaussian ellipse to rotate to the correct direction.

We utilize adaptive point splitting, cloning, and pruning techniques akin to 3DGS. A novel aspect of our implementation involves pruning points that do not receive gradients every N iterations, where N is the number of images. This pruning is motivated by the fact that points lacking gradients correspond to those that are invisible from all views, such as noisy points within the object.

## 4 EXPERIMENTS

*Baseline.* We compare our method with 1) NeuS [Wang et al. 2021], INSR [Zhao et al. 2022] and NeuS2 [Wang et al. 2023], which are implicit surface reconstructions by neural volume rendering, 2) 3DGS [Kerbl et al. 2023] and SuGaR [Guédon and Lepetit 2023] which are 3D Gaussian points-based surface reconstructions. All comparisons use original author's implementations and hyperparameters.

*Evaluation metrics and datasets.* To evaluate our method, we report both surface accuracy as Chamfer distance and render fidelity as Peak Signal-to-Noise Ratio (PSNR) on the DTU [Jensen et al. 2014] and BlendedMVS [Yao et al. 2020] datasets. We follow NeuS2 to leave 7 $\sim$ 8 images for testing on DTU, and use all images for training on BlendedMVS. The Chamfer distance measures the average of accuracy and completeness, i.e. the Chamfer distance of prediction to reference point cloud and vise versa.

*Comparisons.* We first benchmark our method on the DTU dataset [Jensen et al. 2014] both quantitatively (Table 1) and qualitatively (Fig. 8). The dataset encompasses laboratory-captured scenes, each encompassing 49 or 64 images with resolution $1600 \times 1200$, and we choose the same set of scenes selected by IDR [Yariv et al. 2020] that contain 15 objects with manually annotated object masks. We further tested on 18 challenging scenes from the low-res set of the BlendedMVS dataset [Yao et al. 2020], where each scene comprises 24 to 143 images at $768 \times 576$ pixels with masks provided. The evaluation result is obtained using the DTU evaluation code.

The statistics of Chamfer distances in these two tests are shown in Tables 1 and 2. Our method significantly outperforms 3DGS and SuGaR on the DTU and the BlendedMVS datasets. Compared with NeuS2, our results show a larger Chamfer distance on DTU. We hypothesize that it is due to the bias inherent in per-view depth calculation using alpha blending. Despite the higher Chamfer distance, our results often have less noise than NeuS2 and more details than NeuS, as illustrated in Fig. 8. In addition, our method can be

**Table 1: Geometry quality comparison on the DTU dataset. We report the Chamfer distance compared with baselines. Mean scores of Chamfer distance (mm) and training times (minute) are included on the bottom. Best results are highlighted as 1st, 2nd, and 3rd.**

| Methods | Ours | 3DGS | SuGaR | NeuS | NeuS2 | INSR |
|---|---|---|---|---|---|---|
| 24 | 0.66 | 2.89 | 1.85 | 0.83 | 0.56 | 2.86 |
| 37 | 0.93 | 2.65 | 1.19 | 0.98 | 0.76 | 2.81 |
| 40 | 0.54 | 2.66 | 1.91 | 0.56 | 0.49 | 2.09 |
| 55 | 0.41 | 2.31 | 1.64 | 0.37 | 0.37 | 0.81 |
| 63 | 1.06 | 3.55 | 2.76 | 1.13 | 0.92 | 1.65 |
| 65 | 1.14 | 2.94 | 1.94 | 0.59 | 0.71 | 1.39 |
| 69 | 0.85 | 2.19 | 1.97 | 0.60 | 0.76 | 1.47 |
| 83 | 1.29 | 2.66 | 3.07 | 1.45 | 1.22 | 1.67 |
| 97 | 1.53 | 2.99 | 2.38 | 0.95 | 1.08 | 2.47 |
| 105 | 0.79 | 1.83 | 1.33 | 0.78 | 0.63 | 1.12 |
| 106 | 0.82 | 2.54 | 1.69 | 0.52 | 0.59 | 1.22 |
| 110 | 1.58 | 3.38 | 3.61 | 1.43 | 0.89 | 2.30 |
| 114 | 0.45 | 2.05 | 1.53 | 0.36 | 0.40 | 0.98 |
| 118 | 0.66 | 1.99 | 1.83 | 0.45 | 0.48 | 1.41 |
| 122 | 0.53 | 2.02 | 1.97 | 0.45 | 0.55 | 0.95 |
| Mean | 0.88 | 2.58 | 2.05 | 0.77 | 0.70 | 1.68 |
| Time | 6.67 | 5.19 | 30.9 | 408 | 3.27 | 8.48 |

**Table 2: Geometry quality comparison on the BlendedMVS dataset. Best results are highlighted as 1st, 2nd, and 3rd.**

| Methods | Ours | 3DGS | SuGaR | NeuS | NeuS2 | INSR |
|---|---|---|---|---|---|---|
| Basketball | 1.55 | 5.59 | 8.00 | 2.96 | 2.48 | 2.67 |
| Bear | 1.96 | 5.08 | 9.73 | 3.00 | 3.20 | 3.72 |
| Bread | 1.17 | 3.98 | 7.65 | 2.85 | 2.47 | 2.76 |
| Camera | 2.34 | 5.85 | 7.77 | 2.61 | 2.89 | 3.25 |
| Clock | 3.41 | 7.34 | 9.21 | 2.75 | 2.87 | 3.45 |
| Cow | 2.17 | 5.38 | 8.69 | 2.04 | 2.29 | 2.11 |
| Dog | 2.89 | 7.16 | 9.27 | 2.75 | 2.97 | 2.54 |
| Doll | 2.13 | 6.25 | 8.75 | 2.17 | 2.23 | 2.44 |
| Dragon | 2.75 | 4.90 | 9.76 | 2.95 | 2.68 | 2.20 |
| Durian | 2.35 | 6.66 | 8.04 | 3.14 | 3.39 | 5.78 |
| Fountain | 3.01 | 5.03 | 9.05 | 3.03 | 2.68 | 3.79 |
| Gundam | 0.85 | 4.46 | 7.32 | 1.62 | 1.87 | 1.36 |
| House | 2.12 | 4.64 | 7.28 | 3.23 | 3.02 | 2.93 |
| Jade | 3.50 | 6.96 | 10.7 | 4.25 | 4.03 | 3.85 |
| Man | 2.47 | 7.13 | 9.29 | 2.29 | 2.20 | 2.22 |
| Monster | 1.38 | 5.84 | 8.20 | 1.92 | 1.96 | 1.80 |
| Sculpture | 2.90 | 6.68 | 8.98 | 2.10 | 2.06 | 1.70 |
| Stone | 1.84 | 6.01 | 9.19 | 2.51 | 2.01 | 2.12 |
| Mean | 2.27 | 5.83 | 8.71 | 2.67 | 2.63 | 2.82 |
| Time | 3.82 | 3.47 | 18.3 | 377 | 3.61 | 2.78 |

applied to the reconstruction of open surfaces (Fig. 2), since Gaussian surfels do not assume closed surfaces as in the representation

**Table 3: Rendering quality comparison on DTU in common (12.5% of images for testing) and sparse (50% for testing) settings.**

| Test ratio | 12.5% | | | 50% | | |
|---|---|---|---|---|---|---|
| Methods | Ours | 3DGS | NeuS2 | Ours | 3DGS | NeuS2 |
| PSNR ↑ | 32.51 | 32.78 | 31.41 | 31.70 | 30.08 | 30.66 |
| SSIM ↑ | 0.942 | 0.943 | 0.916 | 0.936 | 0.882 | 0.910 |

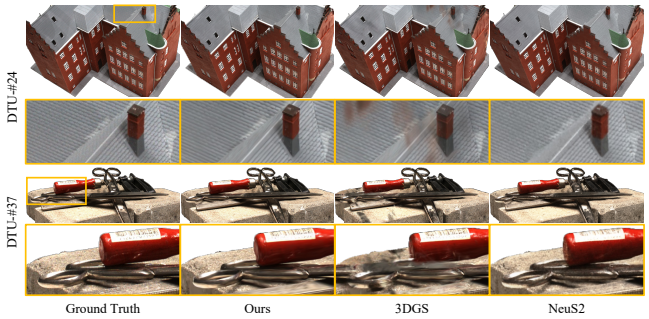


**Figure 7: Rendering quality comparison under sparse inputs on DTU. With geometry constraints, our method and NeuS2 render floater-free images compared to 3DGS. Meanwhile our renderings recover better visual details than NeuS2's.**

of signed distance functions. Overall, our method achieves a good balance between reconstruction speed and quality. In contrast to NeuS, our approach exhibits rapid convergence to high-quality reconstructions, as detailed in Table 1 and 2. Furthermore, while NeuS [Wang et al. 2021] employs a large MLP to model surfaces, offering robustness to outliers, it may result in over-smoothing of the reconstructed surfaces as depicted in Fig. 8. On the other hand, our reconstruction speed is comparable to that of INSR [Zhao et al. 2022] and NeuS2 [Wang et al. 2023], both of which utilize hash feature maps and tiny MLPs to expedite implicit surface optimization.

Compared to point-based reconstruction methods such as 3DGS [Kerbl et al. 2023] and SuGaR [Guédon and Lepetit 2023], both of which utilize Gaussian points to represent radiance fields and surfaces, our method excels in reconstructing noise-free surfaces and capturing intricate details. For 3DGS, we employ a mesh extraction method similar to ours but calculate normals directly based on the rendered depth map. For SuGaR, we utilize the proposed $\lambda$–level set of the density function with $\lambda = 0.3$ for point extraction and Poisson reconstruction with a depth of 10. As illustrated in Fig. 8, while superior to vanilla 3DGS on DTU, this approach still exhibits ellipsoid-like artifacts and holes on the surface. We believe this discrepancy arises because the regularization term that encourages the 3D Gaussians to be flat does not align well enough with the extracted $\lambda$–level set.

We also assess the rendering quality of our method by comparing it with 3DGS and NeuS2 on DTU under two settings: a common setting where 12.5% of images are reserved for testing, and a sparse setting where half of the images are reserved for testing. As shown in Table 3, our method exceeds NeuS2 in both settings,


**Figure 8: Qualitative comparisons on DTU and BlendedMVS Datasets.**

**Table 4: Loss term ablation studies on DTU using average Chamfer distance (CD) and PSNR score.**

| Metrics | Full | w/o $\mathcal{L}_c$ | w/o $\mathcal{L}_n$ | w/o $\mathcal{L}_o$ | w/o $\mathcal{L}_m$ | w/o cut |
|---|---|---|---|---|---|---|
| CD ↓ | 0.882 | 1.243 | 1.070 | 1.085 | 1.015 | 1.189 |
| PSNR ↑ | 32.51 | 31.56 | 32.63 | 32.08 | 29.10 | / |

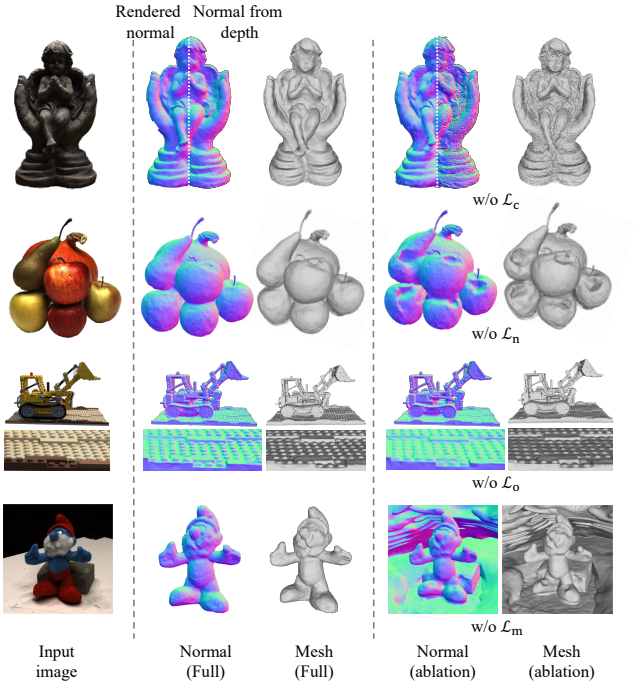


**Figure 9: *Top*: Ablation on depth-normal consistency loss. For the normal column, the part left to the central dashed line on the object represents the rendered normals, while the right part represents the normals calculated from the rendered depth. We can observe better consistency with the depth-normal consistency loss, resulting in improved reconstruction. *Second row*: Ablation on normal-prior loss; the concave parts on the surface caused by highlights are corrected. *Third row*: Ablation on opacity loss. The LEGO studs are better reconstructed, with sharper edges. *Bottom* Ablation without masks, where the absence of masks may lead to noises in texture-less background regions.**

only performs slightly worse than the vanilla 3DGS in common setting. However, due to its precise underlying geometry, our method demonstrates superior generality compared to 3DGS, resulting in a significant improvement in rendering quality in sparse settings (Fig. 7).

*Ablations.* We show quantitative analyses of our loss terms by excluding each individually from the optimization (Table 4). The photometric loss $\mathcal{L}_p$ is always included. In addition, we evaluate the effect of volumetric cutting by excluding volumetric cutting (w/o cut). Removing depth-normal consistency loss $\mathcal{L}_c$ significantly reduces reconstruction quality. The qualitative results of the ablation are shown in Fig. 9. The normal-prior loss helps reduce fluctuations in the geometric reconstruction in highlighted areas (the second row of Fig. 9), contributing to a lower Chamfer distance. However, as indicated in the third colomn of Table 4, removing the normal-prior loss can slightly improve the rendering quality. This is because this loss might prevent the optimizer from approximating the highlights as a virtual light sources behind the actual surface, resulting in lower rendering quality. The opacity loss aids in representing details (the fourth column in 4). As depicted in the third rows of Fig. 9, with this loss, the details on the "LEGO studs" are much more clearly reconstructed and rendered. The last two columns of Table 4 demonstrate the efficacy of mask loss $\mathcal{L}_m$ and volumetric cutting in noise reduction, which can improve the quality of reconstruction.

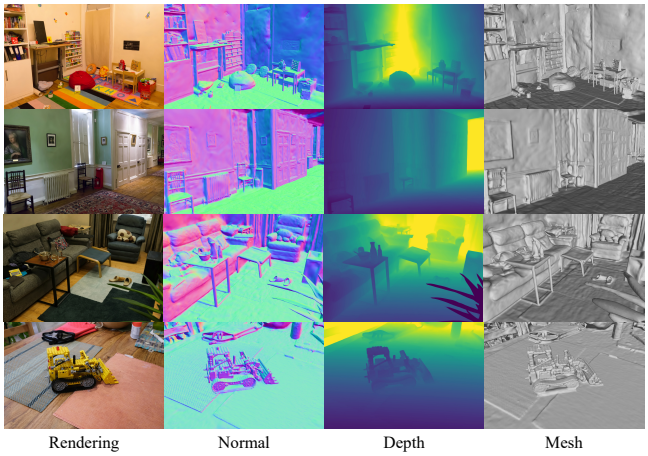


**Figure 10: Reconstruction results of our method on indoor scenes from DeepBlending [Hedman et al. 2018] and MipNeRF360 [Barron et al. 2022] datasets when initialized with sparse SfM points.**

## 5 CONCLUSION

We have demonstrated that our point-based representation, Gaussian surfels, can be efficiently optimized to achieve a good balance between high-quality surface reconstruction and computational cost.

*Limitations.* Even with monocular norm priors, our method cannot guarantee accurate reconstruction results at areas with strong specular reflections. In the future, we will investigate how to store and optimize features at Gaussian surfels to encode their view-dependent appearance. Taking the features and view direction as inputs, we can train a neural network decoder to give us additional capabilities to handle specular reflections than spherical harmonics. Moreover, we have also observed that, for surfaces with very weak textures, our reconstructed surfaces may exhibit a global shift compared to ground-truth surfaces. It is possible to mitigate this issue by incorporating information from depth sensors or more shape priors.

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for their professional and constructive comments. Weiwei Xu is partially supported by NSFC grant No. 61732016. Jiamin Xu is partially supported by NSFC grant No. 62302134 and ZJNSF grant No. LQ24F020031. This paper is supported by Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

## REFERENCES

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

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

Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. 2005. High-quality surface splatting on today's GPUs. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* IEEE, 17–141.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. 2022. Efficient geometry-aware 3D generative adversarial networks. 16123–16133.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. Tensorf: Tensorial radiance fields. 333–350.

Hanlin Chen, Chen Li, and Gim Hee Lee. 2023. NeuSG: Neural Implicit Surface Reconstruction with 3D Gaussian Splatting Guidance. *arXiv preprint arXiv:2312.00846* (2023).

François Darmon, Bénédicte Bascle, Jean-Clément Devaux, Pascal Monasse, and Mathieu Aubry. 2022. Improving neural implicit surfaces geometry with patch warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6260–6269.

Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. 2021. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. 10766–10776.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. 5501–5510.

Qiancheng Fu, Qingshan Xu, Yew Soon Ong, and Wenbing Tao. 2022. Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *Advances in Neural Information Processing Systems* 35 (2022), 3403–3416.

Yasutaka Furukawa, Carlos Hernández, et al. 2015. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision* 9, 1-2 (2015), 1–148.

Yasutaka Furukawa and Jean Ponce. 2009. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence* 32, 8 (2009), 1362–1376.

Antoine Guédon and Vincent Lepetit. 2023. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. *arXiv preprint arXiv:2311.12775* (2023).

Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. 2022. Neural 3d scene reconstruction with the manhattan-world assumption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5511–5520.

Martin Habbecke and Leif Kobbelt. 2007. A surface-growing approach to multi-view stereo reconstruction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep Blending for Free-viewpoint Image-based Rendering. 37, 6 (2018), 257:1–257:15.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. 2014. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 406–413.

Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 3 (2013), 1–13.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (2023).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Leif Kobbelt and Mario Botsch. 2004. A survey of point-based techniques in computer graphics. *Computers & Graphics* 28, 6 (2004), 801–814.

Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. 2022. Neural Point Catacaustics for Novel-View Synthesis of Reflections. *ACM Trans. Graph.* 41, 6, Article 201 (nov 2022), 15 pages. https://doi.org/10.1145/3550454.3555497

Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. 2021. Point-Based Neural Rendering with Per-View Optimization. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 29–43.

Christoph Lassner and Michael Zollhofer. 2021. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1440–1449.

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

Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. 2022. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *European Conference on Computer Vision*. Springer, 210–227.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

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

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library, Vol. 32.

Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. 2000. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 335–342.

Christian Reiser, Richard Szeliski, Dor Verbin, Pratul P Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T Barron, and Peter Hedman. 2023. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *arXiv preprint arXiv:2302.12249* (2023).

Liu Ren, Hanspeter Pfister, and Matthias Zwicker. 2002. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Computer Graphics Forum*, Vol. 21. Wiley Online Library, 461–470.

Darius Rückert, Linus Franke, and Marc Stamminger. 2022. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)* 41, 4 (2022), 1–14.

Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. 2016. Pixelwise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer, 501–518.

Steven M Seitz and Charles R Dyer. 1999. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision* 35 (1999), 151–173.

Sudipta N Sinha, Philippos Mordohai, and Marc Pollefeys. 2007. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *2007 IEEE 11th international conference on computer vision*. IEEE, 1–8.

Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022b. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. 5459–5469.

Jiaming Sun, Xi Chen, Qianqian Wang, Zhengqi Li, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. 2022a. Neural 3d reconstruction in the wild. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.

Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. 2020. State of the art on neural rendering. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 701–727.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *Advances in Neural Information Processing Systems* 34 (2021), 27171–27183.

Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. 2023. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3295–3306.

Thibaut Weise, Thomas Wismer, Bastian Leibe, and Luc Van Gool. 2009. In-hand scanning with online loop closure. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 1630–1637.

Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. 2022. Voxurf: Voxel-based efficient and accurate neural surface reconstruction. *arXiv preprint arXiv:2208.12697* (2022).

Tai-Pang Wu, Sai-Kit Yeung, Jiaya Jia, and Chi-Keung Tang. 2010. Quasi-dense 3D reconstruction using tensor-based multiview stereo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1482–1489.

Jiamin Xu, Weiwei Xu, Yin Yang, Zhigang Deng, and Hujun Bao. 2018. Online Global Non-rigid Registration for 3D Object Reconstruction Using Consumer-level Depth Cameras. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 1–12.

Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. 2020. Blendedmvs: A large-scale dataset for generalized multi-view

stereo networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1790–1799.

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

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. 2020. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems* 33 (2020), 2492–2502.

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

Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. 2022. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in neural information processing systems* 35 (2022), 25018–25032.

Jingyang Zhang, Yao Yao, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. 2022b. Critical regularizations for neural surface reconstruction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6270–6279.

Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5453–5462.

Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492* (2020).

Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. 2022a. Modeling indirect illumination for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18643–18652.

Fuqiang Zhao, Yuheng Jiang, Kaixin Yao, Jiakai Zhang, Liao Wang, Haizhao Dai, Yuhui Zhong, Yingliang Zhang, Minye Wu, Lan Xu, et al. 2022. Human performance modeling and rendering via neural animated mesh. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–17.

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. 2001. Surface splatting *(SIGGRAPH '01)*. 371–378.

Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. 2002. EWA splatting. *IEEE Transactions on Visualization and Computer Graphics* 8, 3 (2002), 223–238.