SplatSDF: Boosting Neural Implicit SDF via Gaussian Splatting Fusion

Runfa Blark Li[*†] Keito Suzuki[*] Bang Du[*] Ki Myung Brian Lee[†]
Nikolay Atanasov[†] Truong Nguyen[*]

[*]Video Processing Lab & [†]Existential Robotics Lab, UC San Diego
{runfa, k3suzuki, b7du, kmblee, natanaso, tqn001}@ucsd.edu
https://blarklee.github.io/splatsdf/

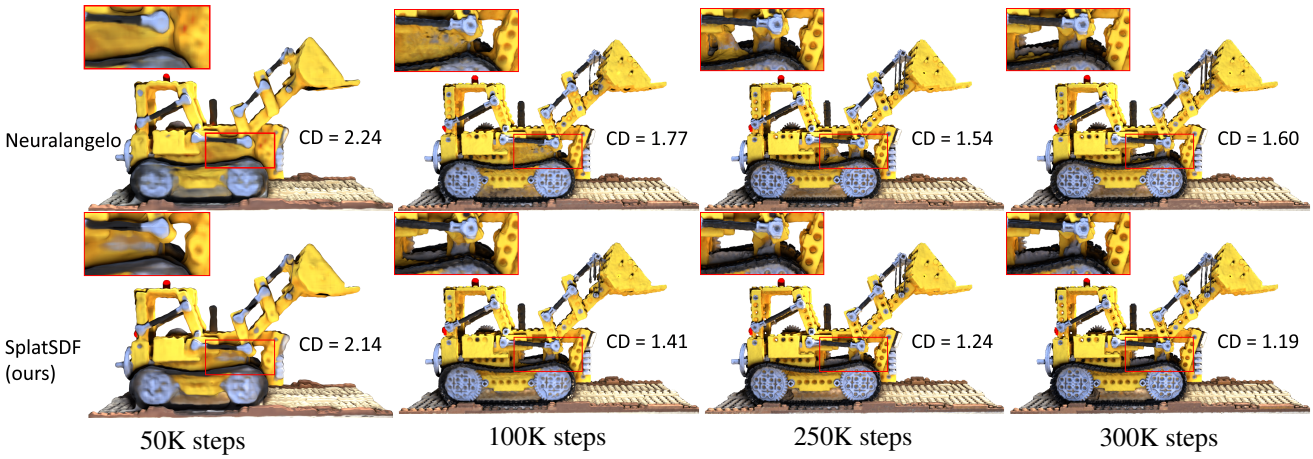

Figure 1. Our proposed SplatSDF boosts Neural Implicit SDF via Gaussian Splatting with novel architecture-level fusion strategies. SplatSDF makes it easier to converge to complex geometry (like the holes in the red boxes), achieves greater geometric and photometric accuracy, and >3 times faster convergence compared to the best baseline, Neuralangelo. ("CD" denotes Chamfer Distance).

Abstract

A signed distance function (SDF) is a useful representation for continuous-space geometry and many related operations, including rendering, collision checking, and mesh generation. Hence, reconstructing SDF from image observations accurately and efficiently is a fundamental problem. Recently, neural implicit SDF (SDF-NeRF) techniques, trained using volumetric rendering, have gained a lot of attention. Compared to earlier truncated SDF (TSDF) fusion algorithms that rely on depth maps and voxelize continuous space, SDF-NeRF enables continuous-space SDF reconstruction with better geometric and photometric accuracy. However, the accuracy and convergence speed of scene-level SDF reconstruction require further improvements for many applications. With the advent of 3D Gaussian Splatting (3DGS) as an explicit representation with excellent rendering quality and speed, several works have focused on improving SDF-NeRF by introducing consistency losses on depth and surface normals between 3DGS and SDF-NeRF. However, loss-level connections alone lead to incremental improvements. We propose a novel neural implicit SDF called "SplatSDF" to fuse 3DGS and SDF-NeRF at an architecture level with significant boosts to geometric and photometric accuracy and convergence speed. Our SplatSDF relies on 3DGS as input only during training, *and keeps the same complexity and efficiency as the original SDF-NeRF during inference. Our method outperforms state-of-the-art SDF-NeRF models on geometric and photometric evaluation by the time of submission.*

1. Introduction

An SDF encodes geometric information of a continuous 3D space as a distance function, making it convenient to obtain other 3D representations, such as surface meshes and point clouds.This makes SDF valuable for many applications, including robotics and XR/VR, where accurate understanding and interaction with 3D environments are essential.

Classically, per-voxel SDF is obtained from multi-view

depth maps with truncated signed distance function (TSDF) fusion [30]. Deep-learning TSDF estimation [18, 29, 34, 36, 37, 45] inherits the per-voxel estimation style but avoids TSDF fusion from depth maps by leveraging 3D convolutions to directly regress a SDF value per-voxel. Although well-trained models generalize to various scenes, the supervised training process is time-consuming and yet the models do not provide continuous-space SDF estimation.

Since the formulation of neural radiance field (NeRF) models [26], SDF estimation has been explored in a similar manner, utilizing volumetric rendering for supervision [7, 8, 11, 14, 20, 27, 31, 32, 40–42, 44, 46, 47, 49, 51, 54, 57, 60, 61]. SDF-NeRF methods leverage a multi-layer perceptron (MLP) to regress SDF and use a density function to link SDF and opacity followed by volumetric rendering (alpha blending) used in NeRF to supervise the rendered color. Once trained, an SDF-MLP can provide a continuous SDF representation of the scene and can be retrieved independently from opacity and color during inference. Instead of explicitly interpolating from voxel-level TSDF estimation, SDF-NeRF implicitly interpolates with MLP weights learned and stored for the whole scene/object and enables photometric rendering.

Another branch of work leverages point clouds to guide the convergence of an SDF-MLP without image rendering [1, 3, 5, 16, 33]. Given accurate point clouds as geometric guidance, these methods explore different losses to bring the implicit surface close to the point cloud. With the rise of 3DGS [15] as an explicit 3D representation with effective rendering speed and quality, the latest research investigates using 3DGS for surface reconstruction to obtain faster convergence [6, 10, 12, 24, 58]. However, these methods only reconstruct the surface, fitting a surface mesh to 3DGS. In contrast, our focus is not on surface reconstruction but recovering a continuous SDF in 3D space, from which the surface mesh can be extracted as the zero-level iso-surface using the marching cubes algorithm [23]. Continuous SDFs offer greater utility than surface meshes in tasks such as robot motion planning [17, 19, 22, 39].

Three relevant concurrent methods [25, 48, 56] were proposed to improve SDF-NeRF using 3DGS. However, they leverage 3DGS only at the loss level where SDF-NeRF and 3DGS are two independent models that only interact through losses. Thus, their results show insignificant improvements over previous SDF-NeRF methods without 3DGS, as our evaluations show in Sec. 5.3.

We propose a novel approach to incorporate 3DGS at an architectural level of an SDF-NeRF model. In our formulation, 3DGS is needed as an input at training time but at inference time, the SDF-NeRF can be used independently. Our method outperforms previous state-of-the-art (SOTA) SDF-NeRF methods in terms of Chamfer distance and peak signal-to-noise ratio (PSNR), while being faster to train. Our contributions can be summarized as follows.

- We propose a novel neural implicit SDF architecture "SplatSDF", which incorporates 3DGS and an SDF-MLP at an architecture level to enable volumetric rendering.
- Experiments show that our SplatSDF outperforms SOTA SDF-NeRF and 3DGS surface reconstruction methods in geometric and photometric accuracy. We also achieve $>$ $3\times$ faster convergence on complex shapes with the same geometric accuracy.
- SplatSDF requires 3DGS only during training and, hence, its inference speed is the same as that of the original SDF-NeRF method.

2. Related Work

We categorize neural implicit SDF into rendering-based, pointcloud-based, and 3DGS-driven.

Rendering-based Neural Implicit SDF. Whereas NeRF uses volumetric rendering to train an MLP that stores the color and opacity of a scene [26], an SDF-NeRF trains an MLP that stores the SDF of a scene [41]. For supervision with images, volumetric rendering is achieved by converting the SDF to opacity using a density function, such as the logistic distribution [41] or the Laplace distribution [54].

An important direction in this domain is to improve the training speed and the reconstruction quality. NeuS2 [43] achieves a significant training speedup by deriving an approximation of the 2nd-order derivative of the SDF in the back-propagation for a specific SDF-MLP designed with ReLU activation and hash encoding. SuperNormal [50] speeds up the back-propagation for the SDF gradient with a patch-ray strategy using normal and silhouette maps. Neuralangelo [20] proposes a numerical SDF gradient for hash-grid encoding and improves geometric accuracy. Additional geometric clues can be used for further improvements, such as off-line depth estimation [57], semantic segmentation [11], and plane estimation [40]. Our approach leverages the 3DGS representation of a scene as an input to the SDF-NeRF during the training stage to achieve faster and more accurate results.

Point Cloud-based Neural Implicit SDF. Another important branch of work uses pure point clouds to train an SDF-MLP without images. DeepSDF [33], widely-considered as the original work on SDF-MLP, is trained from pure point clouds. NeuralPull [1] proposes a "pulling-loss" to pull randomly sampled query points to the nearest point from the point cloud along the direction of SDF gradient. [16] designs a pseudo-SDF ground truth from point cloud for loss at each small step of training to guide the convergence and avoid local minima. However, these methods highly depend on the quality of the point cloud since the SDF-MLP can converge to an inaccurate surface with an inaccurate or sparse point cloud. The absence of images also makes them impossible to render photometric details like SDF-NeRF.

Meanwhile, our approach uses 3DGS to not only capture photometric details for SDF-NeRF, but also geometric cues as is done by point cloud-based approaches.

3DGS-based Surface Reconstruction. Since it is slow to estimate the surface by extracting the 0-level iso-surface from SDF-NeRF with marching cubes, latest approaches turn to 3DGS to directly reconstruct the surface at a faster speed. However, these methods only focus on the surface reconstruction by rendering multiple depth images that are projected to a point cloud, and used for Poisson surface reconstruction [6, 10], TSDF fusion [12], or marching tetrahedra [58]. Surface meshes are of less value than continuous distance field produced by our method in tasks such as motion planning and collision checking in robotics. [17, 19, 22, 39].

3DGS-driven Neural Implicit SDF. A few SOTA concurrent methods improve SDF-NeRF through 3DGS by designing GS-SDF consistency losses. NeuSG [4] proposes a surface normal consistency loss between the SDF and GS separately. Similarly, GSDF [56] adds a depth consistency loss between the depth image from the SDF and GS representations, and 3DGSR [25] introduces another pseudo surface normal loss from GS-rendered depth under the local planarity assumption. GaussianRoom [48] uses off-the-shelf surface normal and edge detection models to extract more constraints specifically for indoor scenes. All of these methods treat SDF-NeRF and 3DGS separately without model-level interactions. Their interactions are only at loss level, which shows insufficient improvement over previous SDF-NeRF methods. In contrast, our approach introduces a novel architecture-level 3DGS fusion algorithm that is directly used in the SDF-MLP, which greatly boosts the speed, geometric & photometry quality of SDF-NeRF, without using any auxiliary GS-SDF consistency losses.

3. Problem Statement

We assume that we are given RGB images with known camera poses and a sufficiently trained 3DGS [15] model $\mathcal{G}$ of the scene. Each Gaussian $G_k \in \mathcal{G}$ is parameterized by a mean $\mu(G_k)$, a covariance $\Sigma(G_k)$, opacity $\alpha(G_k)$, and spherical harmonics $SH(G_k)$. The RGB images may be given, or rendered from the 3DGS model $\mathcal{G}$.

Our objective is to recover the SDF $f_{\mathcal{S}} : \mathbb{R}^3 \to \mathbb{R}$ of the occupied space $\mathcal{S}$ in the scene. The SDF is defined as $f_{\mathcal{S}}(\mathbf{x}) = \pm \inf_{\mathbf{y} \in \partial \mathcal{S}} \|\mathbf{x} - \mathbf{y}\|$ with positive sign when $\mathbf{x} \notin \mathcal{S}$ and negative sign otherwise. This allows geometric reconstruction of the surface of $\mathcal{S}$ as the zero-level set of its SDF $\partial \mathcal{S} = \left\{ \mathbf{x} \in \mathbb{R}^3 \mid f_{\mathcal{S}}(\mathbf{x}) = 0 \right\}$. Our approach uses the 3DGS $\mathcal{G}$ during training but yields the SDF $f_{\mathcal{S}}$ as an MLP that can be queried independently of $\mathcal{G}$ at inference time.

4. SplatSDF

Figure 2 shows an overview of our SplatSDF model, including an *SDF module* and a *color module*. Our contribution focuses on the former, whereas the latter is designed similarly to NeRF [26].

The SDF module constructs spatial and 3DGS embeddings e_{sdf} and e_{gs}, and predicts per-point SDF as:

$$f_{\mathcal{S}}(\mathbf{x}) = f_{sdf}(Fuse(e_{sdf}(\mathbf{x}), e_{gs}(\mathbf{x}, \mathcal{G}))), \tag{1}$$

where $Fuse$ is a novel fusion method we detail in Sec. 4.2. At inference time, fusion with the 3DGS embedding is optional, and the SDF model can be queried as:

$$f_{\mathcal{S}}(\mathbf{x}) = f_{sdf}(e_{sdf}(\mathbf{x})). \tag{2}$$

In what follows, we discuss our main contributions in the SDF module, which consists of *3DGS aggregator* and *3DGS fusion*, in detail.

4.1. 3DGS Aggregator

The 3DGS aggregator constructs per-Gaussian embeddings $e_g(G)$ by aggregating attributes of each Gaussian. To ensure the scale remains the same as the SDF embeddings, we share the same hash-encoder h [28] between SDF and 3DGS. We build the 3DGS aggregator as an MLP f_{agg}. The per-GS embeddings e_g and per-point SDF embeddings e_{sdf} are computed as:

$$\begin{aligned} e_{sdf}(\mathbf{x}) &= h(\mathbf{x}), \\ e_g(G) &= f_{agg}(h(\mu(G)), \Sigma(G), c(G), SH(G)), \end{aligned}$$

where μ, Σ, c, SH are the center coordinate, covariance, color and spherical harmonics parameters of G. We do not aggregate the GS opacity α in f_{agg} since it plays an important role in the following fusion step.

4.2. 3DGS Fusion

The 3DGS fusion stage fuses the per-Gaussian embeddings $e_g(G)$ to query-point Gaussian embeddings $e_{gs}(\mathbf{x}, \mathcal{G})$ to boost the query-point SDF embeddings e_{sdf}. This implements the fusion function $Fuse(e_{sdf}(\mathbf{x}), e_{gs}(\mathbf{x}, \mathcal{G}))$ in (1).

Weighted Fusion of per-Gaussian Embeddings Inspired by the alpha blending in 3DGS [15], we propose a *weighted blending strategy* to fuse the per-Gaussian embeddings $e_g(G)$ into a GS embedding $e_{gs}(\mathbf{x}, \mathcal{G})$ at each query point $\mathbf{x}$, as follows:

$$e_{gs}(\mathbf{x}, \mathcal{G}) = \frac{1}{K} \sum_{G_k \in \text{KNN}(\mathbf{x}, \mathcal{G})} e_g(G_k) w(\mathbf{x}, G_k) \alpha(G_k). \tag{3}$$

Here, $\text{KNN}(\mathbf{x}, \mathcal{G})$ denotes the K Gaussians with means nearest to the query point $\mathbf{x}$, $w(\mathbf{x}, G) =$

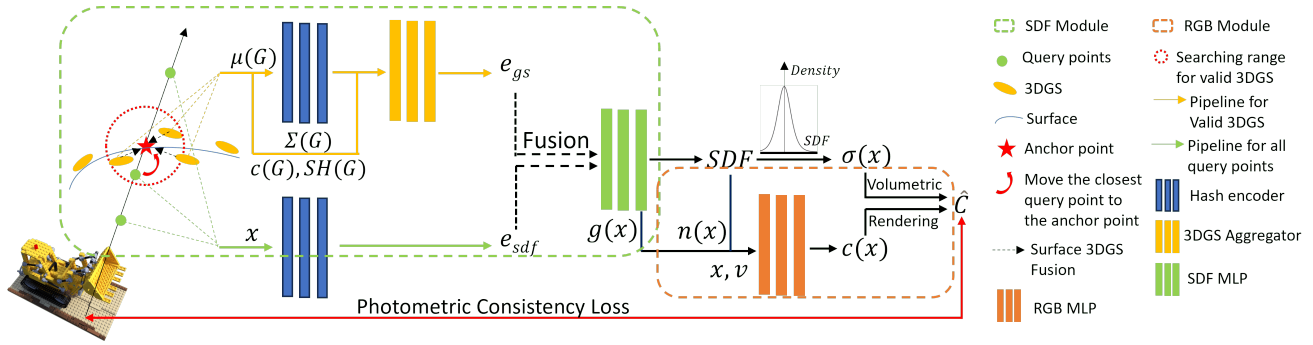

Figure 2. **Overview**. Our SplatSDF takes posed RGB images and 3DGS to train an SDF-NeRF. We use 3DGS-rendered depths to identify the anchor point and shift the closest query point to the anchor point. With a shared hash encoder, we extract query-point SDF embeddings e_{sdf} and 3DGS embeddings e_{gs}. Our method applies a 3DGS aggregator to merge the 3DGS attributes: mean μ, covariance Σ, color c, and spherical harmonics SH. We propose a novel *surface 3DGS fusion* to fuse e_{gs} and e_{sdf} only around the anchor point and regress to SDF. With a density function, SDF is converted to per-point density $\sigma(x)$. We take the geometric features $g(x)$, the surface normal from SDF $n(x)$, the query-point coordinates x and the viewing angle v to estimate per-point color $c(x)$ and obtain the per-pixel color $\hat{C}$ by volumetric rendering to supervise with input images. Our core design is in the the *3DGS aggregator* and the *3DGS fusion*.

$\exp\left(-\frac{1}{2}(\mathbf{x}-\mu(G))^T\Sigma(G)^{-1}(\mathbf{x}-\mu(G))\right)$ is the density of Gaussian G at query point $\mathbf{x}$, and α is the opacity of Gaussian G. The density $w(\mathbf{x}, G)$ and opacity α balance the contribution of neighbor GS embeddings to the query-point GS embedding. Division by the number of selected Gaussians K ensures that the resulting embedding is normalized.

The weighted blending strategy is an extension of 2D the blending strategy in 3DGS [15] to 3D in the following sense. Whereas 3DGS uses a projected 2D Gaussian weight function for blending colors in the image plane, our approach uses the 3D Gaussian weight function to blend the embeddings in the 3D space directly. Furthermore, the KNN algorithm can be seen as the 3D equivalent to frustum culling in the 2D projection step of 3DGS rasterization.

In our implementation, we accelerated the KNN algorithm by hashing the GS $\mathcal{G}$ and the set of query points $\mathbf{X} = \{\mathbf{x}\}$ into L^3 voxels, inspired by PointNeRF [52]. Doing so decreases the computational complexity from $O(|\mathbf{X}||\mathcal{G}|)$ in the naive case to $O(\frac{|\mathbf{X}||\mathcal{G}|}{L^3})$, where $|\mathbf{X}|$ and $|\mathcal{G}|$ are the number of query points and the number of Gaussians. Because we only consider Gaussians belonging to the same voxel as the query point, we are implicitly imposing a radius constraint in addition to the K-nearest requirement. Further details are provided in the Supplementary Material.

Surface 3DGS Fusion. To fuse the GS embedding $e_{gs}(\mathbf{x}, \mathcal{G})$ with the SDF embedding $e_{sdf}(\mathbf{x})$, one possibility is to train an MLP that concatenates both GS and SDF embeddings and regresses another embedding of the same dimension, as is done in PointNeRF [52] with point clouds. We present a fusion approach that is not only simpler but also significantly more effective.

Our approach begins from the observation that all query points $\mathbf{x}$ lie along a ray $\mathbf{r}$ during training. We first compute an *anchor point* $\mathbf{x_r}$, which we define as the first intersection between the ray and the surface. This can be computed using the depth value rendered from the GS $\mathcal{G}$. Then, for each ray, we replace the closest query point to the anchor point $\mathbf{x_r}$, with the anchor point $\mathbf{x_r}$ itself, and take the GS embedding $e_{gs}(\mathbf{x_r}, \mathcal{G})$ as the output. For all other points, we use the SDF embedding $e_{sdf}(\mathbf{x})$. Stated differently, we merely 'replace' the SDF embedding with the GS embedding at the anchor point, rather than 'combining' embeddings at all query points.

This fusion strategy was found to be more effective because the GS embedding is used only near the surface. This avoids many spurious Gaussian blobs that are found further from the surface, which is a common problem in 3DGS. Moreover, the computational complexity is dramatically reduced, since we only compute the GS embedding at one anchor point per ray. Such simple, sparse replacement with GS embeddings near the surface at the anchor points leads to notable improvements in convergence speed and accuracy over SDF-NeRF without GS, as our results show.

Figure 3 shows a qualitative comparison of the proposed strategy against a "dense" fusion via concatenation and MLP regression over both GS and SDF embeddings. The first row of Figure 3 shows that the dense fusion approach (on the left) leads to bumpy surface artifacts. Our experiment shows that flipping signs of SDF match to where the surface artifacts happen, as shown in the second row. According to the density function in the second row, the flipping signs of SDF indicates where opacity $\sigma(\mathbf{x})$ is fused along the ray, which proves that erroneous GS were fused to cause wrong SDF. Such erroneous GS are commonly found further from the true surface. Thus, using the GS embed-

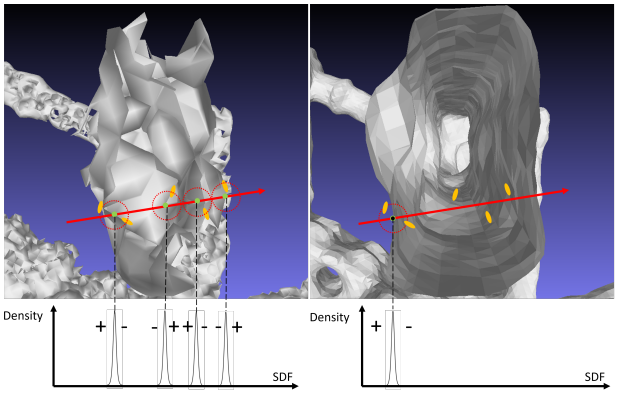

Figure 3. **Dense 3D Fusion vs Surface 3D Fusion**. Left: Dense 3DGS fusion on all valid query points (green points). Right: Surface 3DGS fusion only on the anchor point (black point). Fusing query points inside of surfaces using spurious GS (orange ellipsoids) far from the true surface leads to bumpy surface artifacts.

ding at query points far from the surface will incorporate incorrect GS to embedding, contributing erroneous density. Such erroneous density has a greater impact in our case than in 2D RGB images (e.g. as in PointNeRF [52]), because the embeddings are directly used to predict the SDF in 3D, rather than its projection to an image.

4.3. Training

Out SplatSDF model is trained by supervising images generated by volumetric rendering against the target images. To this end, we construct an auxiliary *color module* that estimates the per query-point radiance $\mathbf{c}(\mathbf{x})$ using an RGB MLP f_{color}, whose inputs are the query position $\mathbf{x}$, viewing direction $\mathbf{v}$, geometric features $g(\mathbf{x})$, and the normals $n(x) = \nabla_{\mathbf{x}} f_{\mathcal{S}}(\mathbf{x})$ (i.e. the gradient of the SDF):

$$\mathbf{c}(\mathbf{x}) = f_{color}(\mathbf{x}, \mathbf{v}, g(\mathbf{x}), \nabla_{\mathbf{x}} f_{\mathcal{S}}(x)). \tag{4}$$

For simplicity, we do not explicitly input the GS $\mathcal{G}$ into the RGB-MLP because the geometric features $g(x)$ and the normals $n(x)$ are already derived from the GS $\mathcal{G}$.

For volumetric rendering, we convert the per-point SDF $f_{\mathcal{S}}(\mathbf{x})$ to per-point opacity $\sigma(\mathbf{x})$ using the logistic distribution $\phi_s(d) = se^{-sd}/(1+e^{-sd})^2$ where s is the inverse of standard deviation, and d is the SDF value at the query point x. The parameter s is learnable, and is expected to approach zero as the SDF-MLP f_{sdf} converges. The opacity is used in conjunction with the color module to volumetrically render images for supervision:

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{\mathbf{x}_i \in \mathbf{r}} T_i \left(1 - \exp\left(-\sigma\left(\mathbf{x}_i\right)\delta_i\right)\right) \mathbf{c}(\mathbf{x}_i)), \tag{5}$$

where $\delta_i = ||\mathbf{x}_{i+1} - \mathbf{x}_i||_2$ is the distance between adjacent query points along the ray $\mathbf{r}$, and $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$ denotes the accumulated transmittance.

The latest GS-based surface reconstruction works [6, 10, 58] and the concurrent GS-SDF works [25, 48, 56] adopt new losses from geometric priors such as GS surface normals and depth. However, to show the effectiveness of our architecture, we only use the same basic losses used by Neuralangelo [20]. The losses are L1 photometric consistency loss, Eikonal loss, and a curvature loss:

$$\mathcal{L} = \mathcal{L}_{\text{RGB}} + w_{\text{eik}}\mathcal{L}_{\text{eik}} + w_{\text{curv}}\mathcal{L}_{\text{curv}}. \tag{6}$$

The photometric consistency loss $\mathcal{L}_{\text{RGB}}$ is given by:

$$\mathcal{L}_{\text{RGB}} = \left\| \hat{\mathbf{C}} - \mathbf{C} \right\|_1, \tag{7}$$

where $\hat{\mathbf{C}}$ and $\mathbf{C}$ are the rendered images by SDF-NeRF and the input image, respectively. The Eikonal loss $\mathcal{L}_{\text{eik}}$ enforces that the gradient of SDF should be equal to 1:

$$\mathcal{L}_{\text{eik}} = \frac{1}{N} \sum_{i=1}^{N} \left(\|\nabla f_{sdf}(\mathbf{x}_i)\|_2 - 1 \right)^2. \tag{8}$$

The curvature loss $\mathcal{L}_{\text{curv}}$ is a regularisation term to smooth the SDF gradients, given by:

$$\mathcal{L}_{\text{curv}} = \frac{1}{N} \sum_{i=1}^{N} \left| \nabla^2 f_{sdf}(\mathbf{x}_i) \right|. \tag{9}$$

5. Experimental Results

5.1. Datasets & Implementation Details

We use the DTU [13] and NeRF Synthetic datasets [26] for training and evaluation. We use 12 scenes from the DTU dataset which contains either 49 or 64 posed images per-scene obtained by a robot-held monocular RGB camera and the point cloud ground truth obtained from a structured-light scanner. For the NeRF Synthetic Dataset, we use 5 objects with 100 posed images for each scene. We use the "Lego" scene for ablation study.

We initialize the point cloud from MVSNet [53], which can be replaced by other SOTA depth estimation methods. We initialize the standard 3DGS [15] and fix the number and coordinates of the point cloud during optimization to keep geometric accuracy. For now, we fix the well-trained 3DGS and do not jointly optimize with SDF-NeRF. For our 3DGS Aggregator, we use a 3-layer MLP where the 2nd layer concatenates the upper-triangle of the GS covariance and the 3rd layer concatenates the GS color and SH. We do not use object segmentation masks as used in some previous works, but randomly sample 512 pixels for each view, and sample 128 points per ray. We follow Neuralangelo [20] to sample foreground and background points separately in the two SDF-MLP. We conduct uniform sampling for the background with 32 points and a coarse-to-fine sampling

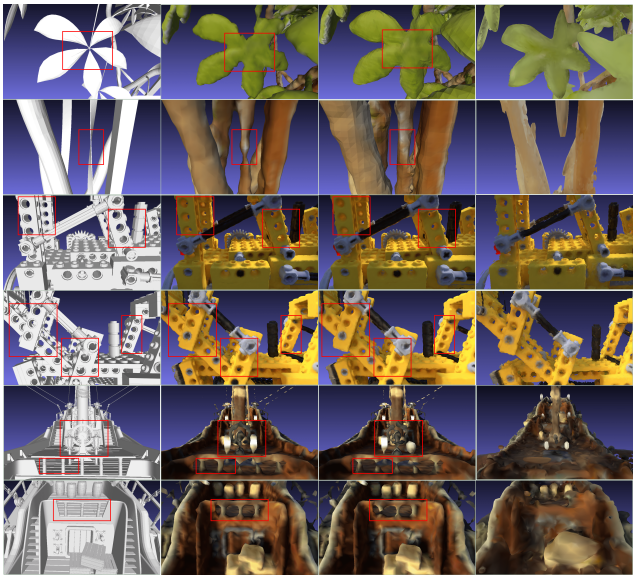

Figure 4. **Surface Mesh Comparison** on the NeRF Synthetic Dataset. Left to right: Ground Truth mesh (color is not available), SplatSDF, Neuralangelo, SuGAR. Row 1-2: Ficus. Row 3-4: Lego. Row 5-6: Ship. Zoom in to check details in red boxes. No red boxes for SuGAR since it is overall worse than SDF-NeRFs.

[26] for the foreground with 96 points. We force the anchor points to be on the foreground by only implementing the surface 3DGS fusion on the foreground NeRF.

We use Chamfer Distance (CD) in mm as the geometric evaluation metric. As per previous work, we sample 3D grid coordinates to estimate the SDF and use Marching Cubes to get the surface mesh. We then sample points from the mesh and compare with ground truth points to compute the CD. We use PSNR as the photometric evaluation metric. For the NeRF Synthetic Dataset, we train on the standard training split with 100 images per scene and test on the standard testing split with 200 images per scene. For the DTU dataset, since there are no standard train/test splits, we train and test on the same images per-scene.

5.2. Qualitative Results

We adopt Neuralangelo [20] as the best baseline. Figure 1 shows that our SplatSDF achieves faster and more accurate convergence. SplatSDF only takes 100K steps to get $CD = 1.41$, indicating > 3 times faster convergence compared to Neuralangelo which takes 300K steps to get $CD = 1.60$. SplatSDF also achieves better final converged accuracy by capturing difficult shapes and details. While SDF-NeRF's iso-surface is always initialized as a unit sphere and "pulled" inside to fit to the convex surface, it is common to see it "under-fitting" to concave surfaces with small/thin details. This is because using the visual momentum alone from previous methods tends to smooth and blur the iso-surface and is insufficient to "pull" it to complex shapes. However, our SplatSDF amends this "under-fitting" with the novel architecture-level 3DGS fusion. An example is shown in the red box in Figure 1 where our model quickly captures the holes, whereas previous methods do not. All results in Figure 4 are trained to converge and the details in the red boxes validate the improvements, such as the thin leaves and stems, the small holes in the Lego, the helm, the white lamp and the rail in the ship. We also include a comparison with the SOTA 3DGS-based surface reconstruction work SuGAR [10]. Although SuGAR achieves faster surface reconstruction, its quality is lower than SOTA SDF-NeRFs since they transform the shape of the Gaussian primitives to fit the surfaces causing noisy artifacts and missing details. Moreover, it cannot estimate the distance field for arbitrary 3D coordinates. Additional visual comparisons on the DTU dataset are in the Supplementary.

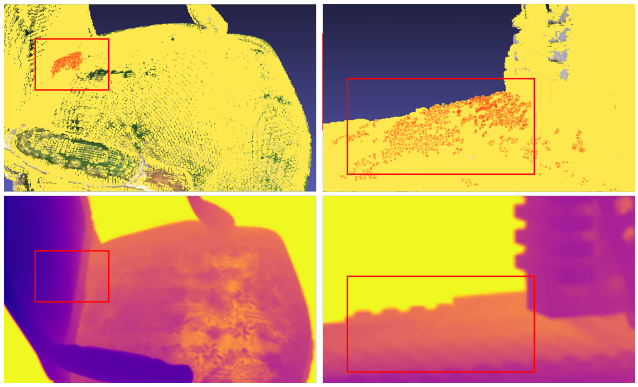

Figure 5. **Tolerance to erroneous 3DGS initialized from noisy point cloud**. First row: GS center in yellow overlap with the estimated surface mesh. Noisy GS centers is in red and shown in red boxes. Second row: No noise in the red box from GS-rendered depth.

Tolerance to noisy initialization. We also show that our SplatSDF can tolerate noise from 3DGS. The first row of Figure 5 shows the GS center (point cloud) used to initialize SplatSDF. For better visualization, we overlay our final estimated mesh with the point cloud (in yellow) obtained from MVSNet [53] used to initialize 3DGS. Although initialized with noisy 3DGS, our SDF estimation manages to nullify it. We attribute the reasons to: 1. As shown in the second row of Figure 5, GS-rendered depth eliminates errors from erroneous GS centers since we use volumetric rendering rather than surface rendering and estimate accurate anchor points to fuse correct surface 3DGS. 2. SDF-NeRF itself tends to under-fit to complex shapes by smoothing and blurring details, which alleviates the errors introduced by noisy 3DGS centers.

	Scan ID	24	37	40	55	63	65	69	83	105	106	110	114	Mean
Traditional	COLMAP [35]	0.81	2.05	0.73	1.22	1.79	1.58	1.02	3.05	2.05	1.00	1.32	0.49	1.43
	NeRF [26]	1.90	1.60	1.85	0.58	2.28	1.27	1.47	1.67	1.07	0.88	2.53	1.06	1.51
SDF-NeRF	UNISURF [31]	1.32	1.36	1.72	0.44	1.35	0.79	0.80	1.49	0.89	0.59	1.47	0.46	1.06
	MVSDF [59]	0.83	1.76	0.88	0.44	1.11	0.90	0.75	1.26	1.35	0.87	0.84	0.34	0.94
	VolSDF [54]	1.14	1.26	0.81	0.49	1.25	0.70	0.72	1.29	0.70	0.66	1.08	0.42	0.88
	NeuS [41]	1.00	1.37	0.93	0.43	1.10	0.65	0.57	1.48	0.83	0.52	1.20	0.35	0.87
	NeuS-12 [41]	0.93	1.07	0.81	0.38	1.02	0.60	0.58	1.43	0.78	0.57	1.16	0.35	0.81
	HF-NeuS [42]	0.76	1.32	0.70	0.39	1.06	0.63	0.63	1.15	0.80	0.52	1.22	0.33	0.79
	MonoSDF [57]	0.66	0.88	0.43	0.40	0.87	0.78	0.81	1.23	0.66	0.66	0.96	0.41	0.73
	RegSDF [60]	0.60	1.41	0.64	0.43	1.34	0.62	0.60	0.90	1.02	0.60	0.59	0.30	0.75
	PET-NeuS [44]	0.56	0.75	0.68	0.36	0.87	0.76	0.69	1.33	0.66	0.51	1.04	0.34	0.71
	OAV [27]	1.92	2.35	1.96	1.11	1.83	2.01	1.30	1.53	1.50	0.71	1.56	0.83	1.55
	TUVR [61]	0.83	1.06	0.57	0.40	1.00	0.62	0.62	1.41	0.94	0.57	1.07	0.35	0.79
	DebSDF [49]	0.71	0.94	0.46	0.39	1.05	0.61	0.59	1.49	0.88	0.61	1.05	0.34	0.76
	NeuralWarp [7]	0.49	0.71	0.38	0.38	0.79	0.81	0.82	1.20	0.68	0.66	0.74	0.41	0.67
	Geo-NeuS [8]	0.46	0.85	0.38	0.43	0.89	0.50	0.50	1.26	0.66	0.52	0.82	0.31	0.63
	Neuralangelo [20]	0.37	0.72	0.35	0.35	0.87	0.54	0.53	1.29	0.73	0.47	0.74	0.32	0.61
	SplatSDF (ours)	0.35	0.67	0.32	0.31	0.84	0.54	0.55	1.16	0.65	0.47	0.75	0.31	0.58
GS-based Surface Reconstruction	Scaffold-GS [24]	7.23	6.23	6.48	7.44	8.17	4.27	5.78	5.45	6.36	5.05	5.95	6.32	6.23
	3DGS [15]	2.14	1.53	2.08	1.68	3.49	2.21	1.43	2.07	1.75	1.79	2.55	1.53	2.02
	SuGAR [10]	1.47	1.33	1.13	0.61	2.25	1.71	1.15	1.63	1.07	0.79	2.45	0.98	1.38
	GaussianSurfels [6]	0.66	0.93	0.54	0.41	1.06	1.14	0.85	1.29	0.79	0.82	1.58	0.45	0.88
	2DGS [12]	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	0.76	0.70	1.40	0.40	0.79
	GausianField [58]	0.50	0.82	0.37	0.37	1.12	0.74	0.73	1.18	0.68	0.77	0.90	0.42	0.72
GS-guided SDF-NeRF	3DGSR [25]	0.68	0.84	0.70	0.39	1.16	0.87	0.77	1.48	0.87	0.69	0.80	0.42	0.81
	GSDF [56]	0.59	0.94	0.46	0.38	1.30	0.77	0.73	1.59	0.76	0.59	1.22	0.38	0.81

Table 1. **Quantitative Results on the DTU Dataset** (Chamfer Distance in mm ↓). Yellow is the best and pink is the second best. Our SplatSDF achieves the best geometric accuracy. All results are from latest papers and we validate the results of the baseline Neuralangelo. See Supplementary for the source of each result.

	Chair	Ficus	Lego	Mic	Ship	Mean
VolSDF [54]	1.18	3.01	2.26	1.13	6.42	2.80
NeuS [41]	3.99	0.94	2.56	1.00	5.38	2.77
NeRO [21]	1.27	1.22	1.90	0.87	4.95	3.72
BakedSDF [55]	1.83	10.9	1.13	0.84	3.88	3.32
NeRF2Mesh [38]	1.62	0.65	1.93	0.78	2.20	1.44
RelightableG [9]	3.65	1.26	1.63	1.76	3.35	2.33
HF-NeuS [42]	0.69	1.12	0.94	0.72	2.18	1.13
3DGSR [25]	1.01	0.69	1.35	1.15	3.35	1.51
Neuralangelo [20]	0.56	1.12	1.60	0.78	0.78	0.97
SplatSDF (ours)	0.71	0.91	1.19	0.75	0.72	0.86
NeRF [26]	33.00	30.15	32.54	32.91	28.34	31.39
VolSDF [54]	25.91	24.41	26.99	29.46	25.65	26.48
NeuS [41]	27.95	25.79	29.85	29.89	25.46	27.79
HF-NeuS [42]	28.69	26.46	30.72	30.35	25.87	28.42
PET-NeuS [44]	29.57	27.39	32.40	33.08	26.83	29.85
NeRO [21]	28.74	28.38	25.66	28.64	26.55	27.60
BakedSDF [55]	31.65	26.33	32.69	31.52	27.55	29.95
NeRF2Mesh [38]	34.25	30.08	34.90	32.63	29.47	32.27
Mip-NeRF [2]	35.14	33.29	35.70	36.51	30.41	34.21
3DGS [15]	35.36	34.87	35.78	35.36	30.80	34.43
Ins-NGP [28]	35.00	33.51	36.39	36.22	31.10	34.44
Neuralangelo [20]	34.72	35.91	33.20	36.79	31.45	34.41
SplatSDF (ours)	34.73	36.15	33.24	37.06	31.47	34.53

Table 2. **Quantitative Results on NeRF Synthetic Dataset**. The top part shows the geometric accuracy in Chamfer Distance in mm ↓, and the bottom part shows the photometric accuracy in PSNR ↑. Yellow is the best and pink is the second best. Our SplatSDF achieves the best overall accuracy.

5.3. Quantitative Results

Table 1 shows a comparison on the DTU dataset over three categories. Our model outperforms the best baseline Neuralangelo [20] and achieves the lowest CD over all previous works. We analyze two reasons why our method performs worse than the baseline on a few scenes. The first reason is that the "anchor points" estimated from 3DGS rendered depth are inaccurate in some area and the 3DGS quality is insufficient. The second reason is that the DTU dataset contains erroneous point clouds at some details. The 3DGS-based surface reconstruction methods generally train faster but their accuracy is lower than SOTA SDF-NeRF methods, as can be seen in Table 1 and Figure 4. We also show the results from two concurrent methods, GSDF [56] and 3DGSR [25], which use 3DGS to guide SDF-NeRF, albeit only at the loss-level. The results show that loss-level connections do not provide improvements over SDF-NeRF, where our architecture-level fusion method does. We conduct more comparisons on the NeRF Synthetic Dataset (Table 2), and the results show that our SplatSDF outperforms SOTA methods on both geometric and photometric accuracy, especially our baseline Neuralangelo [20]. The results in Table 2 further proves that our model improves SDF-

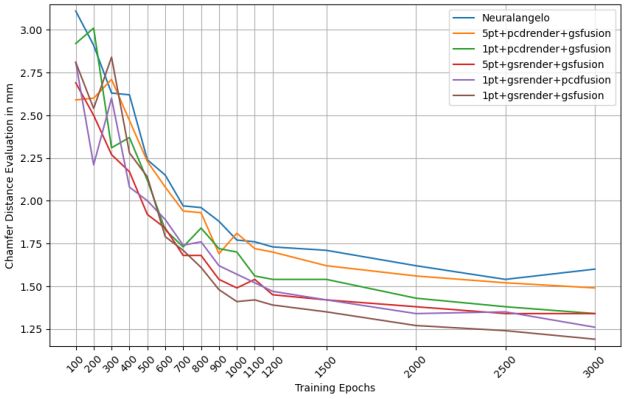

Figure 6. **Ablation study of geometric accuracy on "Lego"**. All variations of our methods achieve faster convergence to better accuracy than the baseline "Neuralangelo". Exact values are summarized in the table of Supplementary.

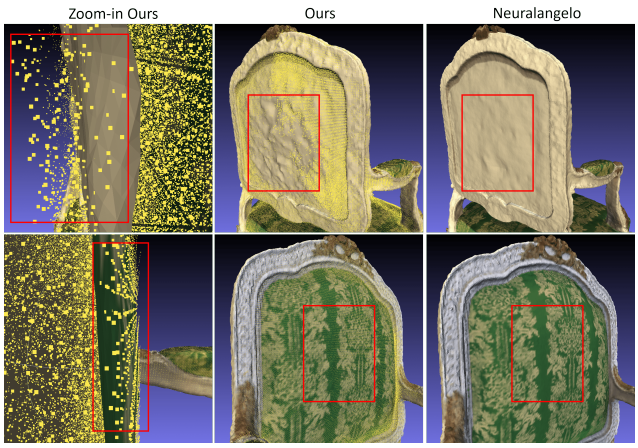

Figure 7. **Effects of erroneous 3DGS**. Red boxes in the first and second row show guidance with erroneous and accurate GS, respectively.

NeRF not only on geometric accuracy, but also on photometric accuracy. Photometric evaluation (PSNR) on the DTU dataset is in Supplementary.

5.4. Ablation Study

We conduct ablation studies (Figure 6) on the influence of three factors - sparse GS embedding fusion, the use of GS for depth images, and the use of GS over point clouds. All of our ablation methods outperform the best baseline, Neuralangelo. Firstly, to prove the importance of "Surface Fusion" over "Dense Fusion", we conduct an experiment to fuse on 4 more closest query points to the anchor point for each ray (5 query points in total). Our results show that fusing only at the anchor point outperforms fusing on more query points closest to the anchor point, which confirms the observation in Figure 3 and proves our proposed "Surface 3DGS Fusion" over "Dense 3DGS Fusion". Secondly, using GS rendered depth outperforms using point cloud rendered depth to find anchor points. This is because the initialized point cloud from MVSNet is noisy, causing the surface anchor point estimation to be inaccurate. In contrast, the depth image from a well-trained GS accurately estimates the anchor points, which again proves that our design has tolerance to noise from the initialized point cloud as shown in Figure 5. Thirdly, our proposed GS fusion outperforms point cloud fusion. We implement a point-cloud based fusion by removing the GS covariance and SH concatenation from the 3DGS Aggregator and cancel the impact of the 3DGS covariance and opacity terms in the distance weight of 3DGS fusion. The results validate the importance of the 3DGS attributes along with the center coordinates to our proposed fusion model.

5.5. Limitation and Future Works

Although our method shows tolerance to noise in small regions from points clouds as shown in Figure 5, it is still sensitive to noise from large regions. Incorrect estimation of the surface along a ray leads to incorrect identification of an anchor point, and subsequently causes errors in the SDF. Figure 7 shows the effect of erroneous GS, where we overlay the GS center (in yellow) to the estimated surface mesh in the first two columns for better visualization. The first row shows the back side of the chair, where GS centers are not well-fitted to the surface in the red box. This leads to a bumpy surface, worse than Neuralangelo which does not use GS. The second row shows the front side of the chair, where GS centers are well-aligned on the surface, which leads to a smooth and accurate surface. Although this proves that our 3DGS fusion method effectively uses prior GS for guidance on SDF, the GS must be accurate, which will be the focus of future work. We build our ideas upon Neuralangelo [20], but our GS aggregation and GS surface fusion methods can be combined with any Neural implicit SDF owing to the low cost of 3DGS training and good performance. We hope to see future work extending our ideas to further improve the accuracy and efficiency of SDF-NeRF.

6. Conclusion

In sum, we proposed a novel Neural Implicit SDF model "SplatSDF", along with a novel architecture-level "Surface 3DGS Fusion" to take advantage of the 3DGS attributes and accurately manipulate SDF embeddings on surface query points. Our SplatSDF significantly improves both the accuracy and efficiency of the SDF-NeRF. It outperforms all SOTA SDF-NeRFs on both geometric and photometric accuracy on two

major datasets, and achieves $>$ 3 times convergence speed compared to the baseline, Neuralangelo [20].

References

[1] Ma Baorui, Han Zhizhong, Liu Yu-Shen, and Zwicker Matthias. Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. In *International Conference on Machine Learning (ICML)*, 2021. 2

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844, 2021. 7, 4

[3] Chao Chen, Yu-Shen Liu, and Zhizhong Han. Gridpull: Towards scalability in learning implicit representations from 3d point clouds. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023. 2

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

[5] Guillaume Coiffier and Louis Béthune. 1-lipschitz neural distance fields. *Computer Graphics Forum*, 2024. 2

[6] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. 2, 3, 5, 7

[7] François Darmon, Bénédicte Bascle, Jean-Clément Devaux, Pascal Monasse, and Mathieu Aubry. Improving neural implicit surfaces geometry with patch warping. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6250–6259, 2022. 2, 7

[8] Qiancheng Fu, Qingshan Xu, Yew-Soon Ong, and Wenbing Tao. Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 7, 3

[9] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. 2024. 7, 4

[10] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*, 2024. 2, 3, 5, 6, 7

[11] Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. Neural 3d scene reconstruction with the manhattan-world assumption. In *CVPR*, 2022. 2

[12] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. 2, 3, 7

[13] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanaes. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 5

[14] Sijia Jiang, Jing Hua, and Zhizhong Han. Coordinate quantized neural implicit representations for multi-view 3d reconstruction. In *IEEE International Conference on Computer Vision*, 2023. 2

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2, 3, 4, 5, 7

[16] C. Koneputugodage, Y. Ben-Shabat, D. Campbell, and S. Gould. Small steps and level sets: Fitting neural surface models with point guidance. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[17] Ki Myung Brian Lee, Zhirui Dai, Cedric Le Gentil, Lan Wu, Nikolay Atanasov, and Teresa Vidal-Calleja. Safe bubble cover for motion planning on distance fields, 2024. 2, 3

[18] Runfa Li, Upal Mahbub, Vasudev Bhaskaran, and Truong Nguyen. Monoselfrecon: Purely self-supervised explicit generalizable 3d reconstruction of indoor scenes from monocular rgb views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 656–666, 2024. 2

[19] Yiming Li, Xuemin Chi, Amirreza Razmjoo, and Sylvain Calinon. Configuration space distance fields for manipulation planning, 2024. 2, 3

[20] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 5, 6, 7, 8, 9, 4

[21] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. In *ACM Trans. Graph.*, 2023. 7, 3, 4

[22] Kehan Long, Hardik Parwana, Georgios Fainekos, Bardh Hoxha, Hideki Okamoto, and Nikolay Atanasov. Neural configuration distance function for continuum robot control, 2024. 2, 3

[23] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, page 163–169, 1987. 2

[24] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 2, 7, 3

[25] Xiaoyang Lyu, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi. 3dgsr: Implicit surface reconstruction with 3d gaussian splatting, 2024. 2, 3, 5, 7, 4

[26] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf:

Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 3, 5, 6, 7, 4

[27] Bailey Miller, Hanyu Chen, Alice Lai, and Ioannis Gkioulekas. Objects as volumes: A stochastic geometry view of opaque solids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 87–97, 2024. 2, 7, 3

[28] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 3, 7, 4

[29] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3d scene reconstruction from posed images. In *ECCV*, 2020. 2

[30] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011. 2

[31] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*, 2021. 2, 7, 3

[32] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. In *Robotics: Science and Systems*, 2022. 2

[33] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[34] Alexander Rich, Noah Stier, Pradeep Sen, and Tobias Höllerer. 3DVNet: Multi-view depth prediction and volumetric refinement. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2021. 2

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 7, 2

[36] Noah Stier, Alexander Rich, Pradeep Sen, and Tobias Höllerer. VoRTX: Volumetric 3d reconstruction with transformers for voxelwise view selection and fusion. In *2021 International Conference on 3D Vision (3DV)*, pages 320–330. IEEE, 2021. 2

[37] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *CVPR*, 2021. 2

[38] Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. Delicate textured mesh recovery from nerf via adaptive surface refinement. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17693–17703, 2023. 7, 3, 4

[39] Vasileios Vasilopoulos, Suveer Garg, Pedro Piacenza, Jinwook Huh, and Volkan Isler. RAMP: Hierarchical Reactive Motion Planning for Manipulation Tasks Using Implicit Signed Distance Functions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023. 2, 3

[40] Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. Neuris: Neural reconstruction of indoor scenes using normal priors. In *European Conference on Computer Vision*, pages 139–155. Springer, 2022. 2

[41] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 2, 7, 3, 4

[42] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. *arXiv preprint arXiv:2206.07850*, 2022. 2, 7, 4

[43] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2

[44] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Pet-neus: Positional encoding triplanes for neural surfaces. 2023. 2, 7, 3, 4

[45] Silvan Weder, Johannes L. Schonberger, Marc Pollefeys, and Martin R. Oswald. NeuralFusion: Online depth fusion in latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3162–3172, 2021. 2

[46] Qianyi Wu, Xian Liu, Yuedong Chen, Kejie Li, Chuanxia Zheng, Jianfei Cai, and Jianmin Zheng. Object-compositional neural implicit surfaces. In *European Conference on Computer Vision*, pages 197–213. Springer, 2022. 2

[47] Qianyi Wu, Kaisiyuan Wang, Kejie Li, Jianmin Zheng, and Jianfei Cai. Objectsdf++: Improved object-compositional neural implicit surfaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 2

[48] Haodong Xiang, Xinghui Li, Xiansong Lai, Wanting Zhang, Zhichao Liao, Kai Cheng, and Xueping Liu. Gaussianroom: Improving 3d gaussian splatting with sdf guidance and monocular cues for indoor scene reconstruction. *arXiv preprint arXiv:2405.19671*, 2024. 2, 3, 5

[49] Yuting Xiao, Jingwei Xu, Zehao Yu, and Shenghua Gao. Debsdf: Delving into the details and bias of neural indoor scene reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024. 2, 7, 3

[50] Cao Xu and Taketomi Takafumi. Supernormal: Neural surface reconstruction via multi-view normal integration. In *CVPR*, 2024. 2

[51] Hongyi Xu, Thiemo Alldieck, and Cristian Sminchisescu. H-NeRF: Neural radiance fields for rendering and temporal reconstruction of humans in motion. In *Neural Information Processing Systems*, 2021. 2

[52] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 4, 5, 1

[53] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. *Computer Vision and Pattern Recognition (CVPR)*, 2020. 5, 6, 1

[54] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 2, 7, 3, 4

[55] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 7, 3, 4

[56] Mulin Yu, Tao Lu, Linning Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai. Gsdf: 3dgs meets sdf for improved rendering and reconstruction, 2024. 2, 3, 5, 7

[57] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 7

[58] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024. 2, 3, 5, 7

[59] Jingyang Zhang, Yao Yao, and Long Quan. Learning signed distance field for multi-view surface reconstruction. *International Conference on Computer Vision (ICCV)*, 2021. 7, 2

[60] Jingyang Zhang, Yao Yao, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Critical regularizations for neural surface reconstruction in the wild. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6260–6269, 2022. 2, 7, 4

[61] Yongqiang Zhang, Zhipeng Hu, Haoqian Wu, Minda Zhao, Lincheng Li, Zhengxia Zou, and Changjie Fan. Towards unbiased volume rendering of neural implicit surfaces with geometry priors. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4359–4368, 2023. 2, 7, 3

SplatSDF: Boosting Neural Implicit SDF via Gaussian Splatting Fusion

Supplementary Material

A. Relations to "PointNeRF"

One potential alternative to our method is to use PointNeRF [52], which takes point clouds to boost NeRF. Although the two works share the high-level idea of "fusing prior 3D primitives to boost the target model", PointNeRF fails on our task. There are major differences persisting in the roots of the two methods. In this section, we summarize the reasons why our SplatSDF is better than PointNeRF.

- **Better fusion strategies: "Dense VS Surface".** PointNeRF's fusion strategy does not work for SDF-NeRF. PointNeRF proposes "Dense Point Fusion", which considers valid neighbor points for all query points on the rays. However, as shown in Figure 3, even though "Dense Fusion" might work for RGB synthesis in PointNeRF, it does not work for SDF estimation. Thus, we propose our novel "Surface Fusion" only at the anchor point. Our fusion strategy gets rid of the bumpy and noisy artifacts on surfaces, and is far more efficient than densely fusing all query points as PointNeRF - The efficiency improvement is further illustrated in Section B.
- **Better purposes: Solely RGB VS RGB+SDF.** PointNeRF is a traditional NeRF that aims at RGB synthesis, while our SplatSDF is Neural Implicit SDF for continuous SDF estimation. In other words, PointNeRF can only synthesize RGB images, our SplatSDF can achieve both continous SDF estimation and RGB synthesis.
- **Better priors: 3D points VS 3DGS.** We use 3DGS as "fusing priors" with more attributes (opacities and shapes of 3D Gaussian primitives) than 3D points. The 3DGS opacity shares confidence priors on how likely the densities are fused to the anchor point - The higher density it carries, the more likely it is at surface. The 3DGS shape shares explicit information on the surface normals and scope of influence of Gaussian primitives, whereas point clouds (Gaussian centers) only occupies discrete and sparse coordinates without further information on shape and surface. As shown in figure 6 and table 3, leveraging these additional attributes of 3DGS enables a faster and better convergence for geometry than only using Gaussian primitive centers (point cloud).
- **Better fusing functions: Inverse distance weighting VS 3D Gaussian weighting.** The use of 3DGS over point clouds allows the novel design of our fusing functions and 3DGS aggregator. The weight term in PointNeRF's fusing function is a basic inverse distance widely used in scattered data interpolation, while we leverage a 3D Gaussian distribution for the weight term in eq. (3) inspired by the 2D Gaussian weights of 3DGS. We also take GS opacity into our novel fusing function as shown in eq. (3), as opposed to PointNeRF's confidence score obtained from the pretrained MVSNet [53].

B. Sparse KNN Details

Our goal is to take advantage of nearby GS embeddings for the SDF-MLP regression at query points. Since densely searching nearby GS for all query points leads to an unacceptable complexity, we adopt a sparse KNN from PointNeRF [52]. For each view, we cast M rays and sample N query points along each ray. Supposing the number of GS is $|\mathcal{G}|$, a brute-force standard KNN on all query points would lead to a complexity of $O(MN|\mathcal{G}|)$. We decrease the complexity by hashing and voxelizing the 3D space into L^3 unit cells and reduce the complexity to $O(\frac{MN|G|}{L^3})$. Specifically, we first hash all query points and GS to unit cells, which leads to $O(|\mathcal{G}|+MN)$ complexity. Since the average number of GS in each unit cell is $\frac{|\mathcal{G}|}{L^3}$ and each query point only searches for neighbor GS within the same cell, the complexity of distance computations is $O(\frac{MN|\mathcal{G}|}{L^3})$. The overall complexity is:

$$O(MN+|\mathcal{G}|)+O(\frac{MN|\mathcal{G}|}{L^3}logK)\approx O(\frac{MN|\mathcal{G}|}{L^3}) \quad (10)$$

where K is the maximum number of nearest neighbors being considered, $\log K$ is the sorting complexity after the distance computation, but can be canceled since K is a much smaller constant than the other parameters. The hashing pre-processing is canceled because $L^3 \ll MN|\mathcal{G}|$. Consequently, the sparse KNN significantly reduces the complexity from $O(MN|G|)$ to $O(\frac{MN|\mathcal{G}|}{L^3})$, especially when separating fine unit cells with a high resolution L. In practice, we also extend the search to neighboring cells of the center cell that the query point lies in, but we ignore the complexity since the number of neighbors are commonly set to a very small number.

However, the $O(\frac{MN|\mathcal{G}|}{L^3})$ complexity is still inefficient to train an SDF-NeRF which has an extra SDF-MLP compared to the standard NeRF. As discussed in the main paper, we further reduce the complexity to $O(\frac{M|\mathcal{G}|}{L^3})$ with our proposed novel "Surface 3DGS Fusion" by only fusing on one query point per ray at the anchor point.

C. Ablation Study results on "Lego"

We show the detailed Chamfer Distance values in mm for Figure 6 in Table 3.

Epochs	Neural Angelo	5pt pcdrender gsfusion	1pt pcdrender gsfusion	5pt gsrender gsfusion	1pt gsrender pcdfusion	1pt gsrender gsfusion
100	3.11	2.59	2.92	2.69	2.81	**2.81**
200	2.91	2.60	3.01	2.50	2.21	**2.54**
300	2.63	2.71	2.31	2.27	2.60	**2.84**
400	2.62	2.47	2.37	2.17	2.08	**2.28**
500	2.24	2.23	2.12	1.92	2.00	**2.14**
600	2.15	2.08	1.83	1.84	1.89	**1.79**
700	1.97	1.94	1.73	1.68	1.74	**1.71**
800	1.96	1.93	1.84	1.68	1.76	**1.61**
900	1.88	1.69	1.72	1.54	1.62	**1.48**
1000	1.77	1.81	1.70	1.49	1.57	**1.41**
1100	1.76	1.72	1.56	1.54	1.52	**1.42**
1200	1.73	1.70	1.54	1.45	1.47	**1.39**
1500	1.71	1.62	1.54	1.42	1.42	**1.35**
2000	1.62	1.56	1.43	1.38	1.34	**1.27**
2500	1.54	1.52	1.38	1.34	1.35	**1.24**
3000	1.60	1.49	1.34	1.34	1.26	**1.19**

Table 3. **Values of Figure 6,** Comparison on geometric accuracy (Chamfer Distance in mm) at different training epochs on "Lego".

Epochs	Neuralangelo	SplatSDF (Ours)
500	1.26	**1.19**
1000	0.99	**0.91**
1500	0.88	**0.80**
2000	0.88	**0.76**
2500	0.84	**0.75**
3000	0.79	**0.76**
3500	0.79	**0.74**
4000	0.79	**0.73**
4500	0.78	**0.73**
5000	0.78	**0.72**

Table 4. **Values of Figure 9,** Comparison on geometric accuracy (Chamfer Distance in mm) at different training epochs on "Ship".

D. Results on NeRF Sythetic Dataset

We show the qualitative and quantitative comparison of "Lego" in the main paper. In this section, we show the qualitative and quantitative comparison on another difficult scene from the NeRF Synthetic Dataset, "Ship".

Figure 8 shows visual comparisons of our SplatSDF to the best baseline Neuralangelo at different training epochs. Figure 9 and Table 4 show the numerical comparison in Chamfer Distance. Both the qualitative and quantitative results prove our faster and better convergence to complex shapes.

E. Results on DTU Dataset

In the main paper, we show the geometric evaluation (in Chamfer Distance) on the DTU Dataset. In this section, we show the visual comparison in Figure 10 and photometric evaluations in Table 5. We train Neuralangelo by ourself since there are no checkpoints released by Neuralangelo. Both Neuralangelo and our SplatSDF are trained to fully converge. For our SplatSDF, we use the same checkpoint as the evaluation in Table 1. Since visual differences are difficult to observe in most of the cases, we select some representative details as shown in Figure 10. Neuralangelo fails to reconstruct some difficult details, while our SplatSDF can reconstruct some difficult details at a high quality. Table 5 shows the geometric evaluation in PSNR on the DTU Dataset. We cannot reproduce the same results that Neuralangelo's paper proposes, so we also show Neuralangelo's result trained by ourself. Our SplatSDF obtains a higher PSNR than Neuralangelo on all of the scenes.

F. Training and Inference Efficiency

We build our SplatSDF upon Neuralangelo [20], where training speed on a single Nvidia 3090Ti GPU is $\sim$ 7FPS with the default training settings - 512 rays per image and 128 query points per ray. When using our Surface 3DGS Fusion, the speed is slightly affected by the number of valid 3DGS being fused and valid rays. The normalized unit sphere's radius is 1. We empirically set the voxel size of sparse KNN to be 0.005, and K is 4. The training speed of our SplatSDF is $\sim$ 5.5 FPS on a single Nvidia 3090Ti. Since we only do 3DGS Fusion for training, we do not change the efficiency of the original SDF-NeRF, and in this work, the inference speed is exactly the same as Neuralangelo.

G. Scene Selections

In this section, we explain more details of the scenes we choose for evaluation. For a fair comparison, since we completely build our algorithm on Neuralangelo [20], we train Neuralangelo to make sure we reproduce comparable results that the Neuralangelo paper proposes as Neuralangelo does not release their well-trained checkpoints. Since the DTU Dataset only scans on one side of the object by swinging at a limited region, SDF-NeRF has no access to the back (or unseen) sides of the object, and it is necessary to crop Neuralangelo's results to get comparable results as their paper proposes. Please refer to the Neuralangelo official GitHub repository to understand the reason for cropping https://github.com/NVlabs/neuralangelo/issues/93. We use 12 but not all of the commonly used 15 scenes for the DTU evaluation because we cannot reproduce comparable results using their code implementation on scene 97, 118 and 122 that Neuralangelo proposes.

H. Resources of Comparison

For the comparison results in Table 1 and Table 2, we use the results from the latest paper or the original paper directly. In Table 1, for SDF-NeRF methods, we get the results of NeRF [26], VolSDF [54], NeuS [41], HF-NeuS [42], RegSDF [60], NeuralWarp [7] and Neuralangelo [20] from Neuralangelo. We get the results of COLMAP [35], MVSDF[59], NeuS-12 [41] from TUVR [61]. Furthermore,

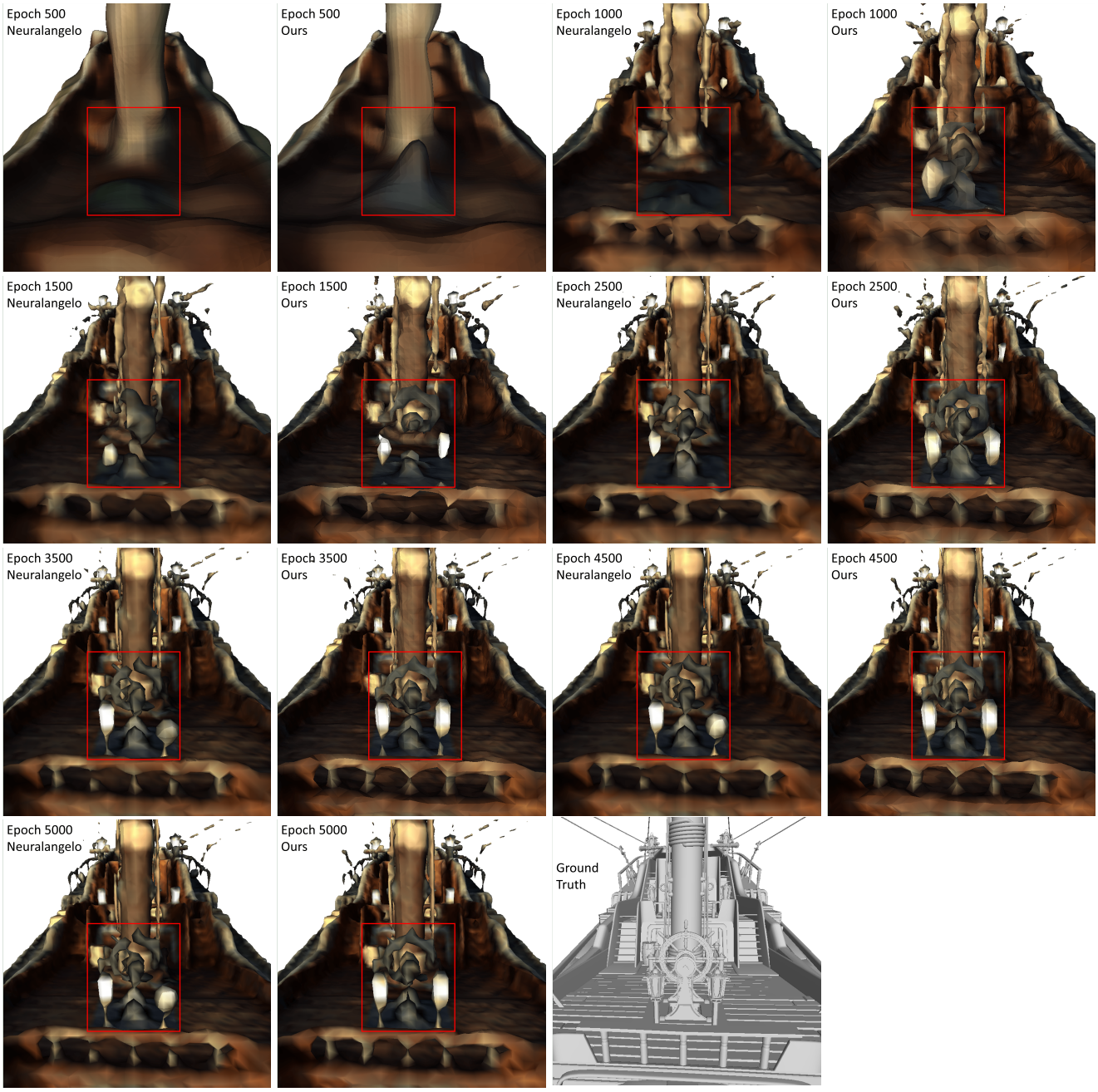

Figure 8. . Similar to Figure 1, we show comparison at different training epochs on "Ship". SplatSDF makes it easier to converge to complex geometry (like the white lamp and the helm in the ship), achieves greater geometric accuracy, and faster convergence compared to the best baseline, Neuralangelo. Color is not available for the ground truth mesh.

the results for TUVR [61] and DebSDF are from [49]. The results UNISURF [31] and Geo-NeuS [8] are from Geo-NeuS. The results of OAV [27] and PET-NeuS [44] are from their own papers. For GS-based Surface Reconstruction methods, we get Scalffold-GS [24] from GSDF [56]. We get 3DGS [15], SuGAR [10], [6], 2DGS [12] and Gaussian Opacity Field [58] from Gaussian Opacity Field. For the two concurrent GS-based Surface Reconstruction 3DGSR [25] and GSDF [56], we get the results from their original paper. In Table 2, since Neuralangelo does not evaluate on NeRF Synthetic dataset, we implement Neuralangelo and our SplatSDF by ourself. For the Chamfer Distance in the top part, we get the results of VolSDF [54], NeuS [41], NeRO [21], BakedSDF [55], NeRF2Mesh [38], Relightab-

Scan ID	24	37	40	55	63	65	69	83	105	106	110	114	Mean
RegSDF [60]	24.78	23.06	23.47	22.21	28.57	25.53	21.81	28.89	27.91	24.71	25.13	26.84	25.24
NeuS [41]	26.62	23.64	26.43	25.59	30.61	32.83	29.24	33.71	31.97	32.18	28.92	28.41	29.18
VolSDF [54]	26.28	25.61	26.55	26.76	31.57	31.50	29.38	33.23	32.13	33.16	31.49	30.33	29.83
NeRF [26]	26.24	25.74	26.79	27.57	31.96	31.50	29.58	32.78	32.08	33.49	31.54	31.00	30.02
Neuralangelo [20]	30.64	27.78	32.70	34.18	35.15	35.89	31.47	36.82	35.92	36.61	32.60	31.20	33.41
Neuralangelo [20] (we trained)	29.65	30.45	28.90	28.42	30.24	28.36	28.15	29.65	28.66	28.57	29.00	28.38	29.04
Ours	**29.63**	**30.39**	**29.75**	**28.44**	**31.24**	**28.74**	**29.00**	**29.61**	**29.41**	**28.83**	**29.45**	**28.68**	**29.43**

Table 5. **Photometric evaluation on DTU dataset in PSNR.** The results of the first 5 methods (RegSDF, NeuS, VolSDF, NeRF and Neuralangleo) are directly from Neuralangelo's paper. Neuralangelo did not release their trained checkpoints, we retrain by ourself but cannot reproduce their paper's results using their code. The bottom two rows show the results of Neuralangelo trained by ourself, and our SplatSDF.

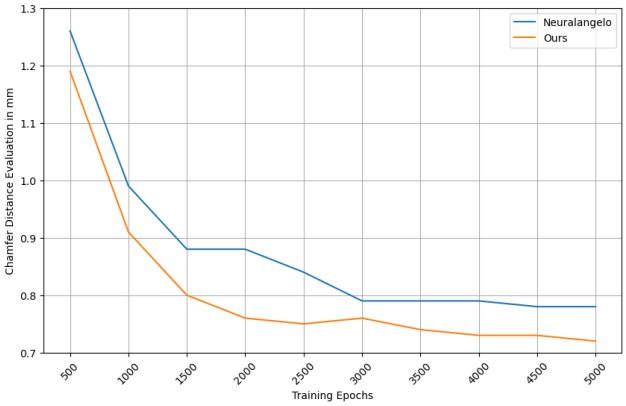

Figure 9. **Comparison of geometric accuracy on "Ship"**. Our method achieves faster convergence to better accuracy than the baseline "Neuralangelo". Exact values are summarized in the table .

leG [9], and 3DGSR [25] from 3DGSR. We get the result of HF-NeuS [42] from its paper. For the PSNR at the bottom part, we get the results of NeRF [26], VolSDF [54], NeuS [41], HF-NeuS [42], PET-NeuS [44] from PET-NeuS. We get the results of NeRO [21], BakedSDF [55], NeRF2Mesh [38], Mip-NeRF [2], 3DGS [25] and Instant-NGP [28] from 3DGSR [25].

Even thought very few previous SDF-NeRF works release their per-scene training checkpoints, we will release our checkpoints along with the evaluation code and implementation details.

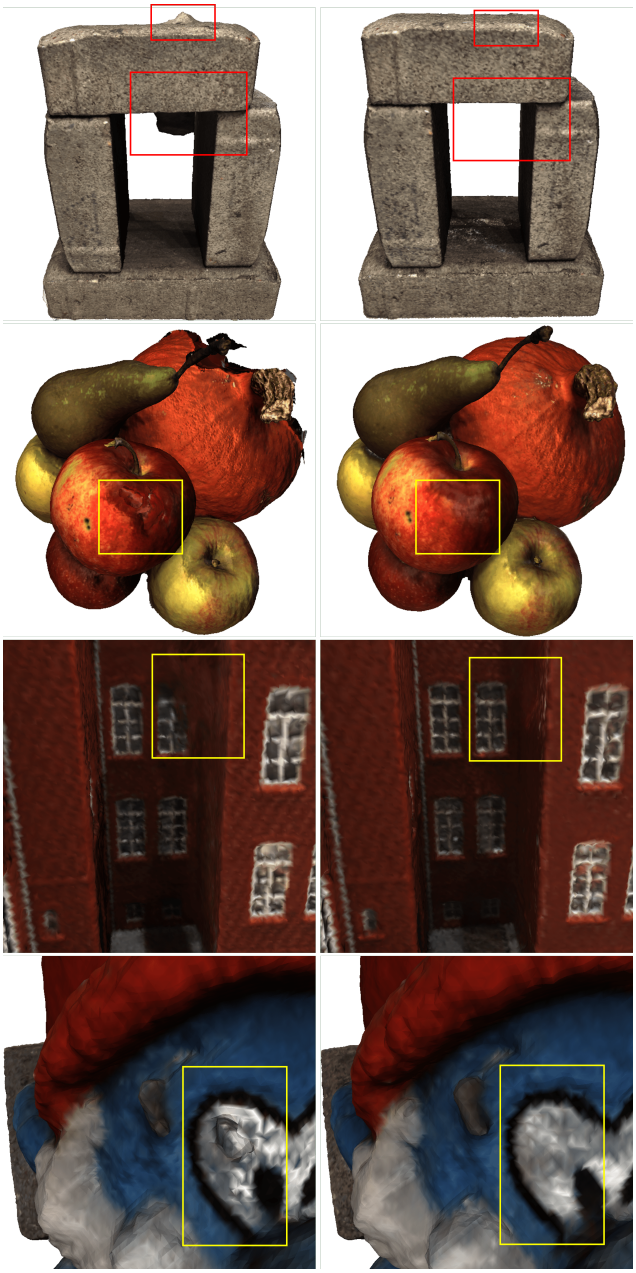

Figure 10. **Qualitative comparisons on the DTU Dataset.** Left column: Neuralangelo [20]. Right column: Our SplatSDF. We use the same models as Quantitative results from Table 1 to generate the visual results. Zoom in to check the details in the red or yellow bounding boxes.