# $\alpha$Surf: Implicit Surface Reconstruction for Semi-Transparent and Thin Objects with Decoupled Geometry and Opacity

Tianhao Wu
University of Cambridge

Hanxue Liang
University of Cambridge

Fangcheng Zhong
University of Cambridge

Gernot Riegler
Unity

Shimon Vainer
Unity

Jiankang Deng
Imperial College London

Cengiz Oztireli
University of Cambridge

## Abstract

*Implicit surface representations such as the signed distance function (SDF) have emerged as a promising approach for image-based surface reconstruction. However, existing optimization methods assume opaque surfaces and therefore cannot properly reconstruct translucent surfaces and sub-pixel thin structures, which also exhibit low opacity due to the blending effect. While neural radiance field (NeRF) based methods can model semi-transparency and synthesize novel views with photo-realistic quality, their volumetric representation tightly couples geometry (surface occupancy) and material property (surface opacity), and therefore cannot be easily converted into surfaces without introducing artifacts. We present αSurf, a novel scene representation with decoupled geometry and opacity for the reconstruction of surfaces with translucent or blending effects. Ray-surface intersections on our representation can be found in closed-form via analytical solutions of cubic polynomials, avoiding Monte-Carlo sampling, and are fully differentiable by construction. Our qualitative and quantitative evaluations show that our approach can accurately reconstruct translucent and extremely thin surfaces, achieving better reconstruction quality than state-of-the-art SDF and NeRF methods.*

## 1. Introduction

Recovering object geometry as surfaces from RGB images is a long-standing problem in computer vision, with numerous practical applications such as photogrammetry, 3D asset creation, and custom 3D fabrication. Traditional approaches rely on Structure-from-Motion [44] and Multi-View Stereo [46] pipelines to first reconstruct a 3D point set of the scene to which surfaces can be fitted [21, 24]. Differentiable rendering techniques have emerged as a more versatile reconstruction procedure. With properly derived gradients, these techniques can simultaneously learn both geometry and appearance by minimizing the error of RGB renderings. This significantly loosens the restrictions on the choice of geometric representations to be learned. In particular, implicit surface representations [19, 53, 59, 64], *i.e.* level sets of a scalar field such as a signed distance function (SDF), show promising results in surface reconstruction due to their robustness to complex geometry and topology.

One open challenge that has not been adequately addressed in this domain is reconstructing implicit surfaces that exhibit translucent effects. The prevailing assumption in earlier works is that the surface is opaque throughout, and differentiable rendering techniques for implicit surfaces only examine the intersection of rays and the nearest surface [19, 30, 34]. As a result, the forward rendering process in those studies cannot simulate scenes with non-opaque effects, leading to an inability to reconstruct their surfaces.

Modeling opaqueness is not solely crucial for reconstructing translucent surfaces, but is also necessary for the recovery of extremely thin surfaces with sub-pixel silhouette. As illustrated in Figure 2, when rendering a thin structure that only partially occupies a pixel with an insufficient number of sampled rays, opaque foreground objects must be treated as semi-transparent in order to achieve an accurate pixel color blending the foreground and background. Methods that neglect this feature may fail to capture the correct reconstruction of thin structures; see Figure 1.

Differentiable volume rendering techniques [32] have demonstrated considerable success in rendering translucent and thin objects by integrating radiance along a ray through an entire scene. Their geometric representation is a density field where a surface can be implicitly defined by applying a density threshold. However, density represents both occupancy (geometry) and transparency (material) in a tightly

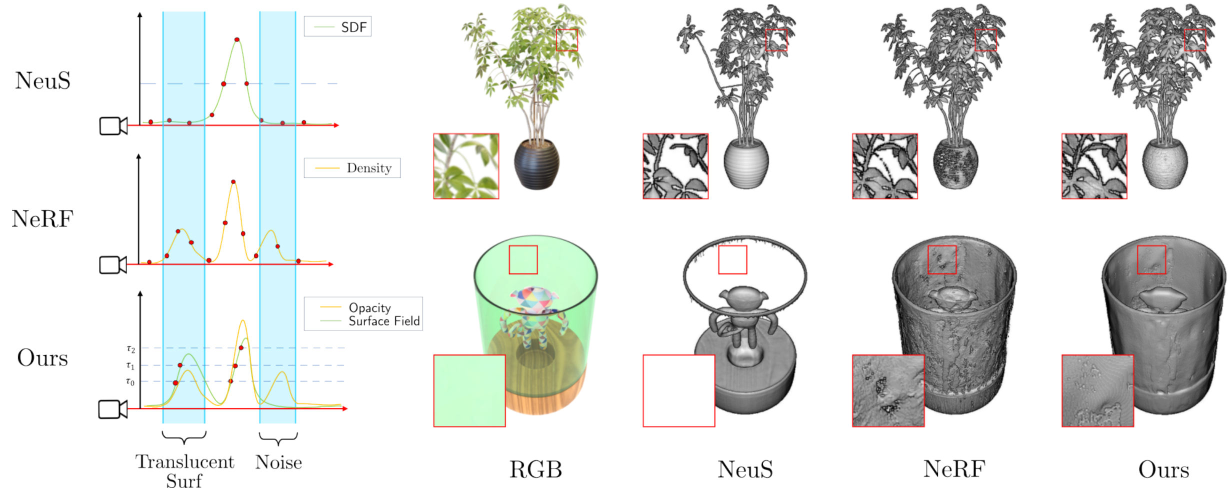

Figure 1. **Teaser.** We illustrate the representations of NeuS, NeRF, and our method, as well as reconstructed surfaces. NeuS [54] uses SDF to optimize for opaque surfaces and hence misses translucent or thin surfaces with blending effects in the reconstruction. NeRF methods such as Plenoxels [42] can represent semi-transparency with density field, but as density couples both occupancy and opacity, surfaces extracted from it would contain holes or redundant surface floater. In contrast, our approach models decoupled surface and opacity fields. We use a surface field without Eikonal constraint and multiple level sets $\tau_0, \tau_1, \ldots$ to model geometry with different levels of confidence and opacity, and utilize a closed-form intersection formula to enable differentiable rendering, and hence can accurately reconstruct surfaces exhibiting semi-transparency.

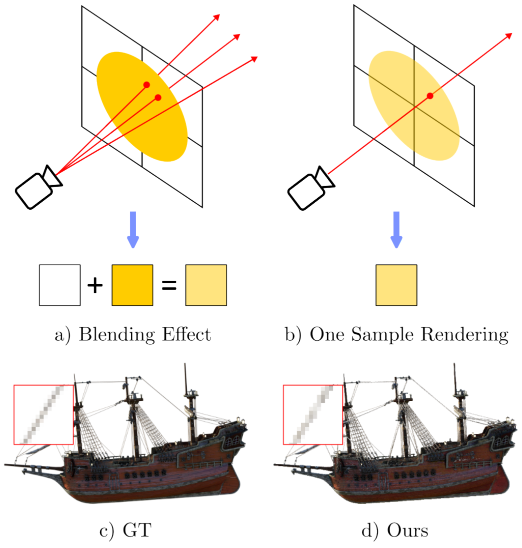

Figure 2. **Blending effect.** a) Real-world capture takes incoming light from multiple rays per pixel. Pixels that are partially occupied by an opaque object are therefore rendered as a mixture of the object and background color. b) With one-sample rendering in reconstruction, it becomes necessary for extremely tiny objects to be modeled as semi-transparent for the pixel color to match the ground truth. c), d) Our representation is fully capable of representing this phenomenon, leading to the accurate surface reconstruction of thin structures.

coupled manner, posing a non-trivial challenge in choosing the threshold for surface extraction. A high threshold may exclude translucent surfaces from the reconstruction, whereas a low threshold results in the erroneous reconstruction of density floaters and redundant surfaces that are not part of the desired reconstruction. NeuS [54] attempts to alleviate this issue by combining SDF with volume rendering, but it still assumes an opaque surface at the end of optimization and hence fails to reconstruct surfaces with semi-transparency. Figure 1 illustrates failure cases of both types of methods on translucent or thin surfaces. Recent Gaussian Splatting methods [12, 22] uses anisotropic 3D Gaussians to achieve efficient rendering, but they similarly ignore the occupancy-transparency ambiguity and only retain opaque Gaussians for surface extraction, and hence cannot reconstruct thin or translucent surfaces.

Two key requirements need be fulfilled for faithfully reconstructing surfaces of translucent and thin objects: 1) a representation that explicitly decouples geometry and materials; and 2) a differentiable rendering pipeline that considers more than the nearest ray-surface intersection, while also enabling gradient flows to both surface and opacity. To this end, we introduce αSurf, a novel surface representation based on a grid structure without neural networks. We use separate values on the grid to represent geometry, opacity, and appearance. We define the surface as multiple level sets of a continuous scalar field, where the field itself is given by a trilinear interpolation of the grid values. Unlike previous methods, our representation does not require the scalar field to be an SDF subject to the Eikonal constraint. An important property of our approach is that the *exact* intersection points between a ray and all the surfaces, regardless of whether they are opaque or transparent, can be determined by analytically solving a cubic polynomial. The *closed-form* solution allows for full *differentiability* to both geometry and material in our forward rendering process, which simulates the semi-transparency effects via alpha compositing of intersection points.

We further propose a series of initialization and optimization strategies that are designed to facilitate efficient and accurate reconstruction. The total training time of our method is around 30 minutes. We evaluate our method on an extended version of the NeRF synthetic dataset [32], which contains 8 original scenes and 16 additional objects with challenging thin structures or translucent materials. We show that our method is capable of reconstructing surfaces with better quality than the existing SDF and NeRF based methods.

In summary, our contributions are: 1) A novel grid-based scene representation for implicit surface reconstruction, specialized for translucent and thin objects. It incorporates a closed-form and differentiable evaluation of all surface intersections along the ray, and properly decouples surface geometry and opacity. 2) An initialization scheme utilizing fast Plenoxels [42] training and optimization with truncated volume rendering and surface smoothness constraint for efficient training and accurate reconstruction. 3) We show superior reconstruction quality compared to state-of-the-art methods on synthetic and real-world scenes featuring thin and translucent objects.

## 2. Related Works

**Neural Radiance Fields** NeRF [32] is a plenoptic function of volume density and view-dependent appearance. Its differentiable volume rendering allows robust image-based 3D reconstruction and motivated many works in high-quality novel view synthesis [2, 52, 57, 63], 3D asset synthesis and editing [8, 14, 37, 47], few-shot reconstruction [18, 29, 58, 62], and efficient rendering [22, 33, 39, 42, 51, 61]. Despite the outstanding novel view synthesis performance, their geometry tends to produce artifacts such as sparse density floaters and inner volume [2, 52, 63]. Direct surface extraction on the density field hence suffers from those artifacts, whereas the depth extraction method does not guarantee complete surfaces and requires additional surface reconstruction [7, 10, 21].

**Signed Distance Field** SDF has been extensively employed with differentiable rendering methods to reconstruct surfaces from multi-view images. IDR [59] proposes a differentiable sphere-tracing algorithm and optimizes the surface together with a volumetric BRDF. VolSDF [60] maps SDF to volume density via Laplacian CDF and optimizes the SDF via volume rendering. NeuS [54] similarly maps an SDF to unbiased weights in the volume rendering equation via a logistic sigmoid function. NeuS has motivated several further applications in different areas, such as sparse view surface reconstruction [31, 40], fast reconstruction [25, 41, 55], and finer details modeling [3, 11, 28, 56, 56]. A crucial and common limitation in existing SDF optimization methods is the assumption of surface opaqueness. Even the methods that utilize volume rendering in surface reconstruction still enforce convergence to opaque surfaces. Hence, they cannot properly reconstruct semi-transparent surfaces and suffer from thin structures with strong blending effects.

**Transparent Object Reconstruction** Many works have tried to address the problem of transparent object reconstruction [15, 26, 27]. They focus on thick and fully transparent objects, and aim to reconstruct the shape by solving the complicated light transportation within the objects. Most methods require additional supervision such as environment matting, which can only be obtained with checkerboard-patterned backgrounds. In comparison, we reconstruct thin translucent surfaces from RGB images alone with no other supervision. NeRRF [4] is a method that similarly reconstructs surfaces from RGB images and mask supervisions.

## 3. Method

Given a set of multi-view posed RGB images, our goal is to recover an implicit surface of the scene objects, particularly in cases involving translucent or thin surfaces. Towards this, we propose αSurf, a grid-based representation where the grid contains values pertaining to the surface's geometry and material properties (Figure 3a). This enables a closed-form evaluation of all the ray-surface intersections (Figure 3b), and thus a fully differentiable alpha composition (Figure 3c) to render the intersection points. Our method pre-trains Plenoxels [42] to efficiently initialize a coarse surface (Figure 3d), and through the optimization of a photometric loss and additional regularizations, we are able to reconstruct clean and accurate surfaces (Figure 3e).

### 3.1. Representation

**Surface** Following voxel-based representations [25, 42], we represent the surface as the level sets of a continuous scalar field $\delta : \mathbb{R}^3 \rightarrow \mathbb{R}$. For each spatial coordinate $\mathbf{x}$ within a voxel $v_{\mathbf{x}}$, the value of the scalar field $\delta(\mathbf{x})$ is obtained by the trilinear interpolation of scalars stored at the voxel vertices:

$$\delta(\mathbf{x}) = \text{trilerp}(\mathbf{x}, \{\hat{\delta}_i\}_{i=1}^{8}) \tag{1}$$

where $\{\hat{\delta}_i\}_{i=1}^{8}$ are the surface scalars stored at its eight adjacent vertices. The surface is then implicitly defined as level

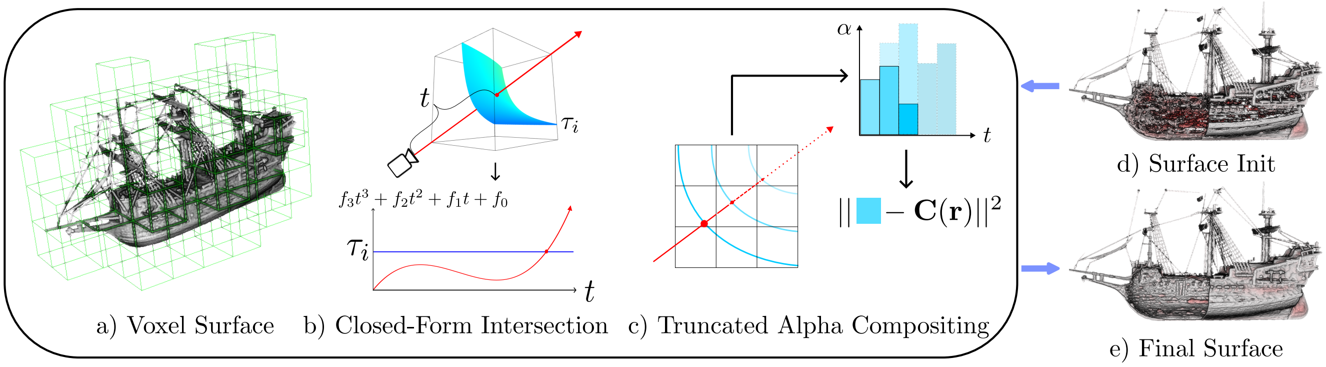
t
$\tau_i$
$f_3t^3 + f_2t^2 + f_1t + f_0$
$\tau_i$
t
α
t
$||$ 🟦 $- \mathbf{C}(\mathbf{r})||^2$
a) Voxel Surface
b) Closed-Form Intersection
c) Truncated Alpha Compositing
d) Surface Init
e) Final Surface

Figure 3. **Method.** a) Our surface representation is based on a voxel grid storing explicit values, without neural networks; see Section 3.1. b) We utilize a closed-form and differentiable method to compute ray-surface intersection. This is achieved by solving a cubic polynomial of depth $t$ with known parameters $f_0, ..., f_3$ and $\tau_i$; see Section 3.2. c) We incorporate surface-specific regularization such as truncated alpha compositing to obtain clean and accurate surfaces; see Section 3.3. d) We utilize a coarse initialization via Plenoxels [42] to start with roughly correct yet noisy surfaces. e) The optimization results in clean and complete surfaces in the end.

sets on the scalar field. Specifically, given a set of $n$ constants $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^{n}$ which we refer to as *level values*, the surface $\mathcal{S}$ is defined as:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 | \exists \tau \in \boldsymbol{\tau} : \delta(\mathbf{x}) = \tau\} . \tag{2}$$

The cardinality $n$ and values of $\boldsymbol{\tau}$ are determined through hyperparameter tuning. Note that this implicit surface field does not model distance to the closest surface as in SDF, therefore, it is not subject to the Eikonal constraint. The motivation behind the multi-level set is tightly related to our NeRF initialization strategy and is further explained in Sec 3.3.

**Opacity and Appearance** Within the same voxel grid, we also model the surface opacity denoted as $\alpha(\mathbf{x})$ and view-dependent appearance denoted as $\mathbf{c}(\mathbf{x}, \mathbf{d})$ in the same explicit style as in Plenoxels [42], which represents $\mathbf{c}(\mathbf{x}, \mathbf{d})$ as coefficients of the spherical harmonic (SH) function that maps view direction $\mathbf{d}$ to radiance. Trilinear interpolation is applied to obtain opacity and SH coefficients at surface locations. Note that although $\alpha(\mathbf{x})$ is essentially defined across all valid voxels in the 3D space, it only represents surface material property rather than geometry, and hence is only meaningful where surface exists.

### 3.2. Differentiable rendering

A key feature in the rendering process of our representation is that it does not involve any Monte-Carlo sampling as in NeRF or sphere tracing as in SDF-based methods. Instead, it relies on ray-voxel traversal and directly takes samples at the ray-surface intersections found through a *closed-form* and fully *differentiable* function. Specifically, for each camera ray with origin $\mathbf{o}$ and direction $\mathbf{d}$, we first determine the set of voxels it traverses through and substitute the ray equation $\mathbf{r}(t) = \mathbf{x} = \mathbf{o} + t\mathbf{d}$ into Eq. 18. for each voxel $v_{\mathbf{x}}$. By setting $\delta(\mathbf{x})$ to each level set value $\tau_i$, we obtain a cubic polynomial $\tau = f_3t^3 + f_2t^2 + f_1t + f_0$ with a single unknown $t$. $f_0, \ldots, f_3$ are known coefficients obtained from camera origin $\mathbf{o}$, ray direction $\mathbf{d}$ and voxel surface scalars $\{\hat{\delta}\}_{i=1}^{8}$. We refer to the supplementary materials for a detailed derivation. The real roots of this cubic polynomial can then be found in closed-form via Vieta's approach [38], which minimizes the error caused by the numerical precision issues. Roots that give intersections outside of each corresponding voxel $v_{\mathbf{x}}$ are deemed invalid and removed. The remaining intersections are ordered from near to far and taken as samples for rendering. For all intersection points $\{t_i\}$ along the ray, we obtain their surface opacity $\alpha$ and view-dependent radiance $\mathbf{c}$ through trilinear interpolation and evaluating the SH function, and then perform alpha compositing to render the pixel color:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_\alpha(t_i)\alpha(t_i)\mathbf{c}(t_i) \tag{3}$$

$$T_\alpha(t_i) = \prod_{j=1}^{i-1}(1 - \alpha(t_j)) \tag{4}$$

where we simplify our notation as $\alpha(t_i) \equiv \alpha(\mathbf{r}(t_i))$ and $\mathbf{c}(t_i) \equiv \mathbf{c}(\mathbf{r}(t_i), \mathbf{d})$. As the samples are taken analytically through cubic polynomial root solving, which is a fully differentiable function, we naturally have gradients from the photometric loss back to the implicit surface field, without requiring any approximation or re-parameterization trick.

### 3.3. Optimization

**NeRF Initialization** One advantage of our representation is that we can easily initialize coarse surfaces from a pre-trained NeRF . In practice, we use Plenoxels [42], a grid-based NeRF method that can be trained efficiently within 10 minutes. After fitting a NeRF, we obtain a density field $\sigma(\mathbf{x})$ from which we select a set of raw level values $\boldsymbol{\tau}_\sigma$ to define the initial surfaces. The values of $\boldsymbol{\tau}_\sigma$ are selected by first determining an upper bound of the initialized density field, then evenly dividing the range by $n$ level values. The number of level sets $n$ is determined by hyperparameter-tuning. We then normalize the density $\sigma(\mathbf{x})$ to be used as our initial surface scalars $\delta(\mathbf{x})$, and normalize the raw level values to be used as our level values:

$$\delta(\mathbf{x}) = \frac{\sigma(\mathbf{x}) - \tilde{\tau}_\sigma}{\overline{||\nabla\sigma||}}, \boldsymbol{\tau} = \left\{ \frac{\tau_\sigma - \tilde{\tau}_\sigma}{\overline{||\nabla\sigma||}} \mid \tau_\sigma \in \boldsymbol{\tau}_\sigma \right\} \tag{5}$$

where $\tilde{\tau}_\sigma$ is the median of the chosen raw level values, $\overline{||\nabla\sigma||}$ is the average over the norms of the finite difference gradient of the voxelated density field. $\overline{||\nabla\sigma||}$ is used to keep the initialized surface field within a constant range to make optimization easier. Note that $\boldsymbol{\tau}_\sigma$ is a hyperparameter and is defined only to make the selection of initial surface level values more convenient, whereas $\boldsymbol{\tau}$ is the actual level set values of the implicit surface field used throughout the optimization.

Although in theory, a single level set is sufficient to represent the surfaces, our multi-level set initialization scheme allows the geometric information from NeRF to be maximally preserved and inherited to the surface field – in NeRF, high density regions represent opaque geometries with high confidence, whereas low density regions can be either low-confidence surfaces or translucent surfaces and hence are ambiguous. Multi-level sets initialization therefore simultaneously captures high-confidence opaque geometry with higher level values, and translucent geometry or low-confidence noise region with lower level values. Later optimization can then easily identify and remove those redundant noise surfaces. The effect of multi-level set initialization is also shown empirically in Sec 4.5.

We also initialize the opacity and SH field from the pre-trained Plenoxels to facilitate optimization. After taking the raw density values $\sigma$ in the voxel grid, they are rescaled with a constant $s_\sigma$ to be used as raw surface opacity $\sigma_\alpha$, then mapped to opacity through a combined exponential-ReLU activation:

$$\sigma_\alpha = s_\sigma\, \sigma, \alpha = 1 - \exp\left(-\operatorname{ReLU}(\sigma_\alpha)\right) \tag{6}$$

Note that this activation resembles the mapping from discretized sample density to opacity in volume rendering where the step size term is replaced by the rescaling factor $s_\sigma$ [32]. Unlike sigmoid which has a vanishing gradient towards 0, this activation can more easily encourage sparsity in the surface opacity to remove redundant surfaces. The rescaling $s_\sigma$ is to ensure that initialized raw opacities do not map to $\alpha$ with too high values after removing the step size term, causing the gradients to be saturated. Note that, during training, we update $\sigma_\alpha$ as the training parameters rather than $\alpha$. The SH coefficients are also initialized from the pre-trained Plenoxels and further optimized.

In comparison to our approach, SDF methods such as NeuS [54] cannot easily take the advantages of initializing from Plenoxels or other NeRF-based methods due to two key aspects: 1) initializing the weights of the network to become an SDF that matches the coarse shape learned by NeRF is difficult, as it requires additional optimization to fit the shape while satisfying the Eikonal constraint; and 2) even for explicit grid-based SDF methods [19, 53], algorithms such as fast sweeping [9, 65] are required to explicitly assign the grid values as distance to closest surface [1, 48, 49]. This process can be done efficiently, but it no longer preserves the information in the initialized density field, making the removal of redundant surfaces and recovery of missing surfaces more difficult.

**Truncated Alpha Compositing** A critical artifact in NeRF methods, including Plenoxels, is that they tend to learn low-density surfaces and inner volumes that are only visible from certain angles to represent high-frequency view-dependent appearance [52]. This leads to very noisy inner surfaces when initialized from Plenoxels; see Figure 3. To regularize those artifacts, we first remove ray intersections on backward-facing surfaces, and define the remaining set $\mathcal{T}$ of intersection points to be considered for rendering and regularization:

$$t_i \in \mathcal{T} \text{ if } \underbrace{\mathbf{n}(t_i) \cdot \mathbf{d} < 0}_{\text{back-face culling}} \tag{7}$$

where $t_i$ is an original intersection, $\mathbf{n}(t_i) = \frac{-\nabla\delta(t_i)}{||\nabla\delta(t_i)||}$ is the surface normal, and $\mathbf{n}(t_i) \cdot \mathbf{d}$ checks whether the surface is facing backwards. This is similar to back-face culling in rendering. We then apply additional constraints by incorporating a truncated version of alpha compositing, where the later intersections along the ray are down-weighted with a truncated Hann window [36] to give less contribution to the rendering. Specifically, we obtain the re-weighted sample opacity as:

$$\alpha^*(t_i) = \gamma(i-1)\, \alpha(t_i) \tag{8}$$

$$\gamma(x) = \left(1 - \cos(\pi \operatorname{clamp}(a - x, 0, 1))\right)/2 \tag{9}$$

where $i$ is the index of the intersection starting from 1. Note that this is the index with the back-face intersections excluded. During training, we linearly reduce $a$ so that $\gamma(i-1)$ is only greater than zero for a smaller range of $i$. I.e., only the first $\operatorname{ceil}(a)$ intersections are kept and the rest are truncated. We now re-define our alpha compositing using this truncated version of opacity values: $\hat{C}^*(\mathbf{r}) = \sum_{t_i \in \mathcal{T}} T^*_\alpha(t_i)\alpha^*(t_i)\mathbf{c}(t_i)$, where $T^*_\alpha(t_i) = \prod_{j=1}^{i-1}(1 - \alpha^*(t_j))$.

**Regularization** We additionally apply regularizations to enforce smooth surfaces and mitigate the artifacts inherited from initialization. As previously mentioned, we initialize with multiple level sets to preserve geometry priors. However, this can lead to redundant surfaces and potentially deteriorate the reconstruction quality. We hence apply a surface convergence loss to each ray-surface intersection with non-trivial truncated opacity to converge different level surfaces together:

$$\mathcal{L}_c(\mathbf{r}) = \sum_{t_i \in \mathcal{T}} |\tilde{t} - t_i| \, \mathbb{I}[\alpha^*(t_i) > 10^{-8}] \tag{10}$$

where $\tilde{t}$ is the depth of the sample with the highest rendering weight $w(t_i) = T^*_\alpha(t_i)\alpha^*(t_i)$ on the ray, and $\mathbb{I}$ is the indicator function. This loss remedies the out-growing surfaces initialized from multi-level sets by encouraging them to move towards the actual surface location. The surface is also smoothed via a combination of an L1 and squared L2 normal smoothness loss and a total variation (TV) loss applied on the surface field:

$$\mathcal{L}_{\mathbf{n_1}} = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{x} \in \mathcal{V}} |\nabla \mathbf{n}(\mathbf{x})|, \, \mathcal{L}_{\mathbf{n_2}} = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{x} \in \mathcal{V}} ||\nabla \mathbf{n}(\mathbf{x})||^2 \tag{11}$$

$$\mathcal{L}_\delta = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{x} \in \mathcal{V}} ||\nabla \hat{\delta}(\mathbf{x})|| \tag{12}$$

where $\mathcal{V}$ contains the spatial coordinates of all voxels, $\nabla \hat{\delta}(\mathbf{x})$ is the surface gradient at voxel grids obtained using finite differences on the adjacent grid values. Note that normal regularization is applied regardless of whether the voxel contains a valid surface or not, as it can help to smooth the overall surface field instead of just the actual surface. While the normal regularization $\mathcal{L}_{\mathbf{n_1}}, \mathcal{L}_{\mathbf{n_2}}$ encourages smooth surfaces in a local scope, it alone struggles to remove redundant inner surfaces. $\mathcal{L}_\delta$ is more effective for regularizing redundant surfaces with large errors; see Supplementary. Note that the use of $\mathcal{L}_\delta$ is only possible because our implicit surface field is not an SDF and does not need to satisfy the Eikonal constraint.

Finally, we regularize the opacity field with a weight-based entropy loss adapted from [23] on each ray and an L1 sparsity loss:

$$\mathcal{L}_\mathcal{H}(\mathbf{r}) = -\sum_{t_i \in \mathcal{T}} \bar{w}(t_i) \log(\bar{w}(t_i)) \tag{13}$$

$$\text{where } \bar{w}(t_i) = \frac{T^*_\alpha(t_i)\alpha^*(t_i)}{\sum_{t_j \in \mathcal{T}} T^*_\alpha(t_j)\alpha^*(t_j)} \tag{14}$$

$$\mathcal{L}_\alpha = \frac{1}{|\mathcal{V}'|} \sum_{\mathbf{x} \in \mathcal{V}'} |\,\mathrm{ReLU}(\sigma_\alpha(\mathbf{x}))| \tag{15}$$

where $\mathcal{V}'$ is 10% of all existing voxels sampled uniformly at each iteration. They together encourage surfaces to have more concentrated and minimal opacity. Note that $\mathcal{L}_\mathcal{H}(\mathbf{r})$ does not always give beneficial regularization for scenes with semi-transparent materials, but we empirically found that having this term with a small weight can help remove the noise in the surface opacity.

|  | Thin | Translucent | Avg |
|---|---|---|---|
| Plen | 0.526 | 0.761 | 0.644 |
| Mip360 | 1.445 | 3.063 | 2.254 |
| NeuS | 1.048 | 2.344 | 1.696 |
| HFS | 0.925 | 3.698 | 2.312 |
| neuralangelo | 0.424 | 1.125 | 0.774 |
| NeRRF | 2.349 | 2.086 | 2.218 |
| Ours | 0.284 | 0.624 | 0.454 |

Table 1. **Chamfer distance ↓ $\times 10^{-2}$ on synthetic datasets.** We highlight the best methods. We justify the choice of density level set values for Plenoxels and MipNeRF360 in the supplementary.

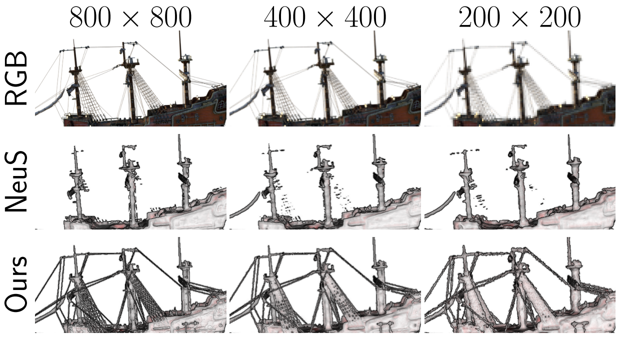

Figure 4. **Surface reconstruction with lower resolutions.** Our method can reconstruct thin surfaces under a low resolution, where thin structures such as ropes heavily blend with the background.

Given a batch $B$ of rays, the optimization target is:

$$\mathcal{L} = \frac{1}{|B|} \sum_{\mathbf{r} \in B} \Big( ||C(\mathbf{r}) - \hat{C}^*(\mathbf{r})||^2 + \lambda_c \mathcal{L}_c(\mathbf{r}) + \lambda_{\mathbf{n_1}} \mathcal{L}_{\mathbf{n_1}} + \lambda_{\mathbf{n_2}} \mathcal{L}_{\mathbf{n_2}} + \lambda_\delta \mathcal{L}_\delta + \lambda_\mathcal{H} \mathcal{L}_\mathcal{H}(\mathbf{r}) + \lambda_\alpha \mathcal{L}_\alpha \Big) \tag{16}$$

where $\lambda_c, \lambda_{\mathbf{n_1}}, \lambda_{\mathbf{n_2}}, \lambda_\delta, \lambda_\mathcal{H}, \lambda_\alpha$ are hyperparameters. The optimization targets include surface scalar $\delta$, grid SH coefficients and raw opacity $\sigma_\alpha$.

## 4. Evaluation

We quantitatively and qualitatively evaluate our method on an extended version of the NeRF synthetic dataset [32], with 8 additional objects with delicate and thin structures, and 8 objects with translucent materials. We also show a qualitative comparison on challenging real-world scenes containing translucent surfaces. We compare with re-

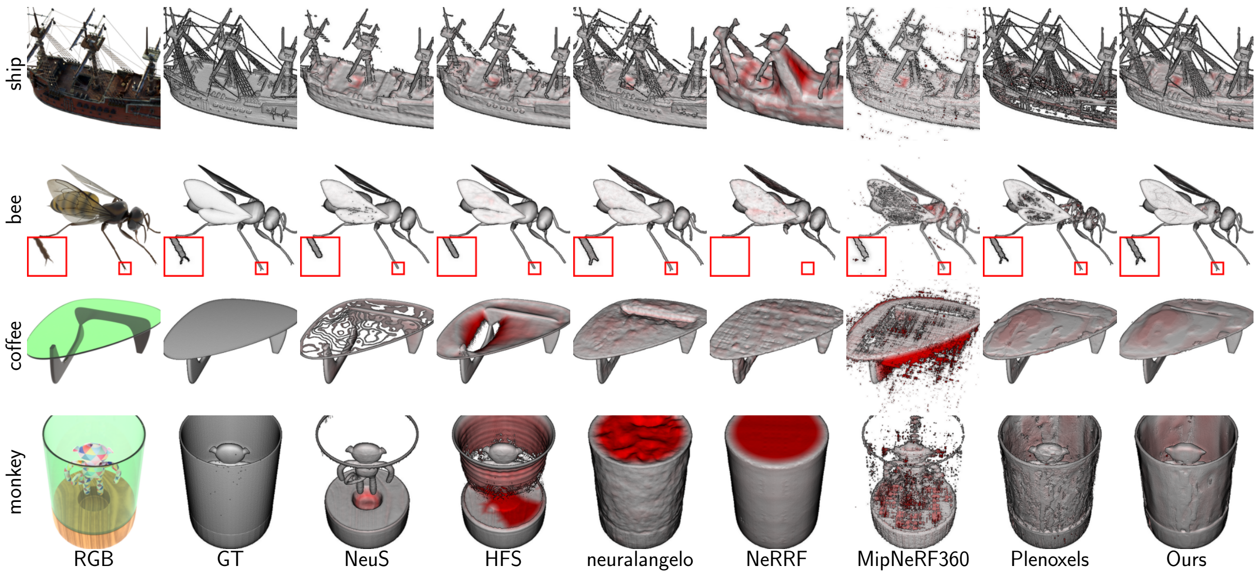

Figure 5. **Qualitative evaluation on synthetic datasets.** The red color indicates the L1 error in the reconstruction. Our method can reconstruct the translucent and thin surfaces missed in NeuS and HFS, while recovering more accurate and noise-free surfaces compared to neuralangelo and NeRF-based methods. Additional scenes can be found in the supplementary.

cent SDF optimization methods including NeuS [54] and HFS [56], and NeRF-based methods including Plenoxels [42] and MipNeRF360 [2].

### 4.1. Datasets

**Synthetic Thin & Translucent**: We render 8 scenes with thin structures and 8 with translucent surfaces with Blender [5]. We sampled 100 different training views from a full sphere. For the Thin dataset, we included the "ficus" and "ship" scenes from the original NeRF Synthetic dataset [32]. For each scene, we also rendered the depth and converted them to dense point clouds for quantitative geometry evaluation. This removes any invisible inner structures in the 3D assets. Except for the translucent scene "monkey" and "vase" where part of the object is completely surrounded by translucent surfaces, we hence directly extracted the scene meshes as the ground truth geometry. We will release all the datasets with reference geometry upon publication.

**Real-World**: We additionally captured a real-world scene with thin structures and two with translucent surfaces to qualitatively evaluate our performance. The real-world captures are processed with Colmap [43, 45] to obtain camera parameters. Note that since our work aims to resolve the geometry-material ambiguity in image-based neural reconstruction instead of handling complicated specular reflection or light transport, we hence focus on real-world thin translucent surfaces with less obvious view-dependent effects, such as cups.

### 4.2. Baselines

We compared our method with the state-of-the-art SDF-based reconstruction methods, as well as NeRF-based methods with level set geometry. We compare with NeuS [54] HFS [56], GeoNueS [11] and neuralangelo [28]. Since GeoNueS [11] requires SfM points and visibility masks as input, we only compare with it qualitatively in real-world scenes. We also compare with Plenoxels [42] and Mip-NeRF 360 [2], a follow-up of NeRF with conical frustum sampling and weight regularization. We also compare with NeRRF [4], which reconstructs fully transparent objects with additional mask supervision.

### 4.3. Evaluation on Synthetic dataset

We quantitatively evaluate our method on the synthetic datasets and report the Chamfer-L1 distance as evaluated in [17] in Table 1. HFS failed and learned empty surfaces on two Translucent scenes, while neuralangelo failed on one scene in Thin Blender. In comparison, our method signifcantly outperforms the baselines, and the difference is particularly noticeable when compared to SDF-based methods on the translucent dataset, as the baselines cannot properly represent translucent surfaces. NeRF-based methods can potentially recover the translucent surfaces with a low density level set, but their ambiguity in representation leads to significant noise in the reconstruction.

We show qualitative comparison in Figure 5. While NeuS, HFS and Neuralangelo are capable of reconstructing highly smooth surfaces, they cannot properly capture

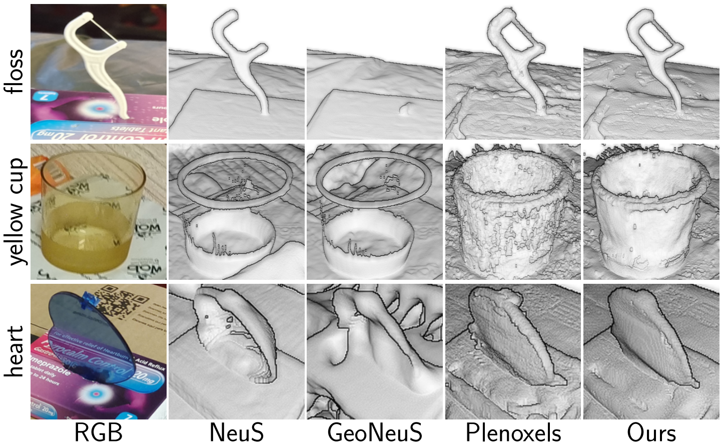

Figure 6. **Qualitative evaluation on translucent real-world scenes.** The images are not aligned perfectly due to the use of NDC during optimization.

| name | ship | ficus | table | monkey |
|---|---|---|---|---|
| No $\mathcal{L}_\delta$ | .317 | .279 | .373 | .777 |
| No $\mathcal{L}_\delta$, $\mathcal{L}_{\mathbf{n}_{1,2}}$ | .373 | .333 | .404 | .970 |
| No trunc | .522 | .837 | .431 | 1.03 |
| No $\mathcal{L}_c$ | .287 | .266 | .412 | .857 |
| $\boldsymbol{\tau}_\sigma = \{10\}$ | .624 | .583 | .470 | .946 |
| Ours | .277 | .240 | .373 | .776 |

Table 2. **Quantitative results of ablation study.** We report the Chamfer distance $\times 10^{-2}$ on two scenes from each synthetic dataset. More results can be found in the supplementary.

translucent or thin surfaces. Surfaces from Plenoxels and MipNeRF360 contain holes and floaters and are unsmooth. NeRRF is capable of reconstructing smooth translucent surfaces, but fails to model intricate structures. Our approach is capable of reconstructing surfaces with minimal artifacts. In Figure 4, we show surfaces reconstructed from 800, 400, and 200 resolution images on a scene with thin structures. Our method can reconstruct the thin surfaces despite the heavy blending effects in the low-resolution images.

### 4.4. Evaluation on Real World Dataset

We show the qualitative comparison on real-world scenes with thin and translucent surfaces in Figure 6. We show surfaces extracted Plenoxels using $\sigma = 10$ level sets, as it has the best qualitative results compared to $30, 50$ levels. Neus and GeoNueS fail to reconstruct the majority of the thin or translucent surface, while surfaces from Plenoxels are noisy and unsmooth. Our method mitigates the artifacts inherited from NeRF initialization and reconstructs surfaces with much higher quality. We would like to highlight that, due to the highly ill-posed nature of the problem and view-dependent appearance changes caused by global brightness shifts in an uncontrolled capture environment, it is nearly impossible to reconstruct the translucent surfaces perfectly. Our method achieves significant improvement upon baselines, validating the feasibility of our approach. Additional comparison with neuralangelo [28] can be found in the supplementary.

### 4.5. Ablation

As shown in Table 2, the truncated alpha compositing deals with the inner volume artifacts inherited from initialization, while the surface regularization $\mathcal{L}_{\mathbf{n}_1}, \mathcal{L}_{\mathbf{n}_2}, \mathcal{L}_\delta$ and convergence loss $\mathcal{L}_c$ encourage smooth and accurate surfaces. Using a single level set $\boldsymbol{\tau}_\sigma = \{10\}$ to initialize the surface fails to capture all information in the pre-trained NeRF and causes artifacts in optimization. Ours with all techniques enabled achieves the best performance. More results can be found in the supplementary.

## 5. Conclusion

We present αSurf, a grid-based surface representation with decoupled geometry, opacity, and appearance. We develop closed-form intersection finding and differentiable alpha compositing to optimize the surface via photometric loss. Our representation utilizes initialization from efficient Plenoxels [42], and incorporates truncated rendering and additional surface regularizations to reconstruct high-quality surfaces for translucent objects and thin structures with heavy blending effects.

**Limitation**: Compared to MLP-based SDF methods such as NeuS [54] and neuralangelo [28], our reconstructed surface tends to be less smooth due to the lack of spatial smoothness encoded in the MLP; see Figure 5. It presents a trade-off: stronger surface regularization can certainly give smoother surfaces, but can also destroy delicate thin structures in the reconstruction. In addition, we focus only on decoupling geometry and material ambiguity in existing volumetric representation, and do not specifically handle strong reflections as in Ref-NeRF [52].

## Acknowledgements

This work was supported by a UKRI Future Leaders Fellowship [grant number G104084]

## References

[1] David Adalsteinsson and James A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995. 5

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 3, 7, 14, 15

[3] Bowen Cai, Jinchi Huang, Rongfei Jia, Chengfei Lv, and Huan Fu. Neuda: Neural deformable anchor for high-fidelity implicit surface reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2023. 3

[4] Xiaoxue Chen, Junchen Liu, Hao Zhao, Guyue Zhou, and Ya-Qin Zhang. Nerrf: 3d reconstruction and view synthesis for transparent and specular objects with neural refractive-reflective fields, 2023. 3, 7

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 7

[6] Cloud Compare. *CloudCompare 3D point cloud and mesh processing software Open Source Project*, 2020. 14

[7] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, page 303–312, New York, NY, USA, 1996. Association for Computing Machinery. 3

[8] Yu Deng, Jiaolong Yang, Jianfeng Xiang, and Xin Tong. Gram: Generative radiance manifolds for 3d-aware image generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 3

[9] Miles Detrixhe, Frédéric Gibou, and Chohong Min. A parallel fast sweeping method for the eikonal equation. *Journal of Computational Physics*, 237:46–55, 2013. 5

[10] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983. 3

[11] Qiancheng Fu, Qingshan Xu, Yew-Soon Ong, and Wenbing Tao. Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *NeurIPS*, 2022. 3, 7

[12] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023. 2

[13] Herman Hansson Söderlund, Alex Evans, and Tomas Akenine-Möller. Ray tracing of signed distance function grids. *Journal of Computer Graphics Techniques (JCGT)*, 11(3):94–113, 2022. 11

[14] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 3

[15] Kejing He, Congying Sui, Tianyu Huang, Rong Dai, Congyi Lyu, and Yun-Hui Liu. 3d surface reconstruction of transparent objects using laser scanning with ltftf method. *Optics and Lasers in Engineering*, 148:106774, 2022. 3

[16] Geoffrey Hinton. Rmsprop. 12

[17] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 7, 14, 15, 20

[18] Hanwen Jiang, Zhenyu Jiang, Yue Zhao, and Qixing Huang. Leap: Liberate sparse-view 3d modeling from camera poses. *ArXiv*, 2310.01410, 2023. 3

[19] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 5, 11

[20] Jzhangbs. Jzhangbs/dtueval-python: A fast python implementation of dtu mvs 2014 evaluation. 14

[21] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Symposium on Geometry Processing*. The Eurographics Association, 2006. 1, 3

[22] 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

[23] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *CVPR*, 2022. 6

[24] D. Lee and B. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Parallel Programming*, 9:219–242, 1980. 1

[25] Hai Li, Xingrui Yang, Hongjia Zhai, Yuqian Liu, Hujun Bao, and Guofeng Zhang. Vox-surf: Voxel-based implicit surface representation. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–12, 2022. 3

[26] Zhengqin Li, Yu-Ying Yeh, and Manmohan Chandraker. Through the looking glass: Neural 3d reconstruction of transparent shapes. pages 1259–1268, 2020. 3

[27] Zongcheng Li, Xiaoxiao Long, Yusen Wang, Tuo Cao, Wenping Wang, Fei Luo, and Chunxia Xiao. Neto: Neural reconstruction of transparent objects with self-occlusion aware refraction-tracing. *arXiv:2303.11219*, 2023. 3

[28] 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 *CVPR*, 2023. 3, 7, 8, 14, 16, 17

[29] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 3

[30] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2019–2028, 2020. 1

[31] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. *ECCV*, 2022. 3

[32] 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. 1, 3, 5, 6, 7

[33] 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

[34] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[35] 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. 14

[36] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 5, 14

[37] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 3

[38] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, 2007. 4, 11

[39] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *International Conference on Computer Vision (ICCV)*, 2021. 3

[40] Yufan Ren, Fangjinhua Wang, Tong Zhang, Marc Pollefeys, and Sabine Süsstrunk. Volrecon: Volume rendering of signed ray distance functions for generalizable multi-view reconstruction, 2023. 3

[41] Radu Alexandru Rosu and Sven Behnke. Hashsdf: Accurate implicit surfaces with fast local features on permutohedral lattices, 2022. 3

[42] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2, 3, 4, 7, 8, 12, 14, 15, 16

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

[44] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[45] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 7

[46] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 1

[47] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[48] James A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93 4: 1591–5, 1996. 5

[49] J. A. Sethian. Fast marching methods. *SIAM Review*, 41(2): 199–235, 1999. 5

[50] C. Bane Sullivan and Alexander Kaszynski. PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK). *Journal of Open Source Software*, 4(37):1450, 2019. 14

[51] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 3

[52] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 3, 5, 8

[53] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Differentiable signed distance function rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4):125:1–125:18, 2022. 1, 5

[54] 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, 3, 5, 7, 8, 14, 16

[55] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction, 2022. 3

[56] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. *arXiv preprint arXiv:2206.07850*, 2022. 3, 7, 14, 18

[57] Huang Xin, Zhang Qi, Feng Ying, Li Hongdong, Wang Xuan, and Wang Qing. Hdr-nerf: High dynamic range neural radiance fields. *arXiv preprint arXiv:2111.14451*, 2021. 3

[58] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. Sinnerf: Training neural radiance fields on complex scenes from a single image. 2022. 3

[59] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 3, 16

[60] 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. 3

[61] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 3

[62] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 3

[63] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 3

[64] Kai Zhang, Fujun Luan, Zhengqi Li, and Noah Snavely. Iron: Inverse rendering by optimizing neural sdfs and materials from photometric images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 1

[65] Hongkai Zhao. A fast sweeping method for eikonal equations. *Math. Comput.*, 74:603–627, 2004. 5

# αSurf: Implicit Surface Reconstruction for Semi-Transparent and Thin Objects with Decoupled Geometry and Opacity

Supplementary Material

## A. Overview

In the supplementary material, we include additional experiment details and evaluation results. We also encourage the reader to watch the video results contained in the supplementary files.

## B. Implementation Details

### B.1. Closed-Form Intersection

As briefly mentioned in Section 3.2 of our paper, we determine the ray-surface intersections through the analytical solution of cubic polynomials. Note that a similar technique has been identified in previous works [13, 19], but they applied it on SDF only. We identify that the same approach can be applied to a more generalized implicit surface field without the Eikonal constraint. We now present the detailed derivation of it.

Given a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with origin $\mathbf{o}$ and direction $\mathbf{d}$, our aim is to find the intersections between the ray and a level set surface with value $\tau_i$ within a voxel $v_{\mathbf{x}}$, which $\mathbf{r}(t)$ is guaranteed to hit. The value of the implicit surface field within $v_{\mathbf{x}}$ can be determined through the trilinear interpolation of eight surface scalars stored on the vertices of $v_{\mathbf{x}}$:

$$\delta(\mathbf{x}) = \mathrm{trilerp}(\mathbf{x}, \{\hat{\delta}_i\}_{i=1}^{8}) \tag{17}$$

$$\begin{aligned} &=(1-z)((1-y)((1-x)\hat{\delta}_1 + x\hat{\delta}_5) \\ &\quad + y((1-x)\hat{\delta}_3 + x\hat{\delta}_7)) \\ &\quad + z((1-y)((1-x)\hat{\delta}_2 + x\hat{\delta}_6) \\ &\quad + y((1-x)\hat{\delta}_4 + x\hat{\delta}_8)) \end{aligned} \tag{18}$$

where $[x, y, z] = \mathbf{x} - \mathbf{l}$ are the relative coordinates within the voxel, and $\mathbf{l} = \mathrm{floor}(\mathbf{x})$. Note that $x, y, z \in [0, 1]$. We first determine the near and far intersections $t_n, t_f$ between the ray and voxel $v_{\mathbf{x}}$ through the ray-box AABB algorithm, and then redefine a new camera origin $\mathbf{o}' = \mathbf{o} + t_n\mathbf{d} - \mathbf{l}$. We hence directly have $[x, y, z] = \mathbf{o}' + t'\mathbf{d} \in [0, 1]$, where $t' = t - t_n$ without the need for calculating relative coordinates again. By denoting $\mathbf{o}' = [o'_x, o'_y, o'_z]$, $\mathbf{d} = [d_x, d_y, d_z]$, we substitute the above as well as $\delta(\mathbf{x}) = \tau_i$ into Equation 18:

$$\begin{aligned} \tau_i =&(1-(o'_z+t'd_z))((1-(o'_y+t'd_y)) \\ &((1-(o'_x+t'd_x))\hat{\delta}_1 + (o'_x+t'd_x)\hat{\delta}_5) \\ &+ (o'_y+t'd_y)((1-(o'_x+t'd_x))\hat{\delta}_3 + (o'_x+t'd_x)\hat{\delta}_7)) \\ &+ (o'_z+t'd_z)((1-(o'_y+t'd_y)) \\ &((1-(o'_x+t'd_x))\hat{\delta}_2 + (o'_x+t'd_x)\hat{\delta}_6) \\ &+ (o'_y+t'd_y)((1-(o'_x+t'd_x))\hat{\delta}_4 + (o'_x+t'd_x)\hat{\delta}_8)) \,. \end{aligned} \tag{19}$$

By re-arranging the equation, we obtain:

$$\tau_i = f_3 t'^3 + f_2 t'^2 + f_1 t' + f_0 \tag{20}$$

where

$$\begin{aligned} f_0 =&(m_{00}(1-o'_y) + m_{01}(o'_y))(1-o'_x) \\ &+ (m_{10}(1-o'_y) + m_{11}(o'_y))(o'_x) \\ f_1 =&(m_{10}(1-o'_y) + m_{11}(o'_y))d_x + k_1(o'_x) \\ &- (m_{00}(1-o'_y) + m_{01}(o'_y))d_x + k_0(1-o'_x) \\ f_2 =&k_1 d_x + h_1(o'_x) - k_0 d_x + h_0(1-o'_x) \\ f_3 =&h_1 d_x - h_0 d_x \end{aligned} \tag{21}$$

and

$$\begin{aligned} m_{00} =&\hat{\delta}_1(1-o'_z) + \hat{\delta}_2(o'_z) \\ m_{01} =&\hat{\delta}_3(1-o'_z) + \hat{\delta}_4(o'_z) \\ m_{10} =&\hat{\delta}_5(1-o'_z) + \hat{\delta}_6(o'_z) \\ m_{11} =&\hat{\delta}_7(1-o'_z) + \hat{\delta}_8(o'_z) \\ k_0 =&(m_{01}d_y + d_z(\hat{\delta}_4 - \hat{\delta}_3)(o'_y)) \\ &- (m_{00}d_y - d_z(\hat{\delta}_2 - \hat{\delta}_1)(1-o'_y)) \\ k_1 =&(m_{11}d_y + d_z(\hat{\delta}_8 - \hat{\delta}_7)(o'_y)) \\ &- (m_{10}d_y - d_z(\hat{\delta}_6 - \hat{\delta}_5)(1-o'_y)) \\ h_0 =&d_y d_z(\hat{\delta}_4 - \hat{\delta}_3) - d_y d_z(\hat{\delta}_2 - \hat{\delta}_1) \\ h_1 =&d_y d_z(\hat{\delta}_8 - \hat{\delta}_7) - d_y d_z(\hat{\delta}_6 - \hat{\delta}_5) \,. \end{aligned} \tag{22}$$

Therefore, we obtain a cubic polynomial with a single unknown $t'$. Note that here we only sketch the main idea. For the actual implementation, we refer to [13] which provides a more concise implementation that formulates the cubic polynomials with fewer operations through the use of fused-multiply-add.

We then incorporate Vieta's approach [38] to solve the real roots for $t'$ in an analytic way. Namely, we first re-write the cubic polynomial as follows:

$$\tau_i = t'^3 + at'^2 + bt' + c \tag{23}$$

$$a = \frac{f_2}{f_3}, b = \frac{f_1}{f_3}, c = \frac{f_0}{f_3}\,. \tag{24}$$

Then, compute:

$$Q = \frac{a^2 - 3b}{9} \tag{25}$$

$$R = \frac{2a^3 - 9ab + 27c}{54}\,. \tag{26}$$

If $R^2 < Q^3$, we have three real roots given by:

$$\theta = \arccos(\frac{R}{\sqrt{Q^3}}) \tag{27}$$

$$t'_1 = -2\sqrt{Q}\cos(\frac{\theta}{3}) - \frac{a}{3} \tag{28}$$

$$t'_2 = -2\sqrt{Q}\cos(\frac{\theta - 2\pi}{3}) - \frac{a}{3} \tag{29}$$

$$t'_3 = -2\sqrt{Q}\cos(\frac{\theta + 2\pi}{3}) - \frac{a}{3} \tag{30}$$

where $t'_1 \leq t'_2 \leq t'_3$. This can be trivially seen from $0 \leq \theta \leq \pi$, $\sqrt{Q} \geq 0$ and $\cos(\frac{\theta}{3}) \geq \cos(\frac{\theta - 2\pi}{3}) \geq \cos(\frac{\theta + 2\pi}{3})$.

If $R^2 \geq Q^3$, we only have a single real root. First compute:

$$A = -\operatorname{sign}(R)\left(|R| + \sqrt{R^2 - Q^3}\right)^{1/3} \tag{31}$$

$$B = \begin{cases} Q/A, & \text{if } A \neq 0 \\ 0, & \text{otherwise} \end{cases}\,. \tag{32}$$

Then, the only real root can be obtained as:

$$t'_1 = (A + b) - \frac{a}{3}\,. \tag{33}$$

The intersection coordinate can therefore be determined as $\mathbf{r}(t_n + t')$. We then check the intersections against the bounding box of each voxel $v_{\mathbf{x}}$ to remove any samples outside of the voxels. Besides, the cubic polynomial might return multiple valid real roots within the voxel if $R^2 < Q^3$. If the roots are unique, that means the ray intersects with the same surface multiple times within the voxel, all the intersections are taken for rendering. However, if the roots are identical, we remove the redundant ones to prevent using the same intersection multiple times.

As both the formulation of cubic polynomials and Vieta's approach are fully differentiable, we hence directly have gradients defined on our surface representation $\hat{\delta}$ from the photometric loss.

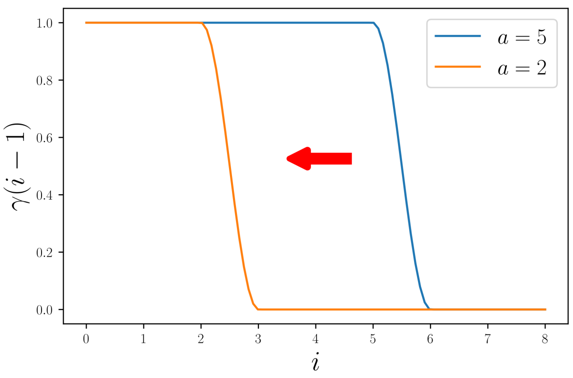

Figure 7. **The truncated alpha compositing reweight function $\gamma(i-1)$.** The x-axis is the index of the intersection starting from 1 (excluding intersections on back-facing surfaces). By reducing $a$, we effectively slide the curve to the left.

## B.2. Hyperparameters

Our code is based on Plenoxels [42]. We similarly use a sparse voxel grid of size $512^3$ where each vertex stores the surface scalar $\delta$, raw opacity $\sigma_\alpha$ and 9 SH coefficients for each color channel. We directly initialize all the grid values from Plenoxels pre-trained with original hyperparameters and prune voxels with densities $\sigma$ lower than 5. We use $s_\sigma = 0.05$ to downscale the density values during initialization. We train for $50k$ iterations with a batch size of $5k$ rays, which takes around 17 minutes for synthetic scenes and 22 minutes for real-world scenes on an NVIDIA A100-SXM-80GB GPU (excluding Plenoxels training). We used grid search to determine the optimal hyperparameters. We use the same delayed exponential learning rate schedule, where the learning rate is delayed with a scale of 0.01 during first $25k$ iterations. As previously mentioned, the interval of level values is selected by first determining a valid range of Plenoxels density field. We then select a suitable number of level values, i.e., the carnality $n$ of our multi level sets, by trying $1, 3, 5, 10$ evenly-spaced level values on the "ship" scene from NeRF Synthetic dataset. We found $n = 5$ to give the best performance. At the end of the training, we also remove invisible surfaces with opacity $\alpha$ less than 0.1.

**Synthetic** For experiments on synthetic datasets, we initialize 5 level sets at $\boldsymbol{\tau}_\sigma = \{10, 30, 50, 70, 90\}$, and linearly decay the truncated alpha compositing parameter $a$ from 5 to 2 in first $10k$ iterations. For surface scalars $\hat{\delta}$, we use $10^{-5}$ as both starting and end learning rate. For raw opacity values $\sigma_\alpha$, we start with $10^{-2}$ and end with $10^{-3}$. For SH we keep the learning rate at $10^{-3}$ without exponential decay or initial delay. We use the RMSProp [16] optimizer for training. For the regularization weights, we set $\lambda_c = 10^{-6}$

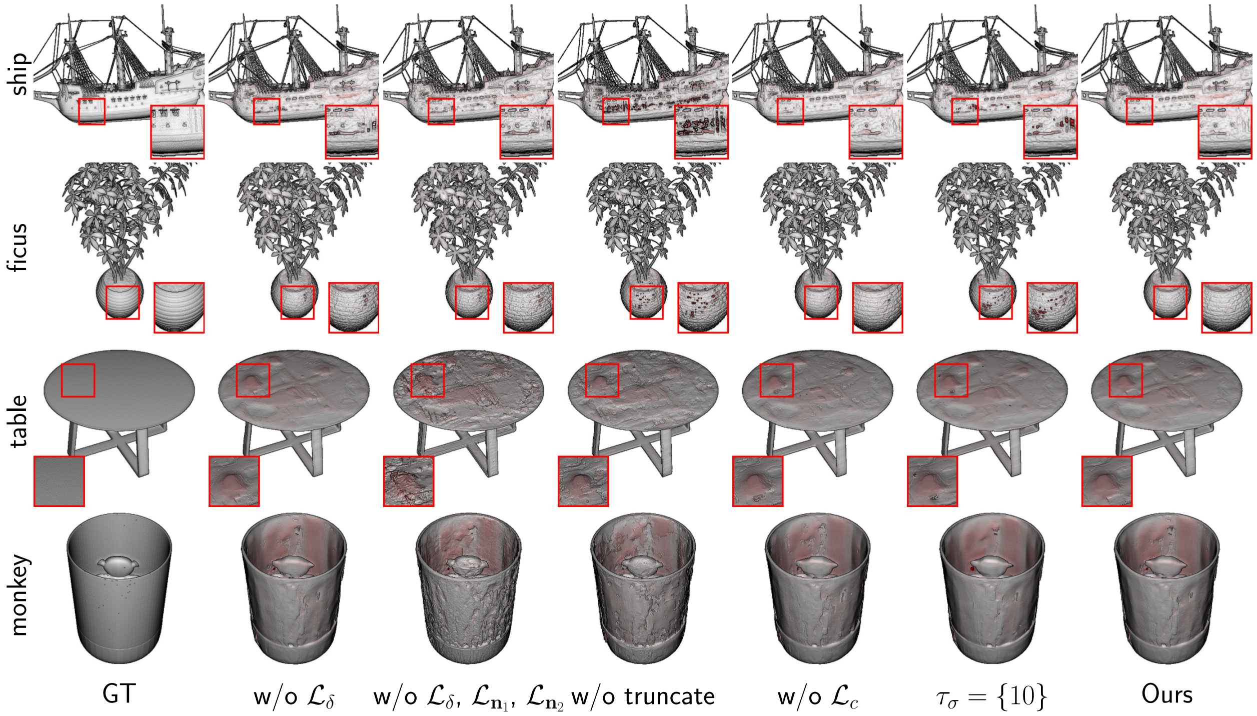

Figure 8. **Ablation Study.** We show the qualitative results of different ablations. Our full approach achieves the best quality overall.

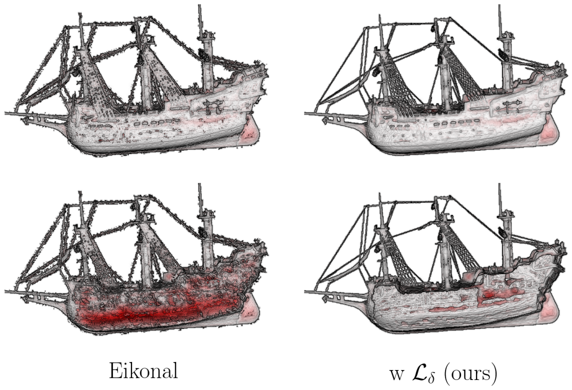

Figure 9. **Comparison with the Eikonal constraint regularization.** We visualize the out view (first column) and the inside view (second column) by cropping the surfaces along the y-axis. It can be clearly seen that the Eikonal constraint does not regularize the surface to be clean and smooth, but rather creates additional noises in optimization.

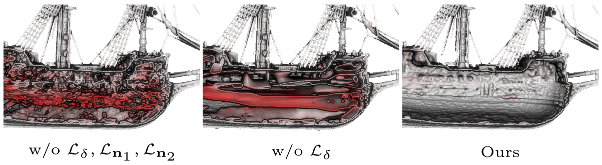
w/o $\mathcal{L}_\delta, \mathcal{L}_{\mathbf{n}_1}, \mathcal{L}_{\mathbf{n}_2}$
w/o $\mathcal{L}_\delta$
Ours

Figure 10. Reconstruction without regularization tends to give an enormous amount of redundant inner surfaces. Encouraging consistent normals in the local neighborhood via $\mathcal{L}_{\mathbf{n}_1}, \mathcal{L}_{\mathbf{n}_2}$ enforces smoother surfaces, but redundant surfaces still remain. With both normal regularization and TV loss applied on the surface scalar field, we can obtain clean and smooth reconstruction.

for the first $10k$ iterations and $0$ for the rest of training. We use $\lambda_\delta = 10^{-3}$, $\lambda_{\mathcal{H}} = 10^{-4}$, $\lambda_\alpha = 10^{-9}$, $\lambda_{\mathbf{n}_1} = 10^{-6}$ and $\lambda_{\mathbf{n}_2} = 0$ for the Thin dataset. $\lambda_{\mathbf{n}_2}$ was disabled as it tends to destroy the thin structures with rapid normal variations. For the Translucent dataset, we use $\lambda_\delta = 10^{-5}$, $\lambda_\alpha = 10^{-11}$, $\lambda_{\mathcal{H}} = 10^{-4}$, $\lambda_{\mathbf{n}_2} = 10^{-4}$, and linearly decay $\lambda_{\mathbf{n}_1}$ from $10^{-2}$ to $10^{-4}$.

**Real-World** For experiments on real-world scenes, we initialize with less level sets $\boldsymbol{\tau}_\sigma = \{10, 30, 50\}$, as we found level values above 50 give almost empty surfaces due to higher density regularization in Plenoxels initialization. We use the hyperparameters used by the original authors to run LLFF experiments to train Plenoxels. For our method, we use level sets $\tau_\sigma = \{10, 30, 50\}$ as level value above 50 gives almost empty space. We change the surface scalar learning rate to start and end both at $10^{-4}$ with a delay ratio of $10^{-2}$ and delay steps of $25k$. The learning rates of opacity and SH are the same as in the synthetic experiments. For regularizations, we use the same $\lambda_c$ and $\lambda_{\mathcal{H}}$ as synthetic experiments, and set $\lambda_\delta = 5 \times 10^{-3}$, $\lambda_\alpha = 10^{-9}$, $\lambda_{\mathbf{n}_2} = 10^{-3}$ and linearly decay $\lambda_{\mathbf{n}_1}$ from $10^{-2}$ to $10^{-3}$.

For the implementation of surface TV loss $\mathcal{L}_\delta$, we calculate the gradient via forward finite difference in the same way as Plenoxels [42]:

$$\nabla_x \hat{\delta}(i,j,k) = \frac{|\hat{\delta}(i+1,j,k) - \hat{\delta}(i,j,k)| D_x}{256} \tag{34}$$

where $i, j, k$ are the vertex coordinate, $D_x$ is the grid resolution in $x$ dimension and is 512 for all experiments in our case. $\nabla_y \hat{\delta}(i,j,k)$ and $\nabla_z \hat{\delta}(i,j,k)$ are calculated accordingly. We simply ignore the edge vertices when computing the surface TV loss by using the Neumann boundary conditions.

The truncated alpha compositing reweight function can be seen as a truncated Hann window [36], as shown in Figure 7. By reducing $a$ during the training, we slide the curve to the left and hence gradually anneal the influence of later intersections.

# C. Additional Experiments

## C.1. Synthetic Dataset

**Experiment Details** For quantitative evaluation, we adapt the Python version of DTU [17] evaluation script [20], where we extracted dense point clouds from all level surfaces and downsampled both predicted and ground truth points with 0.001 density before computing the Chamfer distance. For evaluation of NeuS [54] and HFS [56], we first extracted the mesh using marching cubes with resolution $512^3$, then used the script to sample points on the mesh to compute the Chamfer distance. For evaluation of Plenoxels [42], MipNeRF360 [2] and our method, we directly sample points on the implicit surfaces by sending dense virtual rays within each grid of a $512^3$ voxel grid through our closed-form intersection finding. This makes the computation of sample opacity and trimming of the surface easier.

For training of NeuS [54], HFS [56] and MipNeRF360 [2], we used the provided hyperparameters. We used the hyperparameters for real-world thin structure reconstruction experiments for NeuS, as we found it gives better performance on the NeRF Synthetic dataset. For training on the Translucent Blender dataset, we set the background to white for all methods as the semi-transparent objects are rendered with a white background in Blender.

To select a level set value on the density field of Pleboxels [42] and MipNeRF 360 [2] for surface extraction, we use the same methods as in [35, 54], where we extracted and evaluated surfaces on levels $\tau_\sigma = \{10, 30, 50, 70, 90, 100\}$, which fully covers the surfaces we used to initialize from Plenoxels. We computed the average norm on Synthetic, Thin, and Translucent datasets and selected the level set value with the best Chamfer distance on each of the datasets. For Plenoxels, the level sets are $90, 30$ and for MipNeRF 360, the level sets are $50, 50$ for the two datasets respectively. We report the quantitative results for each level set in Tab 3 and show a few qualitative examples in Figure 11.

| | Thin | Translucent | average |
|---|---|---|---|
| Plen ($\sigma = 10$) | 0.759 | 0.813 | 0.786 |
| Plen ($\sigma = 30$) | 0.886 | 0.761 | 0.824 |
| Plen ($\sigma = 50$) | 0.687 | 0.812 | 0.750 |
| Plen ($\sigma = 70$) | 0.563 | 1.062 | 0.812 |
| Plen ($\sigma = 90$) | 0.526 | 1.597 | 1.062 |
| Plen ($\sigma = 100$) | 0.541 | 1.832 | 1.186 |
| Mip360 ($\sigma = 10$) | 1.882 | 3.76 | 2.821 |
| Mip360 ($\sigma = 30$) | 1.468 | 3.081 | 2.274 |
| Mip360 ($\sigma = 50$) | 1.445 | 3.063 | 2.254 |
| Mip360 ($\sigma = 70$) | 1.526 | 3.07 | 2.298 |
| Mip360 ($\sigma = 90$) | 1.635 | 3.116 | 2.376 |
| Mip360 ($\sigma = 100$) | 1.693 | 3.203 | 2.448 |

Table 3. **Chamfer distance ↓ $\times 10^{-2}$ on synthetic datasets.** We color the best level sets for Plenoxels [42] (Plen in table) and MipNeRF360 [2] (Mip360 in table) respectively.

**Additional Results** We show all the qualitative results in 14, 15, as well as the individual Chamfer distance for each scene in 5 and 6. The qualitative comparisons shown in both main paper and the supplementary are done by first evaluating the L1 error on each sampled point, then rendering the point cloud with Eye-Dome Lighting (EDL) using PyVista [50]. We also show additional novel view RGB renderings of our method in Figure 16. But please note that we do not claim state-of-the-art performance in novel view synthesis.

## C.2. Real-World Dataset

We show additional comparisons with neuralangelo [28] in Fig 12. Note that as neuralangelo uses a different camera normalization for COLMAP scenes instead of Normalized Device Coordinate (NDC), which we use for our method and all other baselines, the reconstruction of neuralangelo is therefore not exactly aligned. We use an interactive viewer with Eye Dome Lighting [6] and manually selected camera positions with close views for comparison. Regardless, it can be clearly seen that although neuralangelo excels at reconstructing smooth surfaces, it fails to faithfully reconstruct thin or translucent surfaces. Our method achieves a significant improvement over it in terms of thin and translucent surface reconstruction.

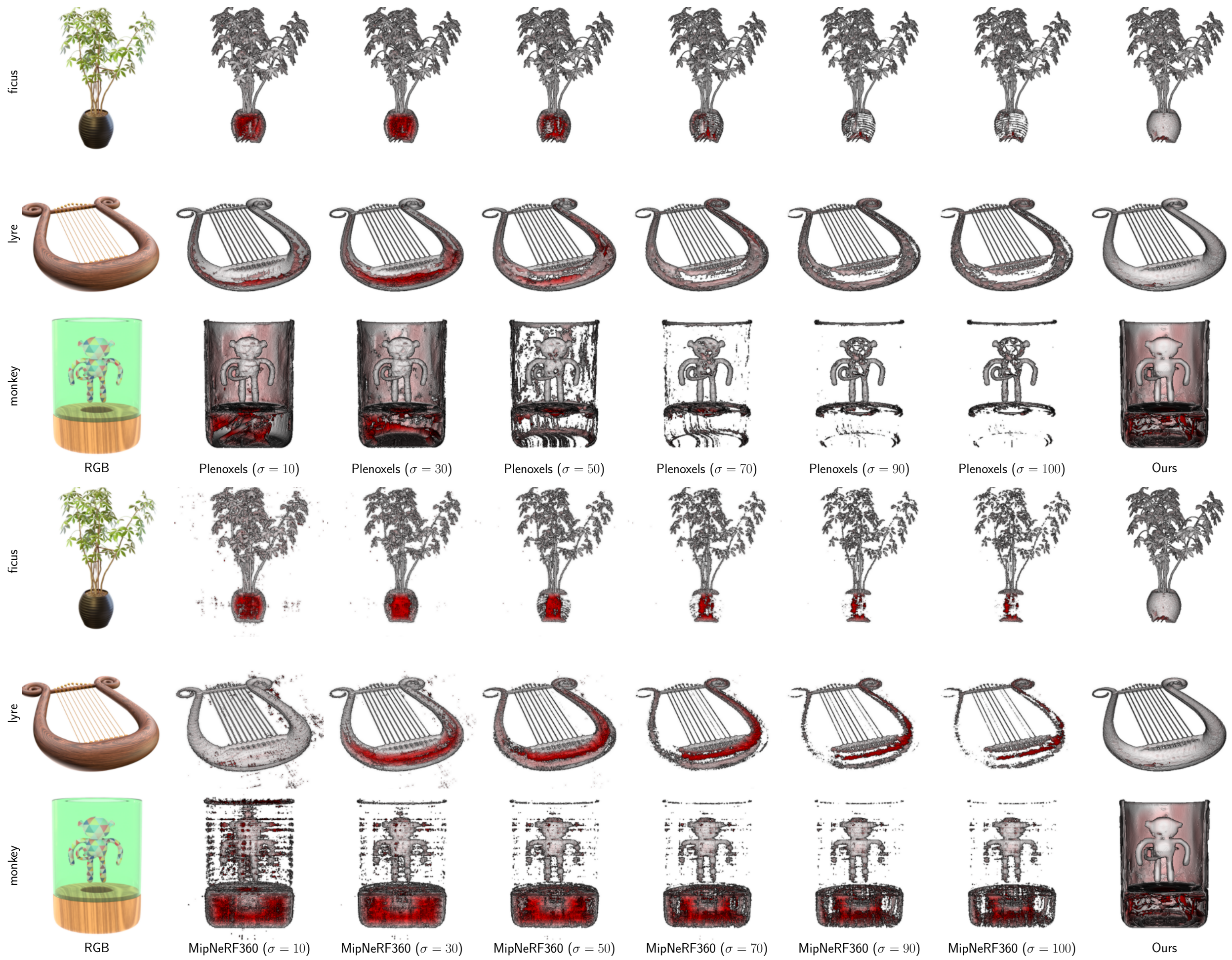
ficus
lyre
monkey
RGB
Plenoxels ($\sigma = 10$)
Plenoxels ($\sigma = 30$)
Plenoxels ($\sigma = 50$)
Plenoxels ($\sigma = 70$)
Plenoxels ($\sigma = 90$)
Plenoxels ($\sigma = 100$)
Ours
ficus
lyre
monkey
RGB
MipNeRF360 ($\sigma = 10$)
MipNeRF360 ($\sigma = 30$)
MipNeRF360 ($\sigma = 50$)
MipNeRF360 ($\sigma = 70$)
MipNeRF360 ($\sigma = 90$)
MipNeRF360 ($\sigma = 100$)
Ours

Figure 11. **Surfaces extracted using different level sets from Plenoxels [42] and MipNeRF360 [2].** We remove part of the exterior surface in each scene to visualize the interior reconstructions. Due to the ambiguity of density representation, a low density level set gives more complete surfaces but could contain a significant amount of noise, whereas a high density level set can miss a lot of surfaces.

### C.3. Ablation

We show additional qualitative ablation of our method in Figure 8. In addition, we show a comparison between the results after applying our TV surface regularization $\mathcal{L}_\delta$ and after applying the Eikonal constraint regularization used in most SDF optimization methods in Figure 9. Namely, in replace of TV surface regularization, we encourage the norm of the gradient of the surface field at every vertex to get close to 1 via mean squared error:

$$\mathcal{L}_{ek} = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{x} \in \mathcal{V}} (||\nabla \hat{\delta}(\mathbf{x})||_2 - 1)^2 \,. \tag{35}$$

From Figure 9, it can be clearly seen that the Eikonal constraint is not sufficient to regularize and remove the noisy inner surfaces inherited from initialization. Moreover, it turns out to even harm the optimization by introducing additional surface floaters while trying to constrain the surface field into an SDF. This also shows that converting the surfaces extracted from a density field into proper SDF is a non-trivial task.

In Figure 10, we show that normal regularization $\mathcal{L}_{\mathbf{n}_1}, \mathcal{L}_{\mathbf{n}_2}$ are insufficient for removing heavily biased surfaces initialized from Plenoxels, whereas $\mathcal{L}_\delta$ is more effective in this case.

### C.4. DTU Dataset

We additionally show reconstruction results on some DTU [17] scenes in Figure 17 and Table 4. We note that as DTU does not contain many thin structures or semi-transparent materials, but mostly smooth surfaces only, our method is therefore not expected to achieve state-of-the-art performance in this scenario. In fact, our method reconstructs

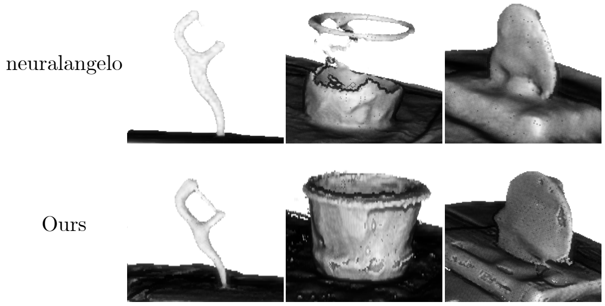

Figure 12. **Additional real-world comparisons with neuralangelo [28]** Note that neuralangelo uses a different coordinate system and camera processing pipeline for COLMAP scenes, therefore the reconstructions are not perfectly aligned, but it can still be clearly seen that our method achieves better reconstruction quality on thin structures and translucent surfaces.

| | 37 | 40 | 63 | 69 | 110 |
|---|---|---|---|---|---|
| Plen ($\sigma = 10$) | 1.90 | 1.86 | 1.86 | 2.04 | 1.96 |
| Plen ($\sigma = 50$) | 1.46 | 1.43 | 1.66 | *1.60* | 1.75 |
| Plen ($\sigma = 100$) | *1.34* | 1.57 | 2.99 | 2.22 | 2.43 |
| NeuS | **0.98** | **0.56** | *1.13* | **1.45** | *1.43* |
| Ours | *1.34* | *1.36* | **0.99** | 1.91 | **1.37** |

Table 4. **Chamfer distance ↓ on DTU scenes.** We color the **best** and *second best* surfaces.

reasonable surfaces, but performs worse than NeuS overall. This is mainly due to a lack of natural spatial smoothness constraint present in the MLP architecture of NeuS, which allows it to perform well on datasets like DTU that contain many smooth surfaces, but worse on our synthetic dataset with a focus on thin structures.

We also note that although the qualitative comparison in Figure 17 shows that our method can refine the level set surfaces extracted from Plenoxels by correcting the outgrowing surfaces while preventing holes, the Chamfer distance does not always show an improvement. This is because the official DTU evaluation provides carefully created masks to remove reconstruction on parts that do not have proper reference geometry scanned by the depth scanner. This also excludes the majority of the inner surfaces from level set surfaces of Plenoxels, making their Chamfer distances much better; see Figure 13.

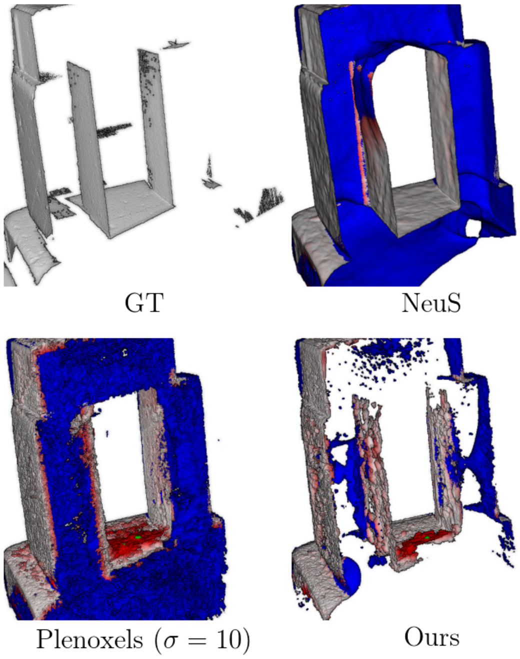

Figure 13. **Inside views of reconstructions on DTU dataset.** Red color indicates the L1 error in reconstruction, and blue indicates the reconstruction masked out by the DTU official masks. Surfaces extracted from Plenoxels contain many noisy inner surfaces that had to be masked out during evaluation to achieve low Chamfer distance.

**Experiment Details** We compared with level set surfaces from Plenoxels [42] and NeuS [54] trained with masks. We used the image masks provided by IDR [59] to set the background to white before training both Plenoxels and ours. For Plenoxels, we used the same hyperparameters for training on NeRF Synthetic dataset. We used slightly different hyperparameters from the ones we used for training NeRF Synthetic and Thin datasets. Namely, we modified the surface scalar learning rate to start with $10^{-4}$ and end with $10^{-6}$. We increased $\lambda_\delta$ to $0.05$, $\lambda_\mathcal{H}$ to $10^{-3}$ and $\lambda_\alpha$ to $10^{-8}$. We also kept the truncated alpha compositing parameter $a$ at 5 throughout training.

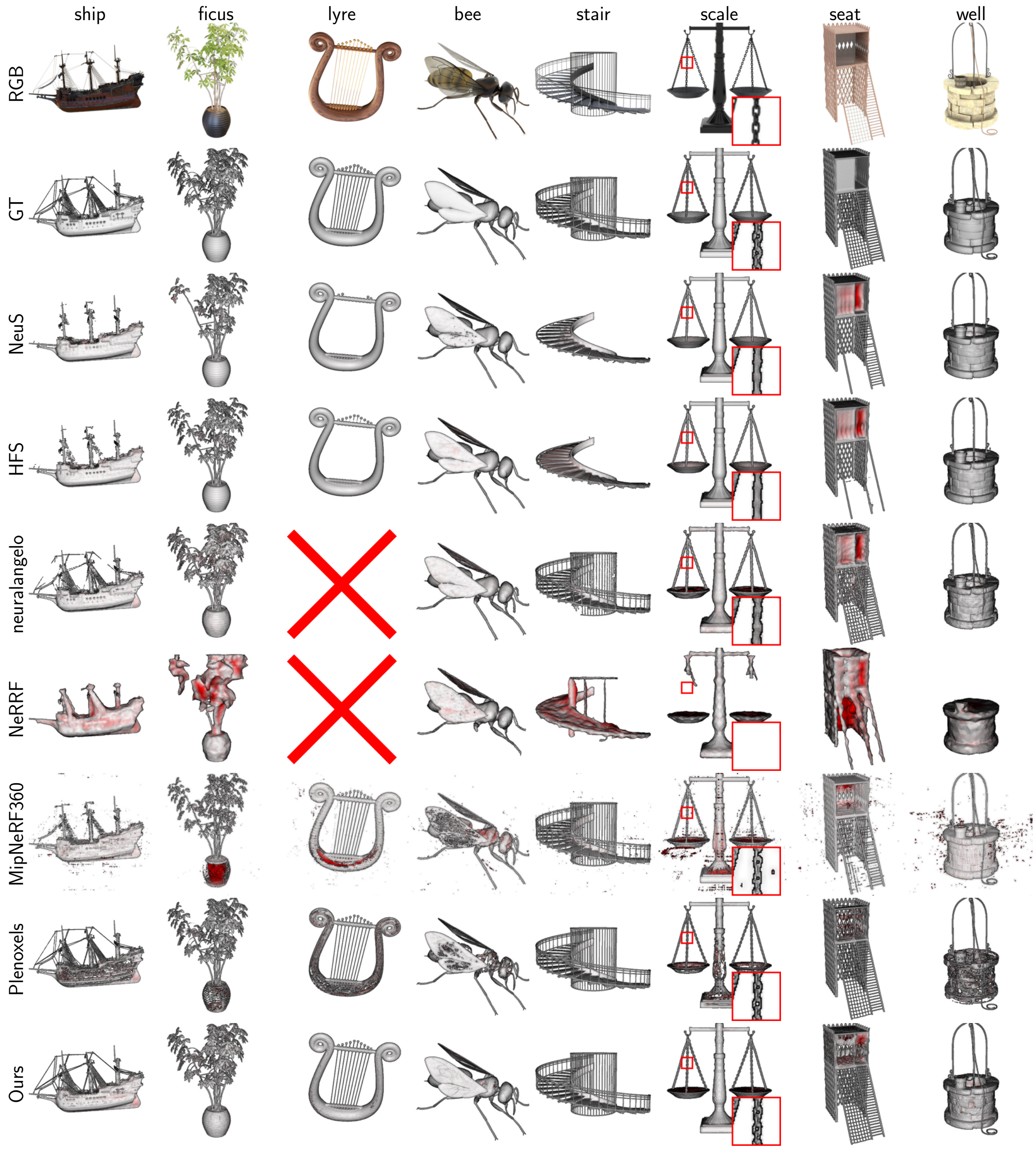

Figure 14. **Qualitative results on Thin Blender dataset.** Note that neuralangelo [28] failed to learn any surface on the "lyre" scene.

| | ship | ficus | lyre | bee | stair | scale | seat | well | avg |
|---|---|---|---|---|---|---|---|---|---|
| Plen ($\sigma = 90$) | 0.476 | 0.431 | 0.522 | 0.541 | 0.206 | 0.884 | 0.497 | 0.647 | 0.526 |
| Mip360 ($\sigma = 50$) | 1.217 | 2.640 | 1.311 | 1.181 | 0.279 | 2.243 | 0.744 | 1.941 | 1.445 |
| NeuS | 0.552 | 1.667 | 0.812 | 0.242 | 4.087 | 0.237 | 0.431 | 0.360 | 1.049 |
| HFS | 0.514 | 0.374 | 0.781 | 0.336 | 4.316 | 0.271 | 0.425 | 0.385 | 0.925 |
| neuralangelo | 0.270 | 0.411 | NaN | 0.377 | 0.426 | 0.789 | 0.432 | 0.266 | 0.424 |
| NeRRF | 1.74 | 2.899 | NaN | 1.164 | 3.731 | 1.359 | 2.17 | 3.381 | 2.349 |
| Ours | 0.277 | 0.240 | 0.188 | 0.207 | 0.176 | 0.575 | 0.288 | 0.319 | 0.284 |

Table 5. **Chamfer distance ↓ $\times 10^{-2}$ on Thin Blender datasets.** We color the best, second best methods.

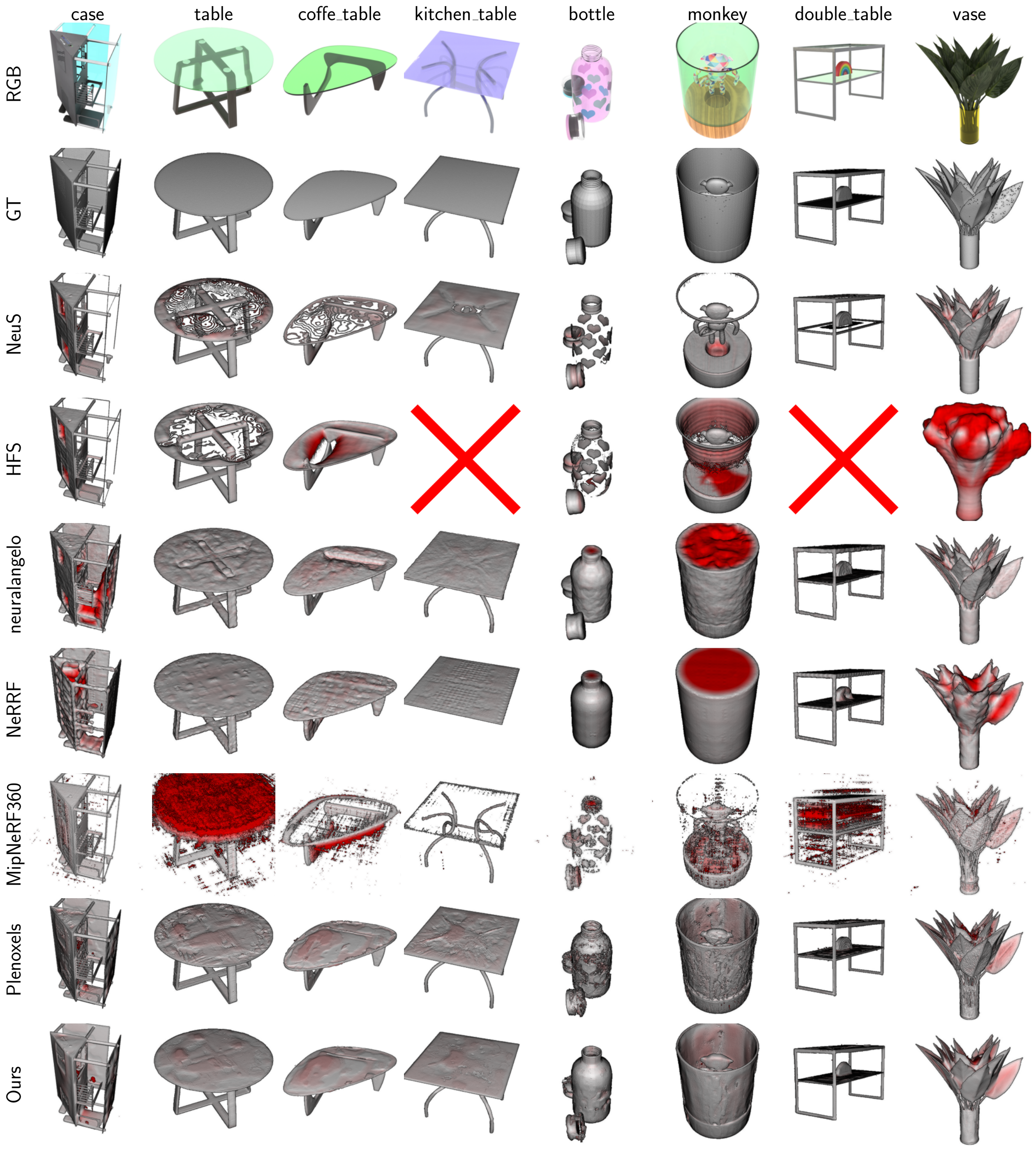

Figure 15. **Qualitative results on Translucent Blender dataset.** Note that HFS [56] fails to learn any surface on "kitchen table" and "double table" scenes. We removed some exterior surfaces in the "monkey" scene to show the interior surfaces.

| name | case | table | coffee | kitchen | bottle | monkey | double | vase | avg |
|---|---|---|---|---|---|---|---|---|---|
| Plen ($\sigma = 30$) | 1.195 | 0.438 | 0.706 | 0.378 | 0.709 | 1.084 | 0.557 | 1.024 | 0.761 |
| Mip360 ($\sigma = 50$) | 4.012 | 5.217 | 2.096 | 2.375 | 2.324 | 4.390 | 2.900 | 1.190 | 3.063 |
| NeuS | 5.091 | 1.188 | 1.070 | 0.392 | 2.271 | 5.923 | 0.874 | 1.946 | 2.344 |
| HFS | 5.094 | 0.854 | 2.839 | NaN | 1.493 | 3.223 | NaN | 8.684 | 3.698 |
| neuralangelo | 2.072 | 0.483 | 0.798 | 0.577 | 0.395 | 2.464 | 0.513 | 1.701 | 1.125 |
| NeRRF | 1.267 | 0.571 | 0.822 | 3.091 | 3.72 | 2.905 | 0.829 | 3.486 | 2.086 |
| Ours | 0.835 | 0.373 | 0.717 | 0.255 | 0.653 | 0.776 | 0.512 | 0.870 | 0.624 |

Table 6. **Chamfer distance ↓ $\times 10^{-2}$ on Semi-Transparent Blender datasets.** We color the best, second best methods. Note that HFS [56] fails to learn any surface on "kitchen table" and "double table" scenes.

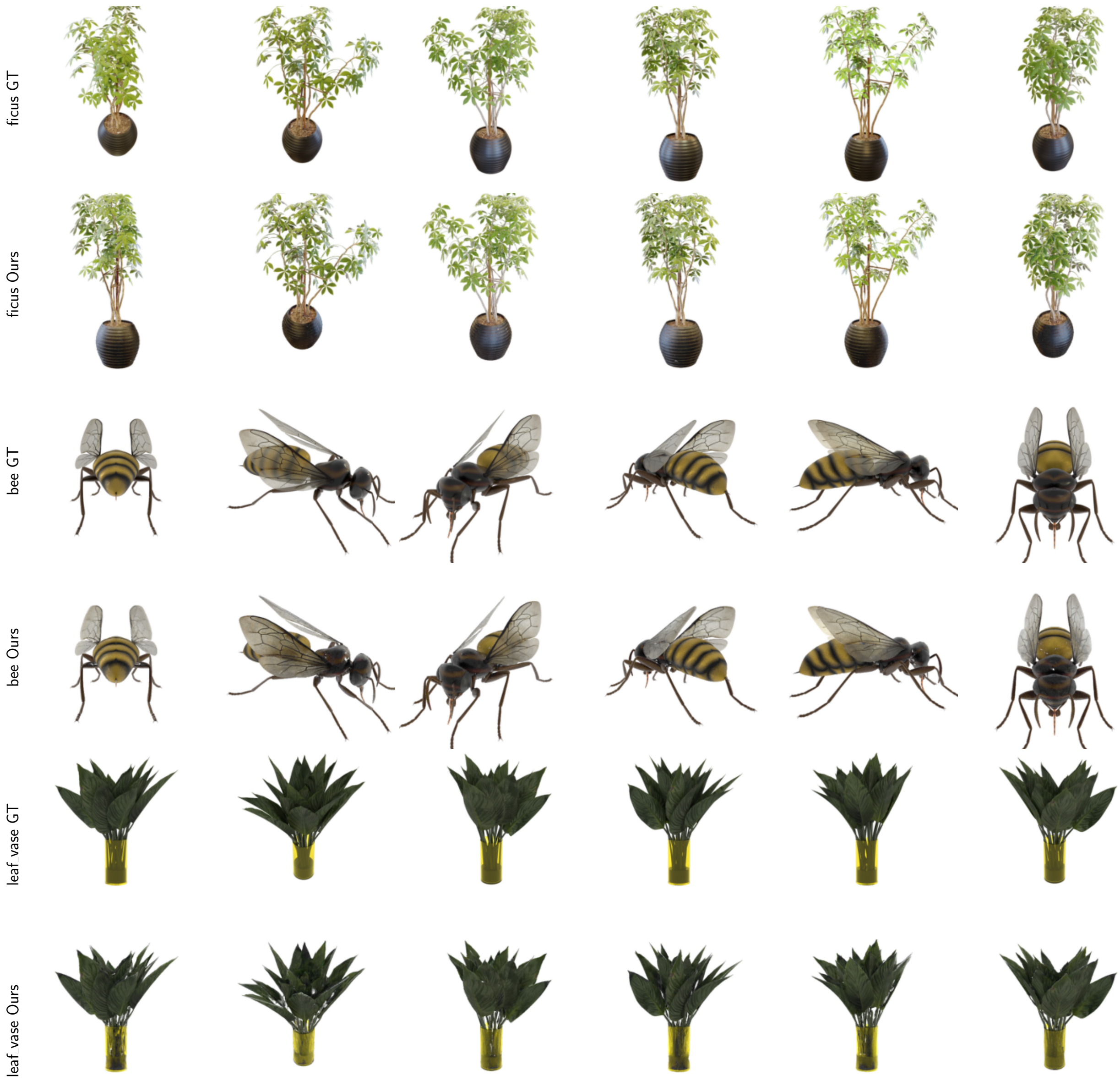

Figure 16. **RGB renderings of our methods on synthetic datasets.**

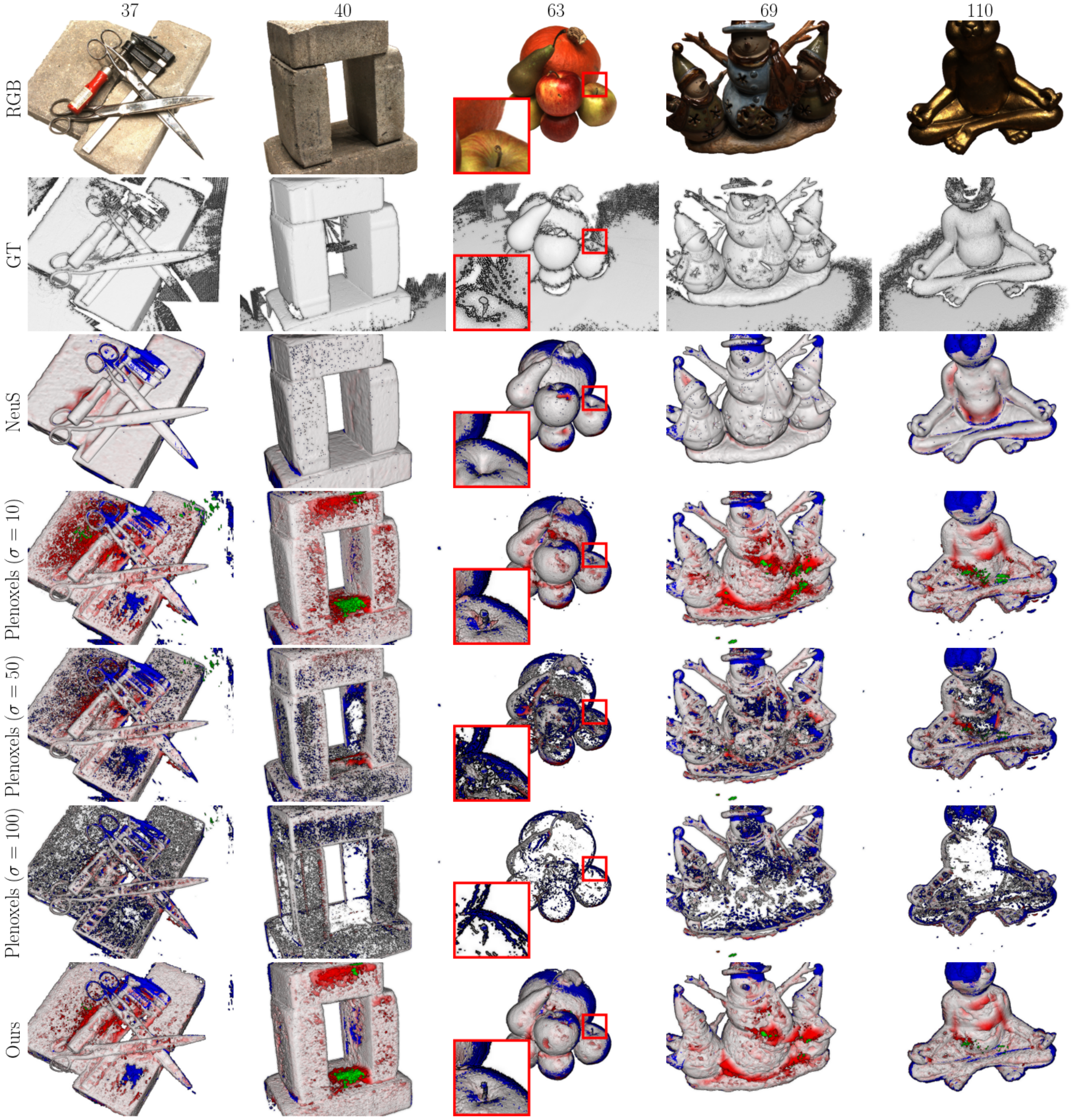
37
40
63
69
110
RGB
GT
NeuS
Plenoxels ($\sigma = 10$)
Plenoxels ($\sigma = 50$)
Plenoxels ($\sigma = 100$)
Ours

Figure 17. **Qualitative results on DTU [17] dataset.** As the DTU scenes mainly contain smooth surfaces without any semi-transparent materials, our method does achieve state-of-the-art performance on this dataset. However, note that our method can still accurately capture the thin structure that is missed by NeuS in Scan 63. Moreover, our method can effectively correct the out-growing surface artifacts in Plenoxels. Red color indicates the L1 error in reconstruction, blue indicates the reconstruction masked out by the DTU official masks, and green indicates reconstructions that are too far away from reference and hence clipped during evaluation.