αSurf: 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

Cengiz Oztireli
Google Research
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 solid surfaces and are therefore unable to properly reconstruct semi-transparent surfaces and thin structures, which also exhibit low opacity due to the blending effect with the background. While neural radiance field (NeRF) based methods can model semi-transparency and achieve photo-realistic quality in synthesized novel views, their volumetric geometry representation tightly couples geometry and opacity, and therefore cannot be easily converted into surfaces without introducing artifacts. We present αSurf, a novel surface representation with decoupled geometry and opacity for the reconstruction of semi-transparent and thin surfaces where the colors mix. Ray-surface intersections on our representation can be found in closed-form via analytical solutions of cubic polynomials, avoiding Monte-Carlo sampling and is fully differentiable by construction. Our qualitative and quantitative evaluations show that our approach can accurately reconstruct surfaces with semi-transparent and thin parts with fewer artifacts, achieving better reconstruction quality than state-of-the-art SDF and NeRF methods.[1]

1. Introduction

Recovering surfaces from RGB images is a complex and challenging task in computer vision, with numerous practical applications such as photogrammetry, 3D asset creation, and custom 3D fabrication. Traditional approaches rely on Structure-from-Motion [29] and Multi-View Stereo [30] pipelines to first reconstruct a 3D point set of the scene to which surfaces can be fitted [13, 15]. Differentiable rendering techniques have emerged as a more versatile reconstruction procedure. With properly derived gradients, these techniques can simultaneously learn all scene parameters 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 [11, 36, 42, 44], *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 semi-transparency effects. The prevailing assumption in earlier works is that the surface is solid throughout, and differentiable rendering techniques for implicit surfaces only examine the intersection of rays and the nearest surface [22, 11, 17]. As a result, the forward rendering process in those studies cannot simulate scenes with non-solid effects, leading to an inability to reconstruct their surfaces; see Figure 1a.

Modeling opaqueness is not solely crucial for reconstructing semi-transparent surfaces, but is also necessary for the recovery of extremely *thin surfaces*. As illustrated in Figure 2, when rendering a thin surface that only partially occupies a pixel with an insufficient number of sampled rays, solid 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 1a.

Differentiable volume rendering techniques [20] have demonstrated considerable success in rendering semi-transparent 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

[1] Preprint. Website: https://alphasurf.netlify.app/

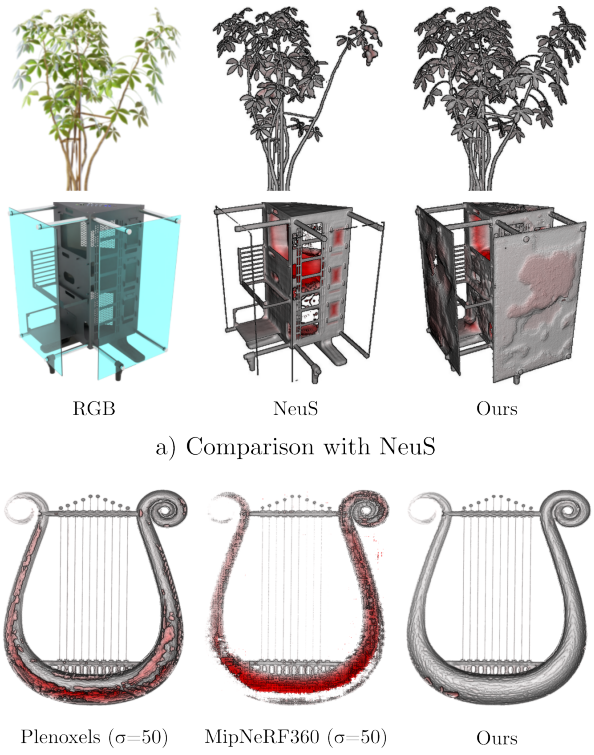


Figure 1: **Comparison with SDF and NeRF methods.** a) Due to the assumption of solid surfaces, SDF methods such as NeuS [37] fail to learn semi-transparent surfaces or thin structures with a strong blending effect. b) We compare the inside views of the reconstructed surfaces with NeRF-based methods [2, 28] by cropping in the middle. Surfaces from NeRF-based methods contain artifacts such as floaters or inner surfaces. In comparison, our approach can accurately reconstruct surfaces exhibiting semi-transparency while removing noisy surface artifacts. The red color in geometry indicates the error of the reconstruction in L1 distance.

by applying a density threshold. However, density represents occupancy (geometry) and transparency (material) in a tightly coupled manner, posing a non-trivial challenge in choosing the threshold for surface extraction. For example, a high threshold value may exclude transparent objects from the reconstruction, whereas a low threshold results in the erroneous reconstruction of density floaters that are not part of the desired surface; see Figure 1b. NeuS [37] attempts to alleviate this issue by combining SDF with volume rendering, but it still assumes a solid surface at the end of optimization and hence fails to reconstruct surfaces with semi-transparency; see Figure 1a.

In order to faithfully reconstruct surfaces of semi-transparent and thin objects, two key requirements must be fulfilled: 1) a representation that explicitly decouples geometry and materials; and 2) a differentiable rendering process that considers more than just the nearest intersection between the ray and the scene. 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 advantage of our approach is that the *exact* intersection points between a ray and all the surfaces, regardless of whether they are solid or transparent, can be determined by analytically solving a cubic polynomial. The *closed-form* solution allows for full *differentiability* 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

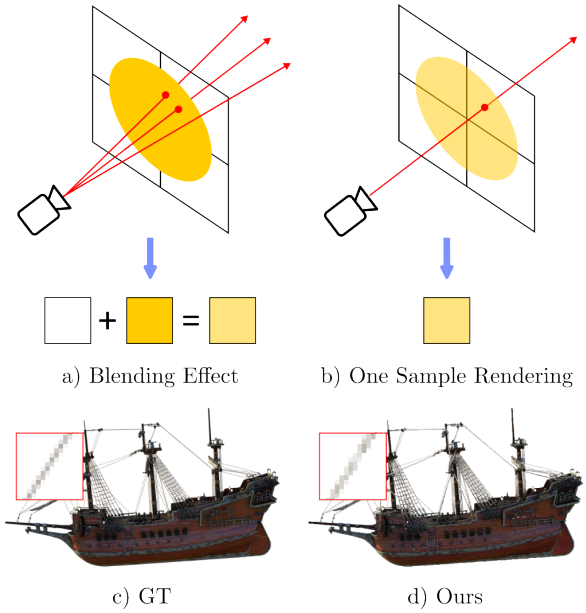


Figure 2: **Blending effect.** a) Real-world capture takes incoming light from multiple rays per pixel. Pixels that are partially occupied by a solid object are therefore rendered as a mixture of the object and background color. b) With one-sample rendering in reconstruction, it becomes necessary for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

method is around 30 minutes, including a short pre-training of Plenoxels [28] as initialization of our surface. We evaluate our method on an extended version of the NeRF synthetic dataset [20], which contains 8 original scenes and 16 additional objects with challenging thin structures or semi-transparent materials. We show that our method is capable of reconstructing surfaces with better overall quality than the existing SDF and NeRF based methods. We also qualitatively evaluate our method on real-world scenes from the LLFF dataset [19] and a scene featuring semi-transparent materials captured by us.

In summary, our contributions are: 1) A novel grid-based scene representation for implicit surface reconstruction, specifically tailored for semi-transparent and thin objects. It incorporates a closed-form and differentiable evaluation of all surface intersections along the ray, and properly decouples geometry and opacity. 2) A series of initialization and optimization strategies for efficient training and accurate reconstruction. 3) We show overall superior reconstruction quality compared to state-of-the-art methods on scenes featuring thin and semi-transparent objects.

2. Related Works

Neural Radiance Fields NeRF [20] is a 5D plenoptic function that models volume density and view-dependent appearance. Its differentiable volume rendering allows robust image-based 3D reconstruction and motivated an explosion of related works in areas including high-quality novel view synthesis [2, 35, 40, 45], 3D asset synthesis [5, 31], and efficient reconstruction and rendering [21, 26, 28, 34, 43]. Despite their outstanding novel view synthesis performance, many analyses suggest their geometry tends to produce artifacts such as sparse density floaters and inner volume [2, 35, 45]. Direct surface extraction on the density field hence suffers from those artifacts, whereas the depth extraction method does not guarantee watertight surfaces and requires additional surface reconstruction [4, 7, 13] . UNISURF [23] attempts to mitigate this issue by gradually transiting from volume rendering to surface rendering, but its surface is tightly coupled as a fixed level set on the opacity field and it still only works on solid objects. Our approach deviates from this by using a separate implicit surface field with decoupled opacity to model semi-transparent and thin surfaces.

SDF For Multi-view 3D Reconstruction SDF has been extensively employed with differentiable rendering methods to reconstruct surfaces from multi-view images. SDFDiff [11] uses a voxel grid and trilinear interpolation to represent the SDF and develops differentiable sphere tracing to find an estimated intersection. Our approach deviates from this by using a closed-form solution to solve for *exact* intersections with a more generalized implicit surface field, while also finding more than just the nearest intersections. IDR [42] proposes a differentiable sphere-tracing algorithm and optimizes the surface together with a volumetric BRDF. VolSDF [41] maps SDF to volume density via Laplacian CDF and optimizes the SDF via volume rendering. NeuS [37] 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 [18], fast reconstruction [38, 16, 27], and finer details modeling such as HFS [39], which applies a mapping from SDF to transparency and incorporates an additional displacement field to improve the reconstruction of fine details. A crucial and common limitation in existing SDF optimization methods is the assumption of surface solidity. Even the methods that utilize volume rendering in surface reconstruction, such as VolSDF and NeuS, still enforce convergence to solid surfaces in the end. Hence, they cannot properly reconstruct semi-transparent surfaces and suffer from thin structures with strong blending effects.

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 thin structures or semi-transparency. Towards this, we design 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 utilizes Plenoxels [28] to efficiently initialize a coarse surface (Figure 3d), and through the optimization of a photometric loss and additional surface regularization, we are able to reconstruct clean and accurate surfaces.

3.1. Representation

Surface We represent the surface as the level sets of a continuous scalar field $\delta : \mathbb{R}^3 \to \mathbb{R}$. 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:

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

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 sets on the scalar field. Specifically, given a set of 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 and values of $\boldsymbol{\tau}$ are determined through hyperparameter tuning. Note that this implicit surface field

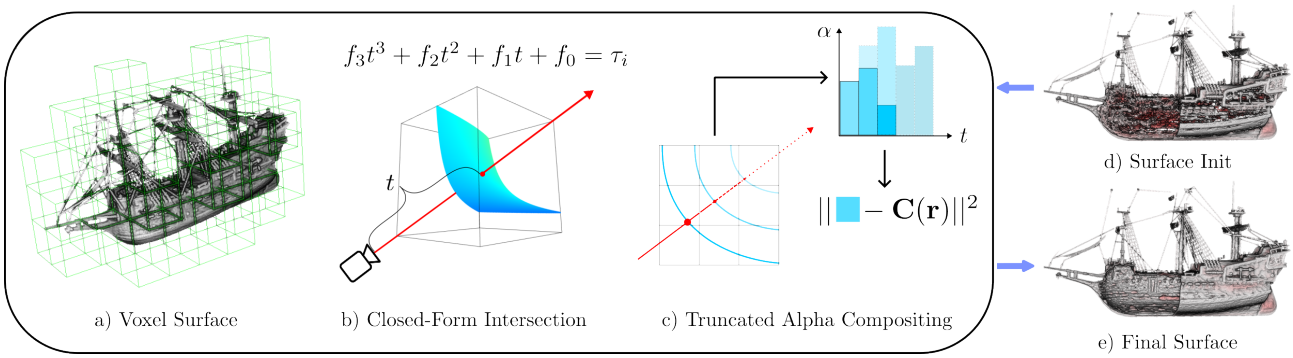


Figure 3: **Method.** a) Our surface representation is based on a voxel grid storing explicit values, without neural networks; see Section 3.1. b) We utilize a closed-form and differentiable method to compute ray-surface intersection. This is achieved by solving a cubic polynomial of depth t with known parameters $f_0, ..., f_3$ and τ_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 [28] to start with roughly correct yet noisy surfaces. e) The optimization results in clean and complete surfaces in the end.

does not model distance as in SDF, therefore, it is not subject to the Eikonal constraint.

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 [28]. $\mathbf{c}(\mathbf{x}, \mathbf{d})$ is represented via coefficients of the spherical harmonic (SH) function which 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 is only utilized where the 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. 19. for each voxel $v_{\mathbf{x}}$. By setting $\delta(\mathbf{x})$ to each level set value τ_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 and [8] for a detailed derivation. The real roots of this cubic polynomial can be found in closed-form via Vieta's approach [25], 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 α 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 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 [28], 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. 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:

$$\delta(\mathbf{x}) = \frac{\sigma(\mathbf{x}) - \tilde{\tau}_\sigma}{\overline{||\nabla\sigma||}} \tag{5}$$

$$\boldsymbol{\tau} = \left\{ \frac{\tau_\sigma - \tilde{\tau}_\sigma}{\overline{||\nabla\sigma||}} \mid \tau_\sigma \in \boldsymbol{\tau}_\sigma \right\} \tag{6}$$

where $\tilde{\tau}_\sigma$ is the median of the chosen raw level values, $\overline{||\nabla\sigma||}$ is the average norm of the finite difference gradient of the voxelated density field and 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.

As Plenoxels also uses a grid-based implementation, this initialization can be done very efficiently by directly taking and transforming the density values stored on the grids. Our initialization scheme allows the geometry information to be maximally preserved and inherited to the surface field. For example, some surfaces could have been modeled using density values slightly lower than the chosen level sets $\boldsymbol{\tau}_\sigma$ due to the ambiguity in NeRF training. After initialization, those spatial locations would have surface scalars very close to $\min(\boldsymbol{\tau})$, regardless of their actual distance to the initialized surfaces. This is a strong indication of potentially missed surfaces and the later optimization can easily update the surface scalars to allow them to emerge back again.

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

$$\sigma_\alpha = s_\sigma \, \sigma \tag{7}$$

$$\alpha = 1 - \exp\left(-\operatorname{ReLU}(\sigma_\alpha)\right) . \tag{8}$$

Note that this activation resembles the mapping from discretized sample density to opacity in volume rendering without the step size term [20]. 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_σ is to ensure that initialized raw opacities do not map to α with too high values after removing the step size term, causing the gradients to be saturated. Note that, during training, we update σ_α as the training parameters rather than α. After initialization, σ from NeRF is discarded and our subsequent optimization is carried out on σ_α. The SH coefficients are also initialized from the pre-trained Plenoxels and further optimized.

In comparison to our approach, SDF methods such as NeuS [37] 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 [11, 36], algorithms such as fast sweeping [6, 46] are required to explicitly assign the grid values as distance to closest surface [1, 32, 33]. 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 As shown in [35], a critical artifact in vanilla NeRF-like 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. This leads to very noisy inner surfaces when initialized from Plenoxels; see Figure 1b and 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:

$$t_i \in \mathcal{T} \text{ if } \underbrace{\mathbf{n}(t_i) \cdot \mathbf{d} \leq 0}_{\text{back-face culling}} \tag{9}$$

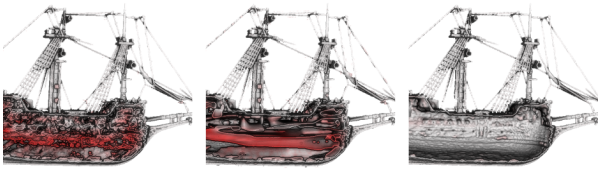
where t_i is an original intersection, $\mathbf{n}(t_i) = \frac{-\nabla\delta(t_i)}{||\nabla\delta(t_i)||}$ is the surface normal, and $\mathbf{n}(t_i) \cdot \mathbf{d}$ checks whether the surface is facing backwards. This is similar to back-face culling in rendering. We then apply additional constraints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

$$\alpha^*(t_i) = \gamma(i-1)\,\alpha(t_i) \tag{10}$$

$$\gamma(x) = \left(1 - \cos(\pi \operatorname{clamp}(a - x, 0, 1))\right)/2 \tag{11}$$

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. First, a surface convergence loss is applied to each ray-surface intersection with non-trivial truncated opacity to encourage different level sets to converge



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

Figure 4: Reconstruction without regularization tends to have redundant inner surfaces. Encouraging consistent normals in the local neighborhood via $\mathcal{L}_\mathbf{n}$ enforces smoother surfaces, but the redundant surfaces still remain. With both normal regularization and TV loss applied on the surface scalar field, we can obtain clean and smooth reconstruction.

together, as a single level set is sufficient to represent multiple disconnected surface patches:

$$\mathcal{L}_c(\mathbf{r}) = \sum_{t_i \in \mathcal{T}} |\tilde{t} - t_i| \, \mathbb{I}[\alpha^*(t_i) > 1\text{e}{-8}] \tag{12}$$

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 the sparse density floaters by encouraging them to move towards the actual surface location. The surface is also smoothed via a combination of an L1 normal smoothness regularization and a total variation (TV) loss applied on the surface field:

$$\begin{aligned} \mathcal{L}_\mathbf{n} &= \frac{1}{|\mathcal{V}|} \sum_{\mathbf{x} \in \mathcal{V}} |\nabla \mathbf{n}(\mathbf{x})| \\ \mathcal{L}_\delta &= \frac{1}{|\mathcal{V}|} \sum_{\mathbf{x} \in \mathcal{V}} ||\nabla \hat{\delta}(\mathbf{x})|| \end{aligned} \tag{13}$$

where $\mathcal{V}$ contains the spatial coordinates of all voxels, $\nabla\hat{\delta}(\mathbf{x})$ is the surface gradient at vertices 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 $\mathcal{L}_\mathbf{n}$ 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 Figure 4. 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 [14] on each ray and an L1 sparsity loss:

$$\mathcal{L}_\mathcal{H}(\mathbf{r}) = -\sum_{t_i \in \mathcal{T}} \bar{w}(t_i) \log(\bar{w}(t_i)) \tag{14}$$

$$\text{where } \bar{w}(t_i) = \frac{T^*_\alpha(t_i)\alpha^*(t_i)}{\sum_{t_j \in \mathcal{T}} T^*_\alpha(t_j)\alpha^*(t_j)} \tag{15}$$

$$\mathcal{L}_\alpha = \frac{1}{|\mathcal{V}'|} \sum_{\mathbf{x} \in \mathcal{V}'} |\operatorname{ReLU}(\sigma_\alpha(\mathbf{x}))| \tag{16}$$

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 effects, but we empirically found that having this term with a small weight can help remove the noise in the surface opacity.

Overall, the optimization target is:

$$\begin{aligned} \mathcal{L} = \frac{1}{|B|} \sum_{\mathbf{r} \in B} \Big(&||C(\mathbf{r}) - \hat{C}^*(\mathbf{r})||^2 + \lambda_c \mathcal{L}_c(\mathbf{r}) \\ &+ \lambda_\mathbf{n} \mathcal{L}_\mathbf{n} + \lambda_\delta \mathcal{L}_\delta + \lambda_\mathcal{H} \mathcal{L}_\mathcal{H}(\mathbf{r}) + \lambda_\alpha \mathcal{L}_\alpha \Big) \end{aligned} \tag{17}$$

where $\lambda_c, \lambda_\mathbf{n}, \lambda_\delta, \lambda_\mathcal{H}, \lambda_\alpha$ are hyperparameters.

name	NeRF	Thin	Semi-T	Avg
Plen ($\sigma = 10$)	1.212	0.595	1.337	1.048
Plen ($\sigma = 50$)	0.944	0.609	1.344	0.966
Plen ($\sigma = 100$)	0.977	0.548	1.990	1.172
Mip360 ($\sigma = 10$)	1.699	1.796	4.900	2.798
Mip360 ($\sigma = 50$)	1.805	1.091	5.170	2.689
Mip360 ($\sigma = 100$)	1.928	1.277	6.098	3.101
NeuS	1.798	0.834	2.313	1.648
HFS	0.550	1.086	2.090	1.242
Ours	0.534	0.366	0.843	0.581
Plen (depth 0.1)	0.590	0.441	0.974	0.668
Mip360 (depth 0.9)	0.396	0.320	1.414	0.710
Mip360 (depth 0.1)	2.047	0.588	2.346	1.660
Ours (trimmed)	0.396	0.304	0.680	0.460

Table 1: **Chamfer distance** $\downarrow$ $\times 10^{-2}$ **on synthetic datasets.** We color the best, second best watertight surfaces and the best non-watertight surfaces. Our method produces higher quality surfaces of both types than the baselines.

4. Evaluation

We quantitatively and qualitatively evaluate our method on an extended version of the NeRF synthetic dataset [20], with 8 additional objects with delicate and thin structures, and 8 objects with semi-transparent materials. We

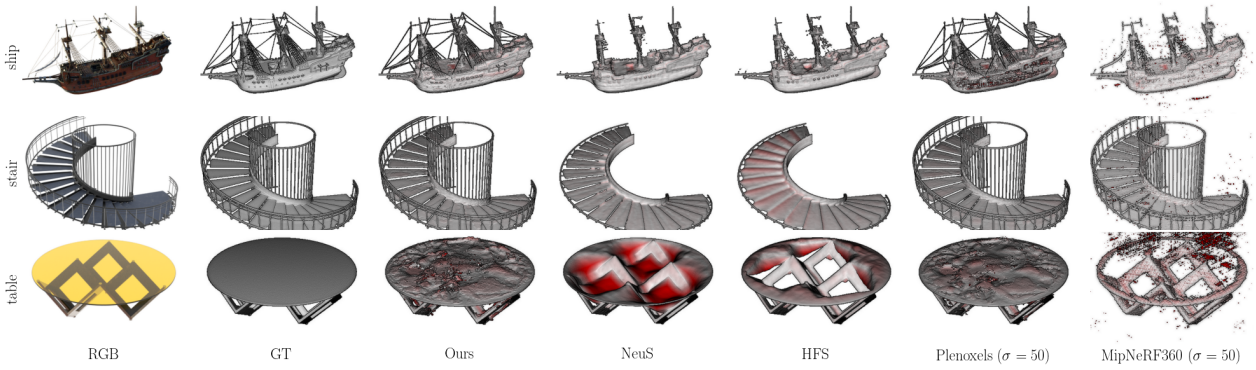


Figure 5: **Qualitative evaluation on synthetic datasets.** Red color indicates the L1 error in the reconstruction. Our method can reconstruct the semi-transparent and thin surfaces missed in NeuS and HFS, while recovering more complete and noise-free surfaces compared to NeRF-based methods. Results on additional scenes and non-watertight surfaces can be found in the supplementary.

also show qualitative comparison on challenging real-world scenes from the LLFF dataset [19] and captured by our own. We compare with recent SDF optimization methods including NeuS [37] and HFS [39], and both density level set and depth extracted geometry from Plenoxels [28] and MipNeRF360 [2].

4.1. Datasets

NeRF Synthetic NeRF synthetic dataset contains 8 objects with various challenging properties such as high specularity and thin structures. The synthetic objects are rendered with Blender [3] to obtain RGB images. We sampled 100 different training views from a full sphere and re-rendered the original scenes to make the bottom part visible, except for "ship" scene we also removed the bottom tank which blocks too many views. For each scene, we also rendered the depth, which we then converted to dense point clouds for quantitative geometry evaluation. This removes any invisible inner structures in the 3D assets.

Thin & Semi-Transparent Blender We build two additional synthetic datasets featuring objects with thin structures and semi-transparent materials respectively. The renderings and ground truth geometry of those additional scenes are obtained in the same way as the above NeRF synthetic dataset. We will release all the datasets with reference geometry upon publication.

LLFF [19] We qualitatively evaluate our method on scenes from the LLFF dataset, which contains forward-facing captures of challenging real-world scenes. As we are not focusing on the novel view synthesis task, we use all the available images for training. We additionally captured a scene featuring semi-transparent materials and processed it using a similar scheme to LLFF.

4.2. Implementation Details

Following Plenoxels [28], we use a sparse voxel grid of size 512^3 where each vertex stores the surface scalar δ, raw opacity σ_α and 9 SH coefficients. We directly initialize all the grid values from Plenoxels pre-trained with provided hyperparameters and prune voxels with densities σ lower than 5. We use $s_\sigma = 0.05$ to downscale the density values during initialization. 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 $10k$ iterations. For experiments on real-world scenes, we initialize a single level set $\boldsymbol{\tau}_\sigma = \{10\}$. A detailed list of hyperparameters can be found in the supplementary. 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).

4.3. Baselines

We compared our method with the state-of-the-art SDF-based reconstruction methods, as well as NeRF-based methods with level set and mode depth-extracted geometry.

SDF Methods We compare with NeuS [37] and HFS [39]. HFS is a follow-up work on NeuS and achieves more detailed geometry reconstruction. We did not include IDR [42], UNISURF [23] or VolSDF [41] as NeuS and HFS have shown superior performance.

NeRF Methods We compared with Plenoxels [28] (referred to as "Plen" in the table) and MipNeRF 360 [2] (referred to as "Mip360"), an improved version of NeRF with conical frustum sampling and weight-concentration regularization. In addition to level set surfaces, we also compared with mode depth-extracted surfaces from NeRF methods.

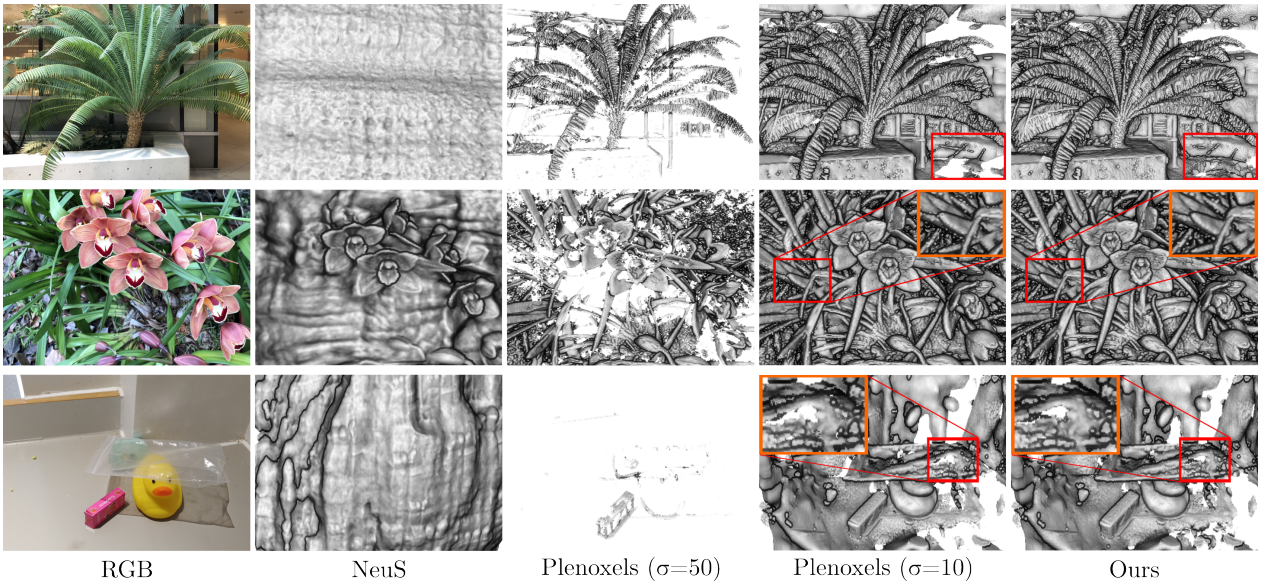


Figure 6: **Qualitative evaluation on real-world scenes.** We show the reconstructed surfaces on LLFF [19] scenes and a scene captured by us. NeuS fails to reconstruct sensible surfaces, while ours can reconstruct more complete surfaces with fewer artifacts.

We send rays from 100 spiral views from elevation $-180°$ to $180°$ to cover the whole object, and extract samples with the highest weight along each ray. We found this approach gives the best performance compared to mean or median depth extraction. We also mask out samples extracted from rays with accumulation weight less than 0.1 from Plenoxels, and less than either 0.1 or 0.9 from MipNeRF 360.

4.4. Evaluation on Synthetic dataset

We quantitatively evaluate our method on all three synthetic datasets and report the Chamfer-L1 distance as used in DTU [10] in Table 1. We followed the evaluation protocol of DTU [10], where we extracted dense point clouds from reconstructions and downsampled both predicted and ground truth points with 0.001 density before computing the Chamfer distance. HFS failed and learned empty surfaces on two scenes in the Thin Blender dataset, hence we removed them when calculating its average Chamfer. Our non-trimmed single surface provides better watertight surfaces than the watertight baselines. Our trimmed representation further improves the Chamfer distance by removing redundant surfaces trapped at low opacity region, also giving better or comparable surfaces than the depth-extracted surfaces from NeRF-based methods.

We show qualitative results of watertight surfaces in Figure 5. While NeuS and HFS are capable of reconstructing very smooth surfaces, they miss the thin structures and do not perform well on semi-transparent materials. Level set surfaces from Plenoxels and MipNeRF360 contain holes, floaters, and inner surfaces; see Figure 1b for an inside view of the surfaces. Our approach is capable of reconstructing those surfaces with minimal artifacts.

4.5. Evaluation on Real World Dataset

We show the qualitative evaluation of watertight surfaces on real-world scenes in Figure 6. NeuS fails to reconstruct proper surfaces and models everything as nearly flat surfaces due to the limited views. Plenoxels can recover surfaces with high-quality details through a low density level value, but tends to miss some surfaces and produce noisy density floaters. By initializing from Plenoxels and further optimizing with surface regularizations, our approach reconstructs surfaces with fewer artifacts. Note that although Plenoxels with level value $\sigma = 10$ gives views with decent quality, it tends to leave many redundant surfaces inside the object. This is not well reflected due to the forward-facing nature of the dataset, but can be seen from our evaluation on synthetic datasets.

4.6. Ablation

We show the ablation study on the watertight surfaces from our method in Table 2. The truncated alpha compositing deals with the inner volume artifacts inherited from initialization, while the surface regularization $\mathcal{L}_{\mathbf{n}}, \mathcal{L}_{\delta}$ and convergence loss $\mathcal{L}_c$ encourage smooth and complete surfaces. Using a single level set $\boldsymbol{\tau}_{\sigma} = \{10\}$ to initialize the surface may fail to capture all information in the pre-trained Plenoxels and causes artifacts in optimization. Ours with all tech-

	w/o $\mathcal{L}_\delta$	w/o $\mathcal{L}_\delta$, $\mathcal{L}_\mathbf{n}$	w/o trunc	w/o $\mathcal{L}_c$	$\tau_\sigma = \{10\}$	Ours
ficus	0.828	1.121	0.922	0.467	0.637	**0.389**
lyre	0.586	0.702	0.583	0.242	0.404	**0.182**

Table 2: **Quantitative results of ablation study.** We report the Chamfer distance $\times 10^{-2}$ of our watertight surfaces.

niques enabled achieves the best overall performance. More qualitative results can be found in the supplementary.

5. Conclusion

We present αSurf, a grid-based implicit 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 coarse initialization from efficient Plenoxels [28], and incorporates truncated rendering and additional surface regularizations to reconstruct high-quality surfaces for semi-transparent objects and thin structures with heavy blending effects.

Limitation Compared to MLP-based SDF methods such as NeuS [37], 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: Using stronger surface regularization can certainly enhance the smoothness, but also it destroys thin structures in the reconstruction. Adapting to a hybrid representation similar to Instant-NGP [21] where the surface scalars are obtained via latent codes and shallow MLP might help with this issue. Our representation also requires a separate background model to work on 360°real-world scenes, but this can be implemented very easily using a similar approach to [28, 45].

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. 2, 3, 7, 13

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

[4] 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*, SIGGRAPH '96, page 303–312, New York, NY, USA, 1996. Association for Computing Machinery. 3, 13

[5] 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

[6] 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

[7] 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, 13

[8] 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, September 2022. 4, 11

[9] Geoffrey Hinton. Rmsprop. 12

[10] 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. 8, 13, 15, 22

[11] 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)*, June 2020. 1, 3, 5, 11

[12] Jzhangbs. Jzhangbs/dtueval-python: A fast python implementation of dtu mvs 2014 evaluation. 13

[13] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In Alla Sheffer and Konrad Polthier, editors, *Symposium on Geometry Processing*. The Eurographics Association, 2006. 1, 3, 13

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

[15] D. Lee and B. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Parallel Programming*, 9:219–242, 06 1980. 1

[16] 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

[17] 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

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

[19] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 3, 7, 8, 14

[20] 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

[21] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 3, 9

[22] 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

[23] 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. 3, 7

[24] 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. 13

[25] 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, 12

[26] 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

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

[28] 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, 9, 12, 13, 15

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

[30] 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

[31] 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

[32] 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

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

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

[35] 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

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

[37] 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, 9, 13, 15

[38] 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

[39] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. *arXiv preprint arXiv:2206.07850*, 2022. 3, 7, 13, 18, 19

[40] 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*, November 2021. 3

[41] 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, 7

[42] 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, 7, 15

[43] 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

[44] 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

[45] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 3, 9

[46] 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. Code and Data

We attach the source code, the example training script, as well as an example scene in the zip file. Due to the size limit, we are unable to attach the ground truth geometry. Full data will be released upon publication.

C. Implementation Details

C.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 [8, 11], 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 τ_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 $v_\mathbf{x}$:

$$\begin{aligned}\delta(\mathbf{x}) =& \text{trilearp}(\mathbf{x}, \{\hat{\delta}_i\}_{i=1}^8) \qquad (18)\\ =&(1-z)((1-y)((1-x)\hat{\delta}_1 + x\hat{\delta}_5)\\ &+ y((1-x)\hat{\delta}_3 + x\hat{\delta}_7))\\ &+ z((1-y)((1-x)\hat{\delta}_2 + x\hat{\delta}_6)\\ &+ y((1-x)\hat{\delta}_4 + x\hat{\delta}_8)) \qquad (19)\end{aligned}$$

where $[x, y, z] = \mathbf{x} - \mathbf{l}$ are the relative coordinates within the voxel, and $\mathbf{l} = \text{floor}(\mathbf{x})$. 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 19:

$$\begin{aligned}\tau_i =&(1-(o'_z+t'd_z))((1-(o'_y+t'd_y))\\ &((1-(o'_x+t'd_x))\hat{\delta}_1 + (o'_x+t'd_x)\hat{\delta}_5)\\ &+ (o'_y+t'd_y)((1-(o'_x+t'd_x))\hat{\delta}_3 + (o'_x+t'd_x)\hat{\delta}_7))\\ &+ (o'_z+t'd_z)((1-(o'_y+t'd_y))\\ &((1-(o'_x+t'd_x))\hat{\delta}_2 + (o'_x+t'd_x)\hat{\delta}_6)\\ &+ (o'_y+t'd_y)((1-(o'_x+t'd_x))\hat{\delta}_4 + (o'_x+t'd_x)\hat{\delta}_8))\,. \qquad (20)\end{aligned}$$

By re-arranging the equation, we obtain:

$$\tau_i = f_3 t'^3 + f_2 t'^2 + f_1 t' + f_0 \qquad (21)$$

where

$$\begin{aligned}f_0 =&(m_{00}(1-o'_y) + m_{01}(o'_y))(1-o'_x)\\ &+ (m_{10}(1-o'_y) + m_{11}(o'_y))(o'_x)\\ f_1 =&(m_{10}(1-o'_y) + m_{11}(o'_y))d_x + k_1(o'_x)\\ &- (m_{00}(1-o'_y) + m_{01}(o'_y))d_x + k_0(1-o'_x)\\ f_2 =&k_1 d_x + h_1(o'_x) - k_0 d_x + h_0(1-o'_x)\\ f_3 =&h_1 d_x - h_0 d_x \qquad (22)\end{aligned}$$

and

$$\begin{aligned}m_{00} =&\hat{\delta}_1(1-o'_z) + \hat{\delta}_2(o'_z)\\ m_{01} =&\hat{\delta}_3(1-o'_z) + \hat{\delta}_4(o'_z)\\ m_{10} =&\hat{\delta}_5(1-o'_z) + \hat{\delta}_6(o'_z)\\ m_{11} =&\hat{\delta}_7(1-o'_z) + \hat{\delta}_8(o'_z)\\ k_0 =&(m_{01}d_y + d_z(\hat{\delta}_4 - \hat{\delta}_3)(o'_y))\\ &- (m_{00}d_y - d_z(\hat{\delta}_2 - \hat{\delta}_1)(1-o'_y))\\ k_1 =&(m_{11}d_y + d_z(\hat{\delta}_8 - \hat{\delta}_7)(o'_y))\\ &- (m_{10}d_y - d_z(\hat{\delta}_6 - \hat{\delta}_5)(1-o'_y))\\ h_0 =&d_y d_z(\hat{\delta}_4 - \hat{\delta}_3) - d_y d_z(\hat{\delta}_2 - \hat{\delta}_1)\\ h_1 =&d_y d_z(\hat{\delta}_8 - \hat{\delta}_7) - d_y d_z(\hat{\delta}_6 - \hat{\delta}_5)\,. \qquad (23)\end{aligned}$$

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 [8] 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 [25] to solve the real roots for t' in an analytic way. Namely, we first re-write the cubic polynomial as follows:

$$\tau_i = t'^3 + at'^2 + bt' + c \tag{24}$$

$$a = \frac{f_2}{f_3}, b = \frac{f_1}{f_3}, c = \frac{f_0}{f_3}\,. \tag{25}$$

Then, compute:

$$Q = \frac{a^2 - 3b}{9} \tag{26}$$

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

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

$$\theta = \arccos(\frac{R}{\sqrt{Q^3}}) \tag{28}$$

$$t'_1 = -2\sqrt{Q}\cos(\frac{\theta}{3}) - \frac{a}{3} \tag{29}$$

$$t'_2 = -2\sqrt{Q}\cos(\frac{\theta - 2\pi}{3}) - \frac{a}{3} \tag{30}$$

$$t'_3 = -2\sqrt{Q}\cos(\frac{\theta + 2\pi}{3}) - \frac{a}{3} \tag{31}$$

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:

$$A = -\operatorname{sign}(R)\left(|R| + \sqrt{R^2 - Q^3}\right)^{1/3} \tag{32}$$

$$B = \begin{cases} Q/A, & \text{if } A \neq 0 \\ 0, & \text{otherwise} \end{cases}\,. \tag{33}$$

Then, the only real root can be obtained as:

$$t'_1 = (A + b) - \frac{a}{3}\,. \tag{34}$$

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.

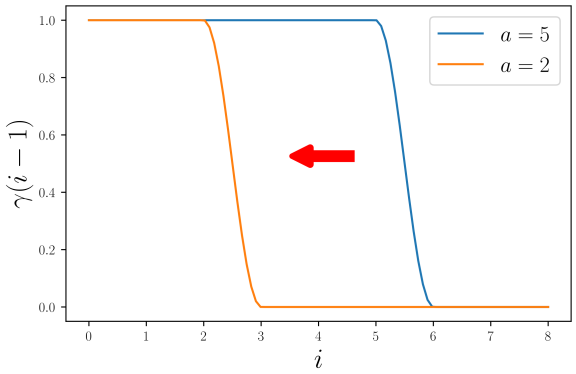


Figure 7: **The truncated alpha compositing reweight function $\gamma(i-1)$.** The x-axis is the index of the intersection starting from 1 (excluding intersections on back-facing surfaces). By reducing a, we effectively slide the curve to the left.

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.

C.2. Hyperparameters

Our code is based on Plenoxels [28]. 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. For surface scalars $\hat{\delta}$, we use 10^{-5} as both starting and end learning rate. For raw opacity values σ_α, 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 [9] optimizer for training.

We used grid search to determine the optimal hyperparameters. For the regularization weights, we set $\lambda_c = 10^{-6}$ for the first $10k$ iterations and 0 for the rest of training. We use $\lambda_{\mathbf{n}} = 10^{-6}$, $\lambda_\delta = 10^{-3}$, $\lambda_{\mathcal{H}} = 10^{-4}$ and $\lambda_\alpha = 10^{-9}$. For the implementation of surface TV loss $\mathcal{L}_\delta$, we calculate the gradient via forward finite difference in the same way as Plenoxels [28]:

$$\nabla_x\hat{\delta}(i,j,k) = \frac{|\hat{\delta}(i+1,j,k) - \hat{\delta}(i,j,k)|D_x}{256} \tag{35}$$

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.

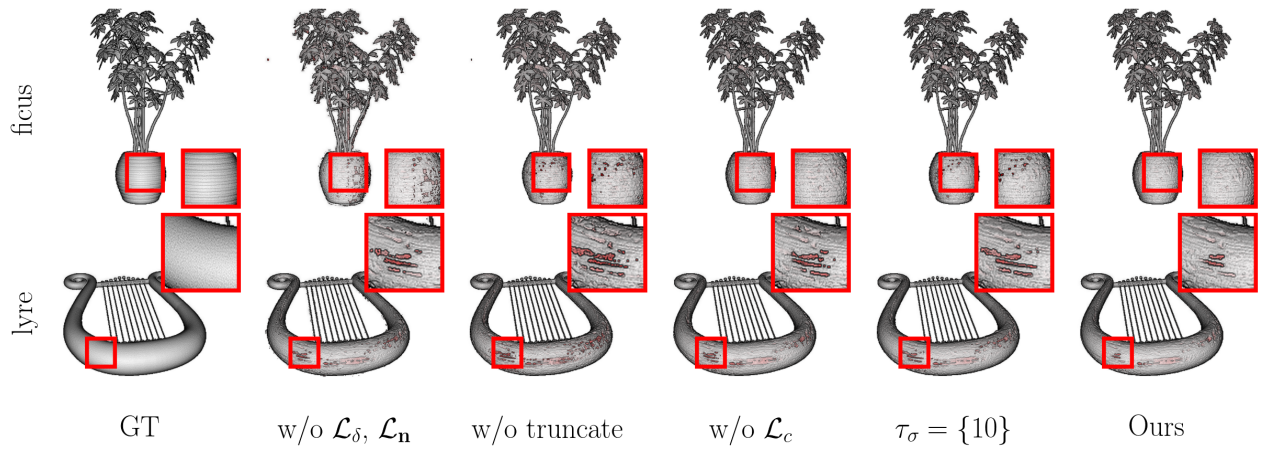


Figure 8: **Ablation Study.** We show the qualitative results of different ablations. Our full approach achieves the best quality overall.

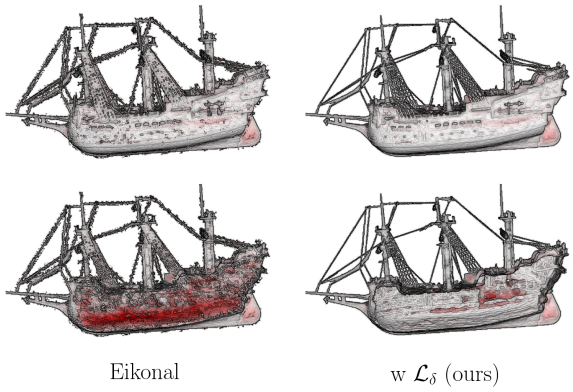


Figure 9: **Comparison with the Eikonal constraint regularization.** We visualize the out view (first column) and the inside view (second column) by cropping the surfaces along the y-axis. It can be clearly seen that the Eikonal constraint does not regularize the surface to be clean and smooth, but rather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

The truncated alpha compositing reweight function can be seen as a truncated Hann window [24], 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.

D. Additional Experiments

D.1. Synthetic Dataset

Experiment Details For training of NeuS [37], HFS [39] 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 Semi-Transparent Blender dataset, we set the background to white for all methods as the semi-transparent objects are rendered with a white background in Blender.

For the Chamfer distance evaluation on the synthetic datasets, we adapt the Python version of DTU [10] evaluation script [12]. For evaluation of NeuS [37] and HFS [39], 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 the level set surface of Plenoxels [28], MipNeRF 360 [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 depth-extracted surfaces from Plenoxels [28] and MipNeRF 360 [2], to ensure the evaluation is done on proper surfaces and deal with the potential imbalance in the distribution of depth-extracted points, we first downsampled the points with density 0.001, then ran alpha shapes [7] with $\alpha = 0.003$ to reconstruct meshes for Chamfer distance evaluation. We did not use TSDF [4] or Poisson surface reconstruction [13], as the former tends to remove delicate structures and the latter can produce many redundant surfaces and requires careful pruning. Overall, we found out that alpha shape gives the best result when reconstructing surfaces from very densely sampled points.

Additional Results We show all the qualitative results in Figure 12, 13, 14, 15, 16 and 17, as well as the individual Chamfer distance for each scene in Table 4, 5 and 6. Qual-

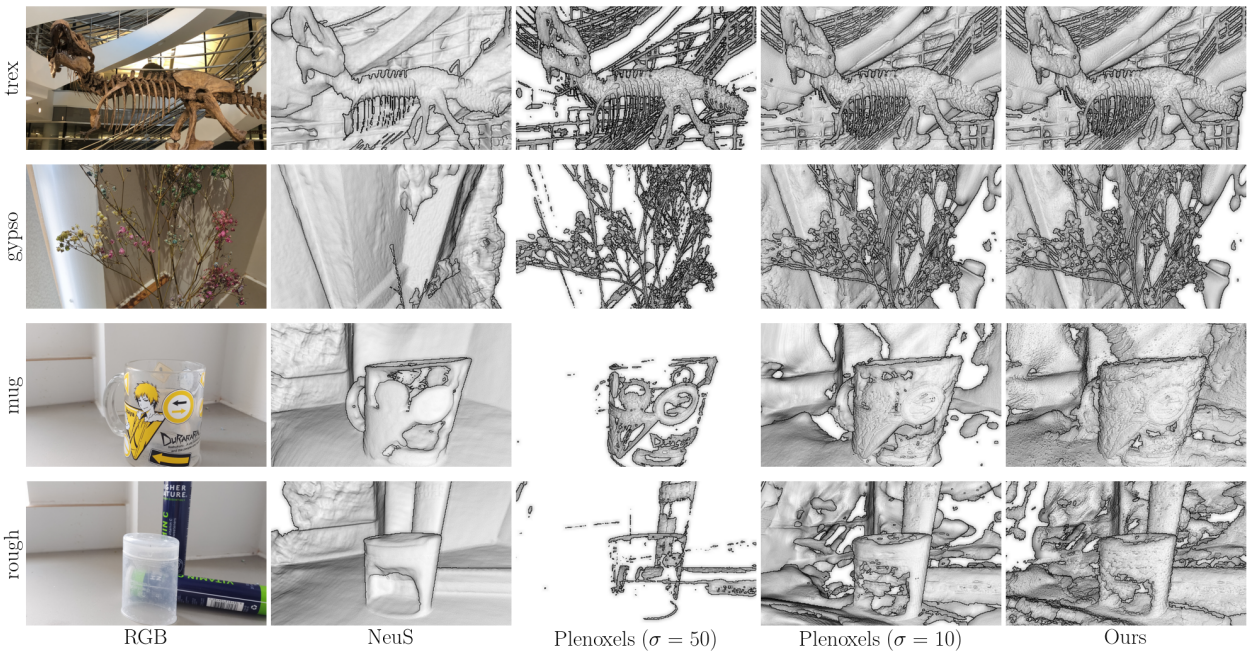


Figure 10: **Additional results on real-world scenes.** Note that the geometry renderings do not exactly match the RGB due to the use of normalized device coordinates.

itative results of MipNeRF 360 with level set $\sigma = 100$ are not included due to space constraint, but note that MipNeRF 360 $\sigma = 50$ already starts to fail and present surfaces with severe holes.

We note that in some scenes, although reconstructions from NeuS and HFS missed a significant part of the thin structures, they turned out to give comparable or lower Chamfer distances than our reconstruction. This is particularly obvious in scenes such as "ficus" (the branches and leaves), "seat" (the lattice), "well" (the rope), and "slide" (bars on the ladder). The reason behind this is that our method might have learned slightly incorrect locations for some smooth surfaces in the scene, but as the thin structures occupy a relatively small portion of the whole geometry, the error caused by missing them is significantly down-weighted.

D.2. Real-World Dataset

Experiment Details For experiments on forward-facing real-world scenes from LLFF [19] or captured by our own, we used the provided LLFF hyperparameters to train Plenoxels, whereas for the training of our method, we used a single-level set initialization $\tau_\sigma = \{10\}$. We found that this setting can produce cleaner and more smooth surfaces, as the training of Plenoxels was applied with a very strong TV regularization on the density field σ. As a result, the initialized surface field is much smoother and spread out. This increases the difficulty in converging the surfaces initialized with multiple level sets. For the additional experiments on semi-transparent objects ("mug" and "rough"), as surfaces initialized from Plenoxels are very incomplete, we increased the surface learning rate to 10^{-3} with the same delay schedule applied.

Additional Results We show qualitative results on additional scenes in Figure 10. It can be seen that both ours and Plenoxels can recover thin structures accurately, whereas NeuS reconstructed over smooth surfaces. Besides, Plenoxels and NeuS missed a more significant chunk of semi-transparent surfaces. In comparison, our method managed to recover more complete geometry. However, a part of the semi-transparent surfaces is still missing due to the strong ambiguity in the scene. More sophisticated regularization and additional priors might be required to fully recover those surfaces with minimal artifacts.

D.3. Ablation

We should 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

	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 3: **Chamfer distance ↓ on DTU scenes.** We color the best and second best surfaces.

close to 1 via mean squared error:

$$\mathcal{L}_{ek} = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{x} \in \mathcal{V}} ||\nabla \hat{\delta}(\mathbf{x}) - 1||^2 . \tag{36}$$

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.

D.4. DTU Dataset

We additionally show reconstruction results on some DTU [10] scenes in Figure 18 and Table 3. 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 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 18 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 11.

Experiment Details We compared with level set surfaces from Plenoxels [28] and NeuS [37] trained with masks. As our implementation does not contain a background model, we used the image masks provided by IDR [42] 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 for training our method on DTU scenes. Namely, we modified the surface scalar learning rate to start with 10^{-4} and end with 10^{-6}. We increased λ_δ to 0.05, $\lambda_\mathcal{H}$ to 10^{-3} and λ_α to 10^{-8}. We also kept the truncated alpha compositing parameter a at 5 throughout training. For this reason, when extracting surfaces for evaluation, we simply sample points from all the level sets.

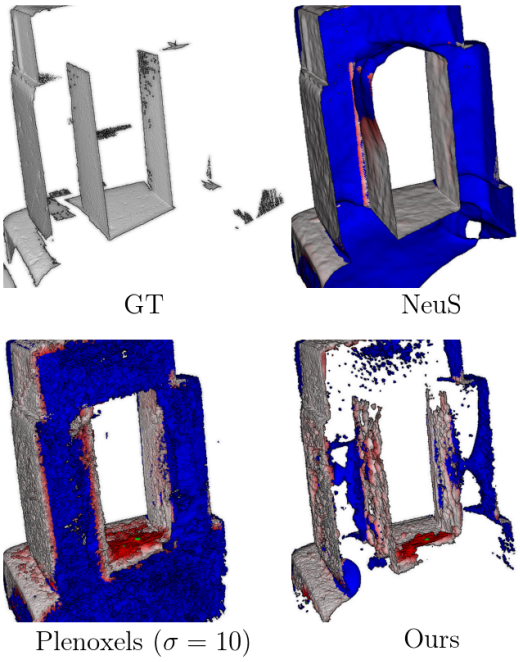


Figure 11: **Inside views of reconstructions on DTU dataset.** Red color indicates the L1 error in reconstruction, and blue indicates the reconstruction masked out by the DTU official masks. Surfaces extracted from Plenoxels contain many noisy inner surfaces that had to be masked out during evaluation to achieve low Chamfer distance.

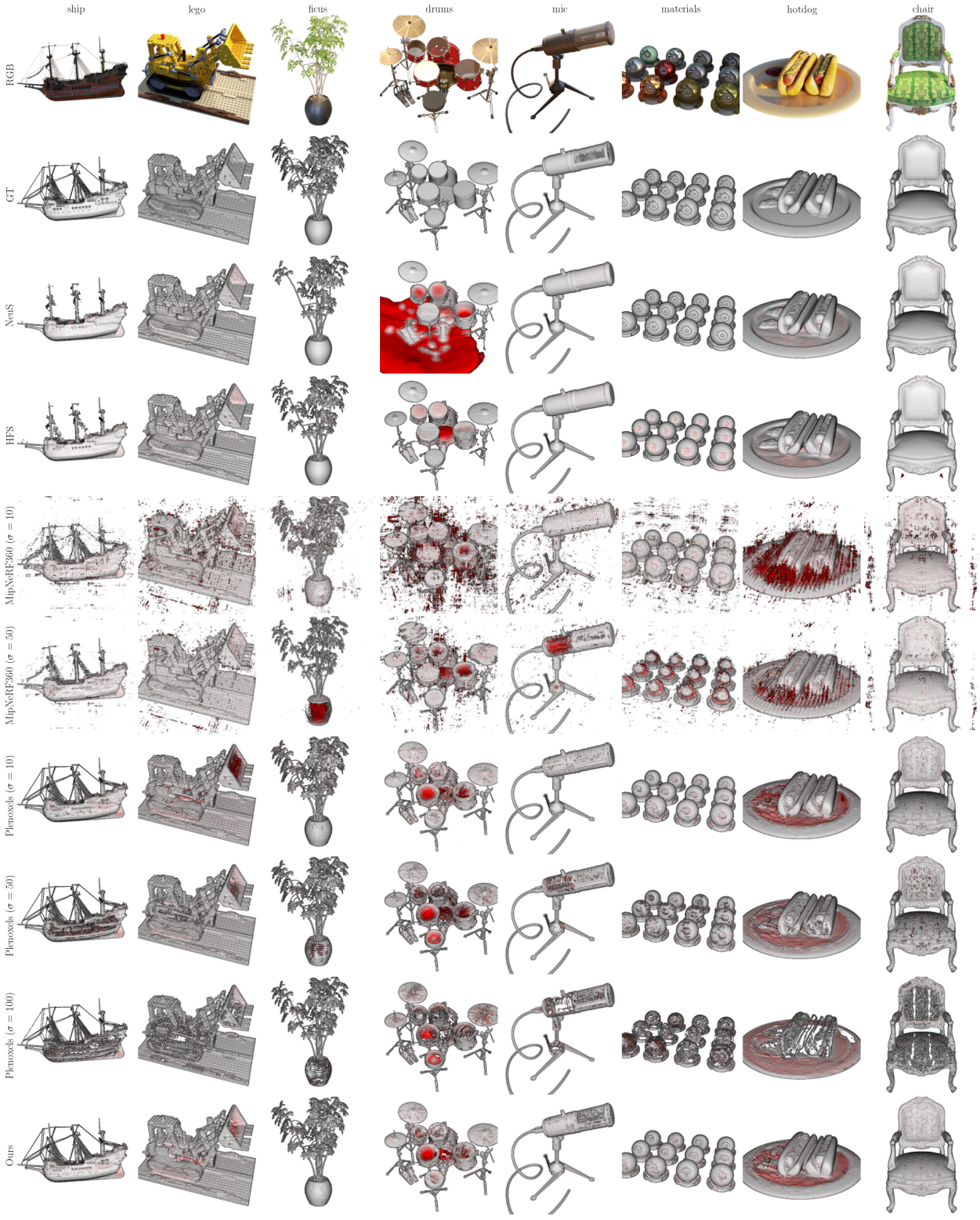


Figure 12: **Qualitative results of watertight surfaces on NeRF Synthetic dataset.**

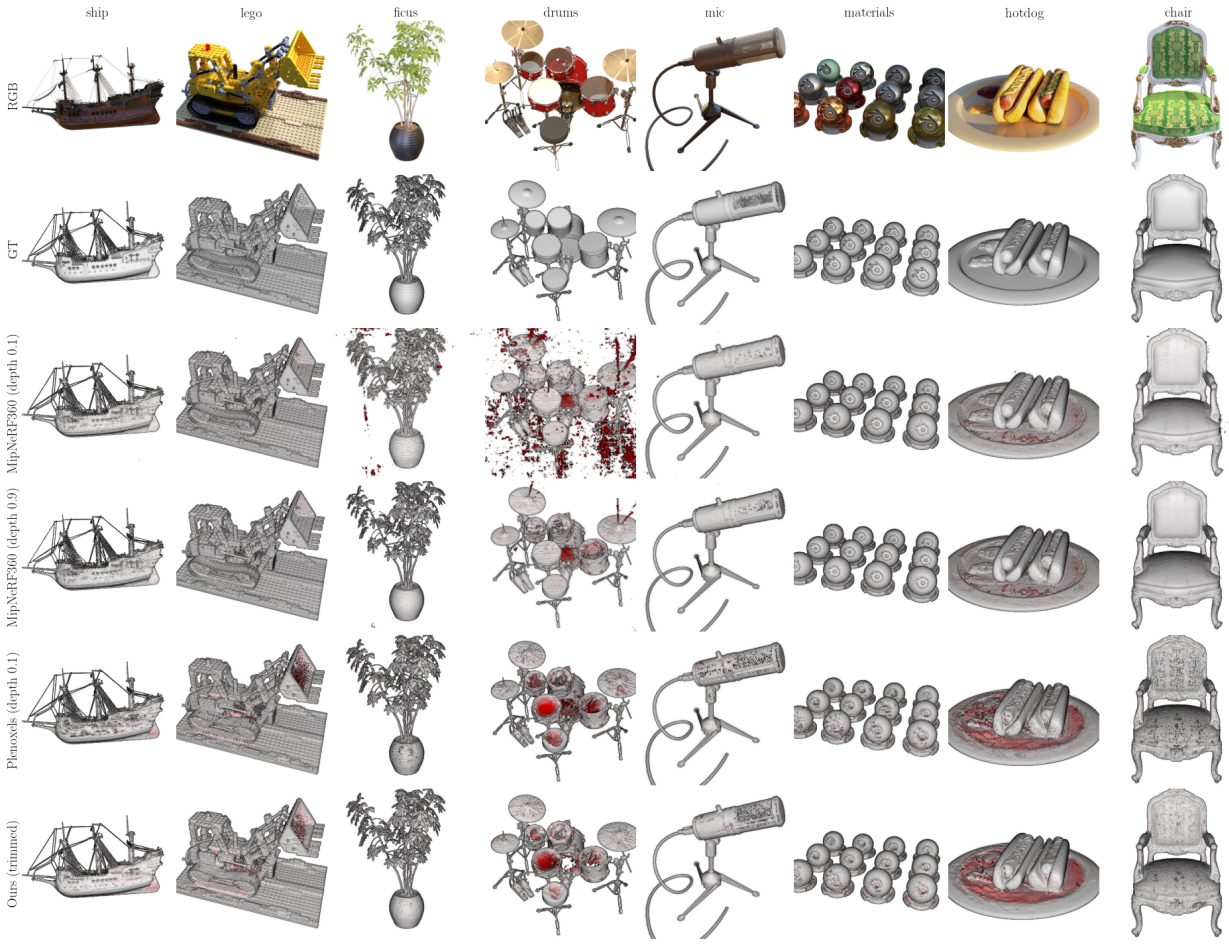


Figure 13: **Qualitative results of non-watertight surfaces on NeRF Synthetic dataset.**

	ship	lego	ficus	drums	mic	materials	hotdog	chair	avg
Plen ($\sigma = 10$)	0.985	0.726	1.131	1.763	1.012	0.945	1.141	1.991	1.212
Plen ($\sigma = 50$)	0.672	0.528	0.793	1.460	1.011	0.947	0.937	1.201	0.944
Plen ($\sigma = 100$)	0.479	0.703	0.417	1.048	0.808	1.025	2.259	1.077	0.977
Mip360 ($\sigma = 10$)	1.094	1.052	1.457	3.726	1.156	1.501	1.953	1.651	1.699
Mip360 ($\sigma = 50$)	1.217	0.704	2.640	2.301	1.420	2.474	1.463	2.224	1.805
Mip360 ($\sigma = 100$)	1.412	0.625	2.883	2.095	1.394	3.143	1.672	2.200	1.928
NeuS	0.552	0.619	1.667	10.273	0.251	0.210	0.602	0.210	1.798
HFS	0.514	0.463	0.374	1.820	0.274	0.255	0.272	0.428	0.550
Ours	0.365	0.514	0.389	1.037	0.310	0.369	0.833	0.456	0.534
Plen (depth 0.1)	0.391	0.452	0.366	1.118	0.591	0.469	0.798	0.537	0.590
Mip360 (depth 0.9)	0.305	0.237	0.293	0.858	0.315	0.425	0.496	0.239	0.396
Mip360 (depth 0.1)	0.300	0.236	1.716	12.553	0.305	0.413	0.611	0.241	2.047
Ours (trimmed)	0.277	0.376	0.240	0.617	0.288	0.317	0.702	0.349	0.396

Table 4: **Chamfer distance** $\downarrow \times 10^{-2}$ **on NeRF Synthetic datasets.** We color the best, second best watertight surfaces and the best non-watertight surfaces.

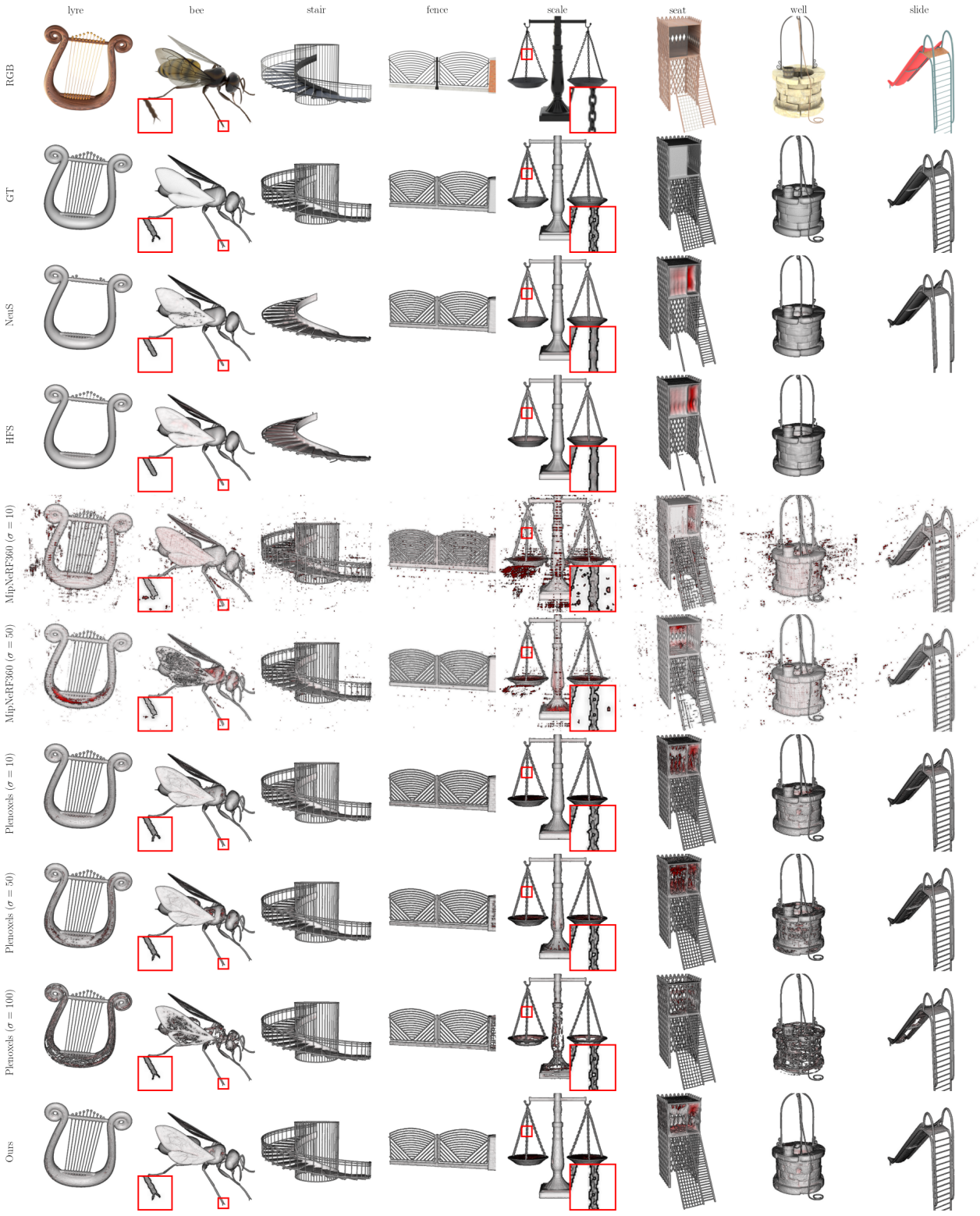


Figure 14: **Qualitative results of watertight surfaces on Thin Blender dataset.** Note that HFS [39] fails to learn any surface on "fence" and "slide" scenes.

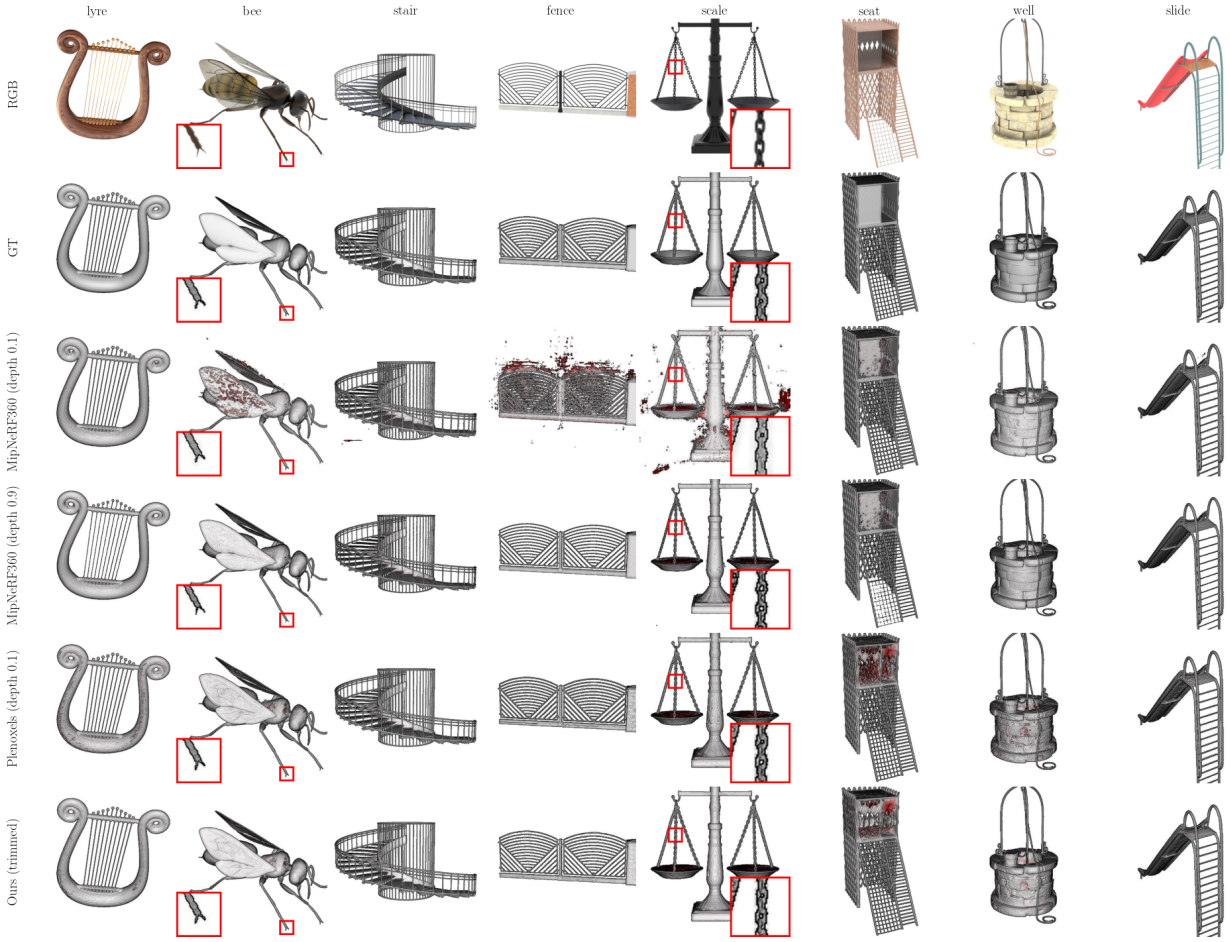


Figure 15: **Qualitative results of non-watertight surfaces on Thin Blender dataset.**

name	lyre	bee	stair	fence	scale	seat	well	slide	avg
Plen ($\sigma = 10$)	0.740	0.422	0.265	0.205	0.793	1.128	0.608	0.599	0.595
Plen ($\sigma = 50$)	0.805	0.593	0.208	0.328	0.959	0.760	0.708	0.509	0.609
Plen ($\sigma = 100$)	0.543	0.598	0.212	0.305	0.884	0.492	0.703	0.648	0.548
Mip360 ($\sigma = 10$)	0.913	0.803	0.527	0.864	6.221	1.286	2.754	0.999	1.796
Mip360 ($\sigma = 50$)	1.311	1.181	0.279	0.433	2.243	0.744	1.941	0.597	1.091
Mip360 ($\sigma = 100$)	2.613	2.252	0.268	0.445	2.024	0.541	1.547	0.526	1.277
NeuS	0.812	0.242	4.087	0.141	0.237	0.431	0.360	0.358	0.834
HFS	0.781	0.336	4.316	-	0.271	0.425	0.385	-	1.086
Ours	0.182	0.229	0.184	0.155	0.641	0.536	0.442	0.558	0.366
Plen (depth 0.1)	0.416	0.351	0.215	0.263	0.788	0.495	0.500	0.498	0.441
Mip360 (depth 0.9)	0.192	0.361	0.200	0.139	0.741	0.270	0.210	0.448	0.320
Mip360 (depth 0.1)	0.180	0.456	0.244	0.523	2.401	0.259	0.214	0.426	0.588
Ours (trimmed)	0.188	0.207	0.176	0.176	0.575	0.288	0.319	0.506	0.304

Table 5: **Chamfer distance** $\downarrow \times 10^{-2}$ **on Thin Blender datasets.** We color the best, second best watertight surfaces and the best non-watertight surfaces. Note that HFS [39] fails to learn any surface on "fence" and "slide" scenes.

Figure 16: **Qualitative results of watertight surfaces on Semi-Transparent Blender dataset.**

Figure 17: **Qualitative results of non-watertight surfaces on Semi-Transparent Blender dataset.** All methods failed to reconstruct complete surfaces for "jar" as it has a very similar color to the background.

name	glass table	jar	glasses	wine	lego t	bottle	cup	case	avg
Plen ($\sigma = 10$)	1.355	3.031	1.215	1.598	0.637	0.717	0.886	1.255	1.337
Plen ($\sigma = 50$)	0.989	2.943	0.871	2.050	0.628	0.929	1.141	1.200	1.344
Plen ($\sigma = 100$)	0.719	2.817	2.009	3.814	1.104	1.870	1.690	1.895	1.990
Mip360 ($\sigma = 10$)	12.123	9.770	0.906	2.685	7.636	2.437	1.801	1.846	4.900
Mip360 ($\sigma = 50$)	4.201	14.763	0.595	6.882	6.913	2.324	1.669	4.012	5.170
Mip360 ($\sigma = 100$)	3.181	19.943	0.799	8.065	5.988	2.915	3.188	4.703	6.098
NeuS White	3.877	5.032	0.436	0.240	0.685	2.271	0.872	5.091	2.313
HFS White	2.586	4.615	0.320	0.527	0.598	1.493	1.961	4.621	2.090
Ours	0.820	1.945	0.874	0.351	0.534	0.688	0.811	0.721	0.843
Plen (depth 0.1)	0.809	2.742	0.694	0.702	0.492	0.631	0.956	0.768	0.974
Mip360 (depth 0.9)	4.070	2.383	0.215	1.325	0.387	1.323	0.792	0.816	1.414
Mip360 (depth 0.1)	5.374	4.932	0.219	1.763	0.806	3.101	1.959	0.611	2.346
Ours (trimmed)	0.670	1.560	0.548	0.411	0.427	0.604	0.711	0.510	0.680

Table 6: **Chamfer distance** $\downarrow \times 10^{-2}$ **on Semi-Transparent Blender datasets.** We color the best, second best watertight surfaces and the best non-watertight surfaces.

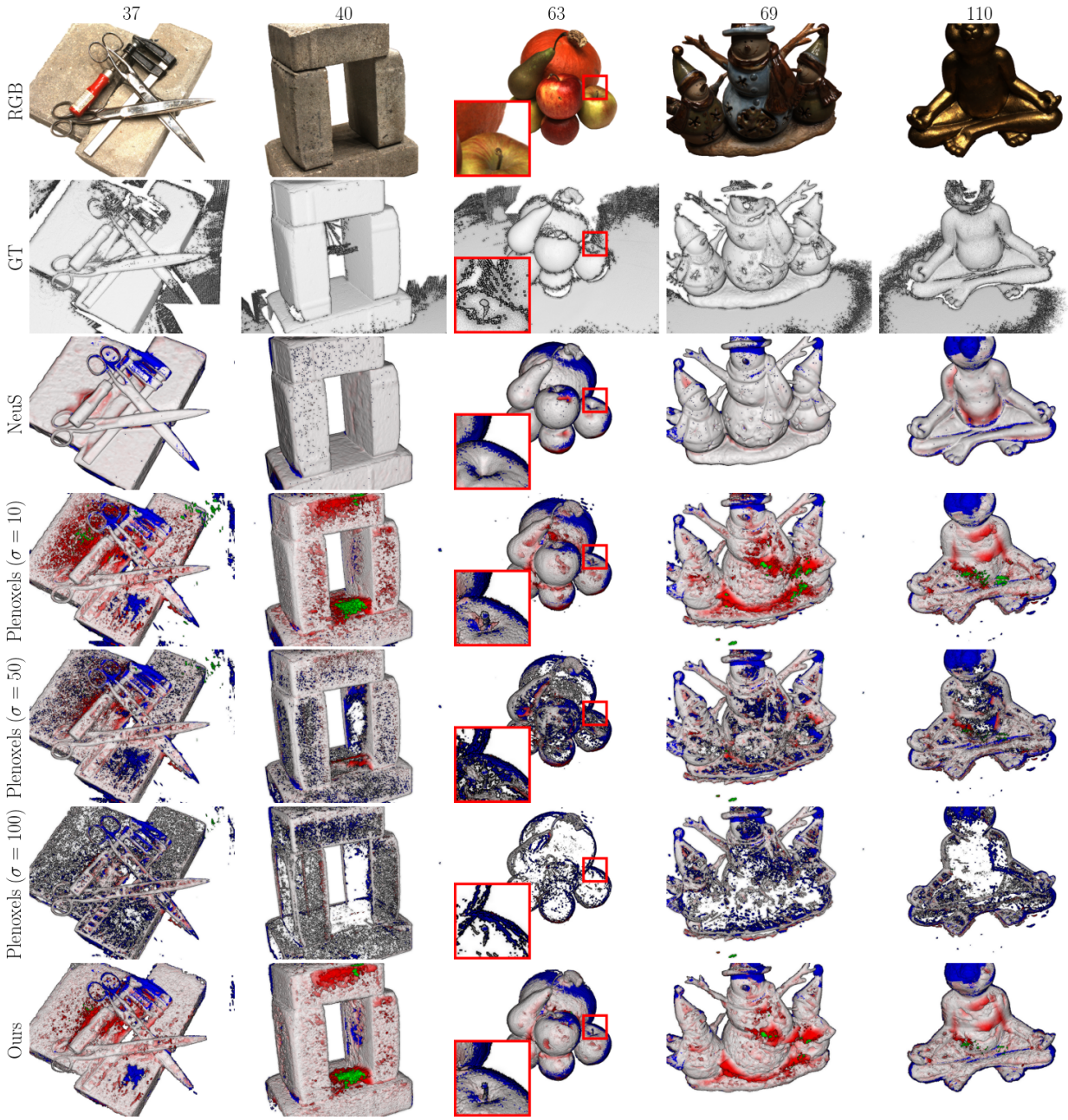


Figure 18: **Qualitative results on DTU [10] dataset.** As the DTU scenes mainly contain smooth surfaces without any semi-transparent materials, our method does achieve state-of-the-art performance on this dataset. However, note that our method can still accurately capture the thin structure that is 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.