

# $\alpha$ Surf: 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  $v_{\mathbf{x}}$ :

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

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

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 18:

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

By re-arranging the equation, we obtain:

$$\tau_i = f_3 t'^3 + f_2 t'^2 + f_1 t' + f_0 \quad (20)$$

where

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

and

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

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:

$$\tau_i = t'^3 + at'^2 + bt' + c \quad (23)$$

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

Then, compute:

$$Q = \frac{a^2 - 3b}{9} \quad (25)$$

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

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

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

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

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

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

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:

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

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

Then, the only real root can be obtained as:

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

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  $v_x$  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.

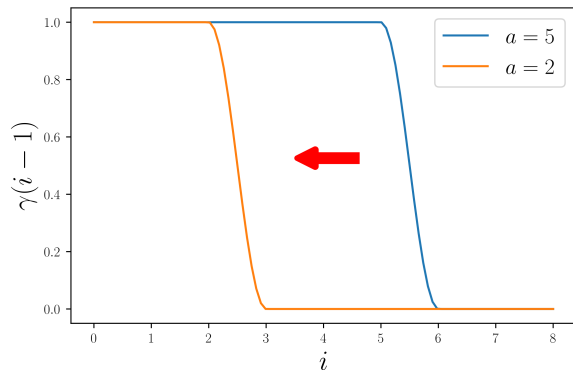


Figure 7. **The truncated alpha compositing reweight function**  $\gamma(i-1)$ . The x-axis is the index of the intersection starting from 1 (excluding intersections on back-facing surfaces). By reducing  $a$ , we effectively slide the curve to the left.

## 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  $\tau_\sigma = \{10, 30, 50, 70, 90\}$ , 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  $\lambda_c = 10^{-6}$

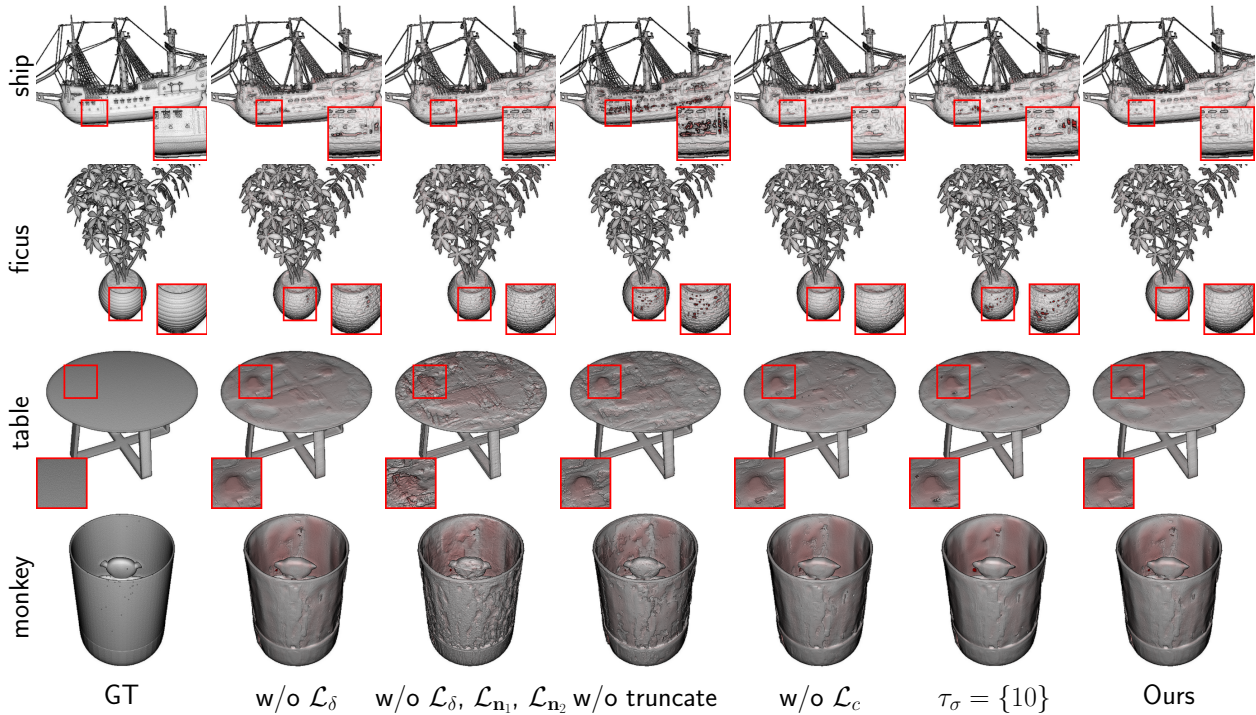


Figure 8. **Ablation Study.** We show the qualitative results of different ablations. Our full approach achieves the best quality overall.

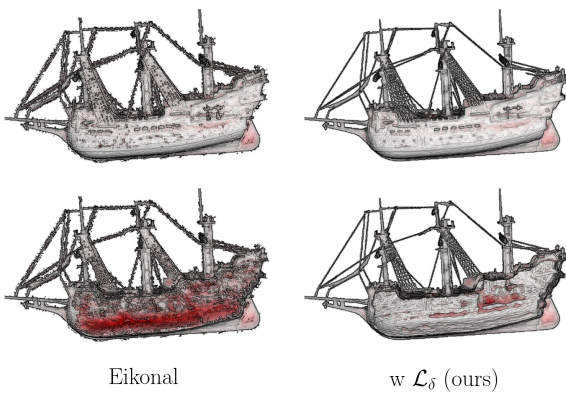


Figure 9. **Comparison with the Eikonal constraint regularization.** We visualize the out view (first column) and the inside view (second column) by cropping the surfaces along the y-axis. It can be clearly seen that the Eikonal constraint does not regularize the surface to be clean and smooth, but rather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

for the first  $10k$  iterations and 0 for the rest of training. We use  $\lambda_\delta = 10^{-3}$ ,  $\lambda_{\mathcal{H}} = 10^{-4}$ ,  $\lambda_\alpha = 10^{-9}$ ,  $\lambda_{n_1} = 10^{-6}$  and  $\lambda_{n_2} = 0$  for the Thin dataset.  $\lambda_{n_2}$  was disabled as it tends to destroy the thin structures with rapid normal variations. For the Translucent dataset, we use  $\lambda_\delta = 10^{-5}$ ,  $\lambda_\alpha = 10^{-11}$ ,

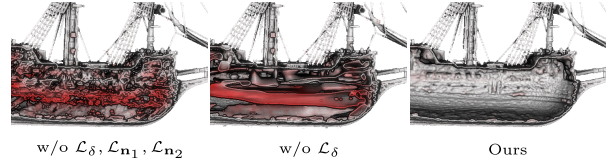


Figure 10. Reconstruction without regularization tends to give an enormous amount of redundant inner surfaces. Encouraging consistent normals in the local neighborhood via  $\mathcal{L}_{n_1}, \mathcal{L}_{n_2}$  enforces smoother surfaces, but redundant surfaces still remain. With both normal regularization and TV loss applied on the surface scalar field, we can obtain clean and smooth reconstruction.

$\lambda_{\mathcal{H}} = 10^{-4}$ ,  $\lambda_{n_2} = 10^{-4}$ , and linearly decay  $\lambda_{n_1}$  from  $10^{-2}$  to  $10^{-4}$ .

**Real-World** For experiments on real-world scenes, we initialize with less level sets  $\tau_\sigma = \{10, 30, 50\}$ , 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  $\lambda_H$  as synthetic experiments, and set  $\lambda_\delta = 5 \times 10^{-3}$ ,  $\lambda_\alpha = 10^{-9}$ ,  $\lambda_{n_2} = 10^{-3}$  and linearly decay  $\lambda_{n_1}$  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]:

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

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

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.

**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.





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 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 close to 1 via mean squared error:

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

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  $\mathcal{L}_{n_1}, \mathcal{L}_{n_2}$  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

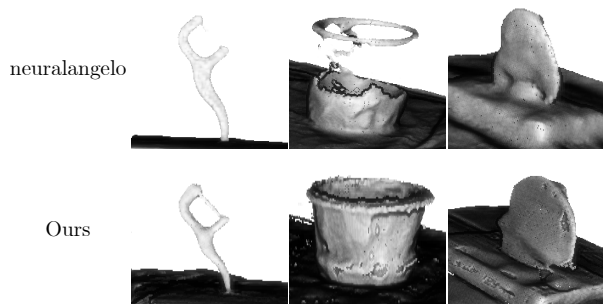


Figure 12. **Additional real-world comparisons with neuralangelo [28]** Note that neuralangelo uses a different coordinate system and camera processing pipeline for COLMAP scenes, therefore the reconstructions are not perfectly aligned, but it can still be clearly seen that our method achieves better 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  $\downarrow$  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 out-growing 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.

**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 train-

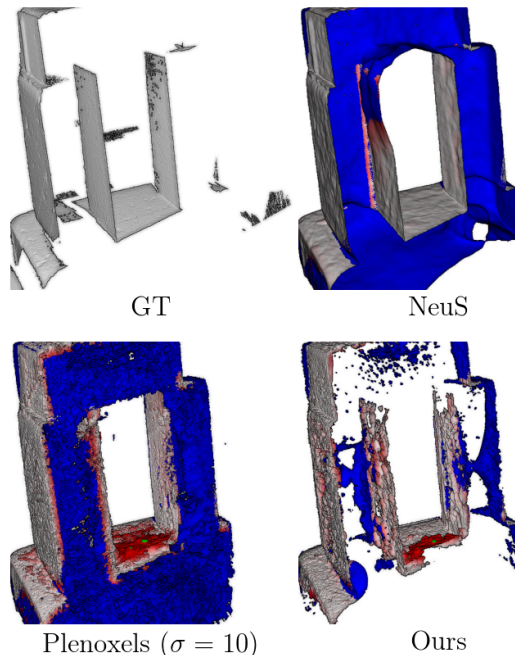


Figure 13. **Inside views of reconstructions on DTU dataset.** Red color indicates the L1 error in reconstruction, and blue indicates the reconstruction masked out by the DTU official masks. Surfaces extracted from Plenoxels contain many noisy inner surfaces that had to be masked out during evaluation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

ing 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,  $\lambda_H$  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

17

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



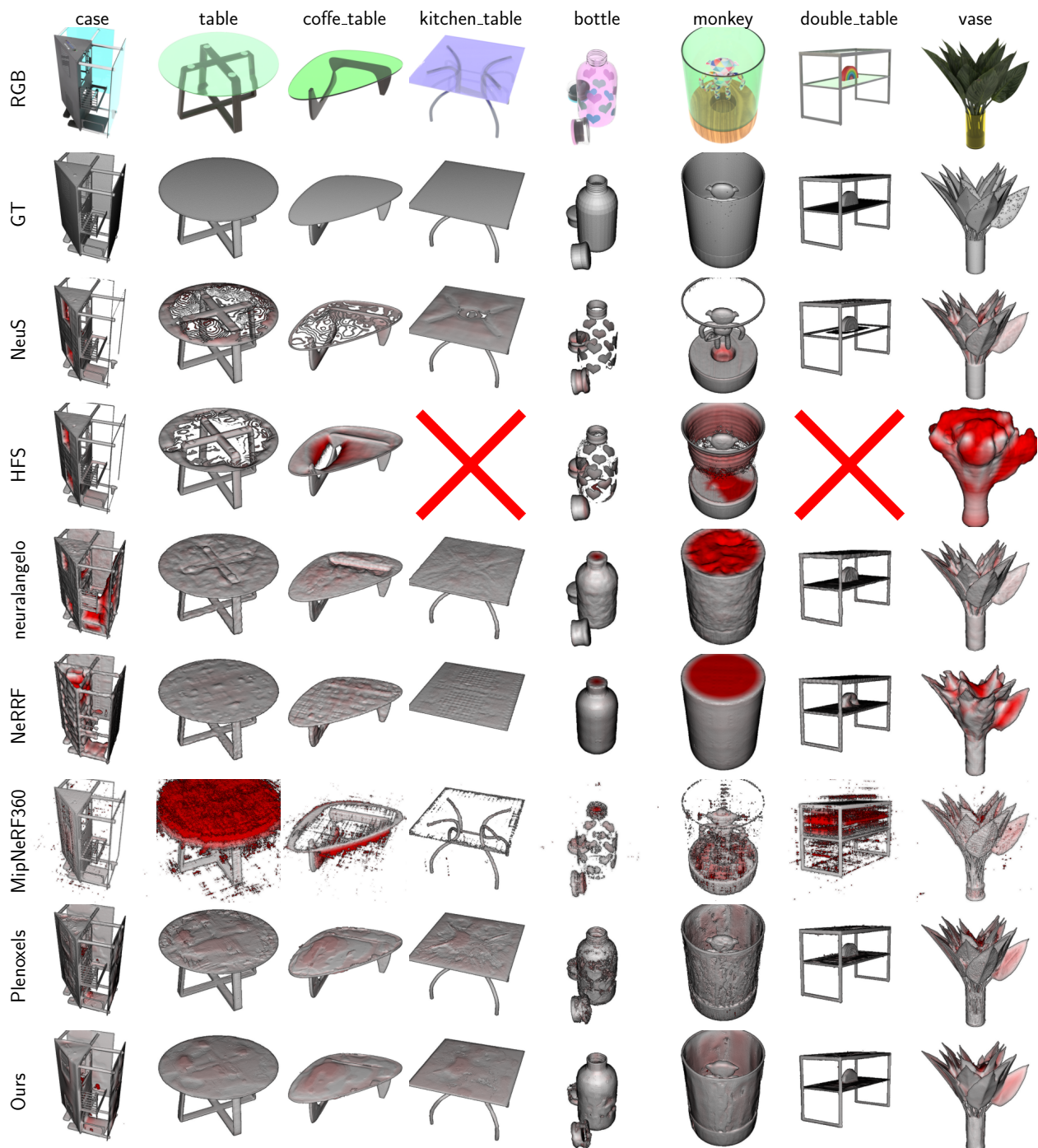


Figure 15. **Qualitative results on Translucent Blender dataset.** Note that HFS [56] fails to learn any surface on “kitchen table” and “double table” scenes. We removed some exterior surfaces in the “monkey” scene to show the interior surfaces.

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	0.555	0.653	0.776	0.512	0.870	0.624

Table 6. **Chamfer distance  $\downarrow \times 10^{-2}$  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.



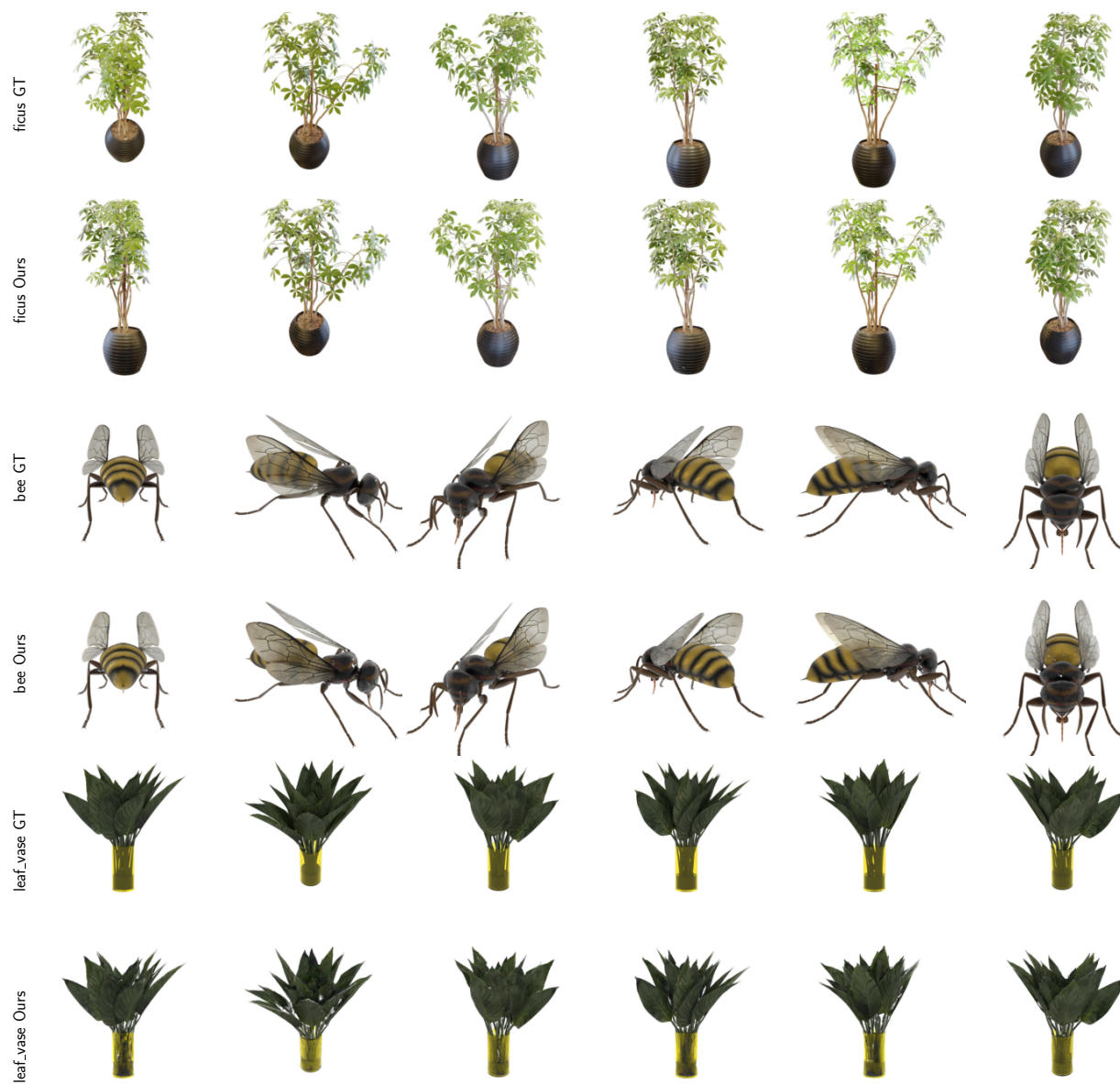


Figure 16. RGB renderings of our methods on synthetic datasets.

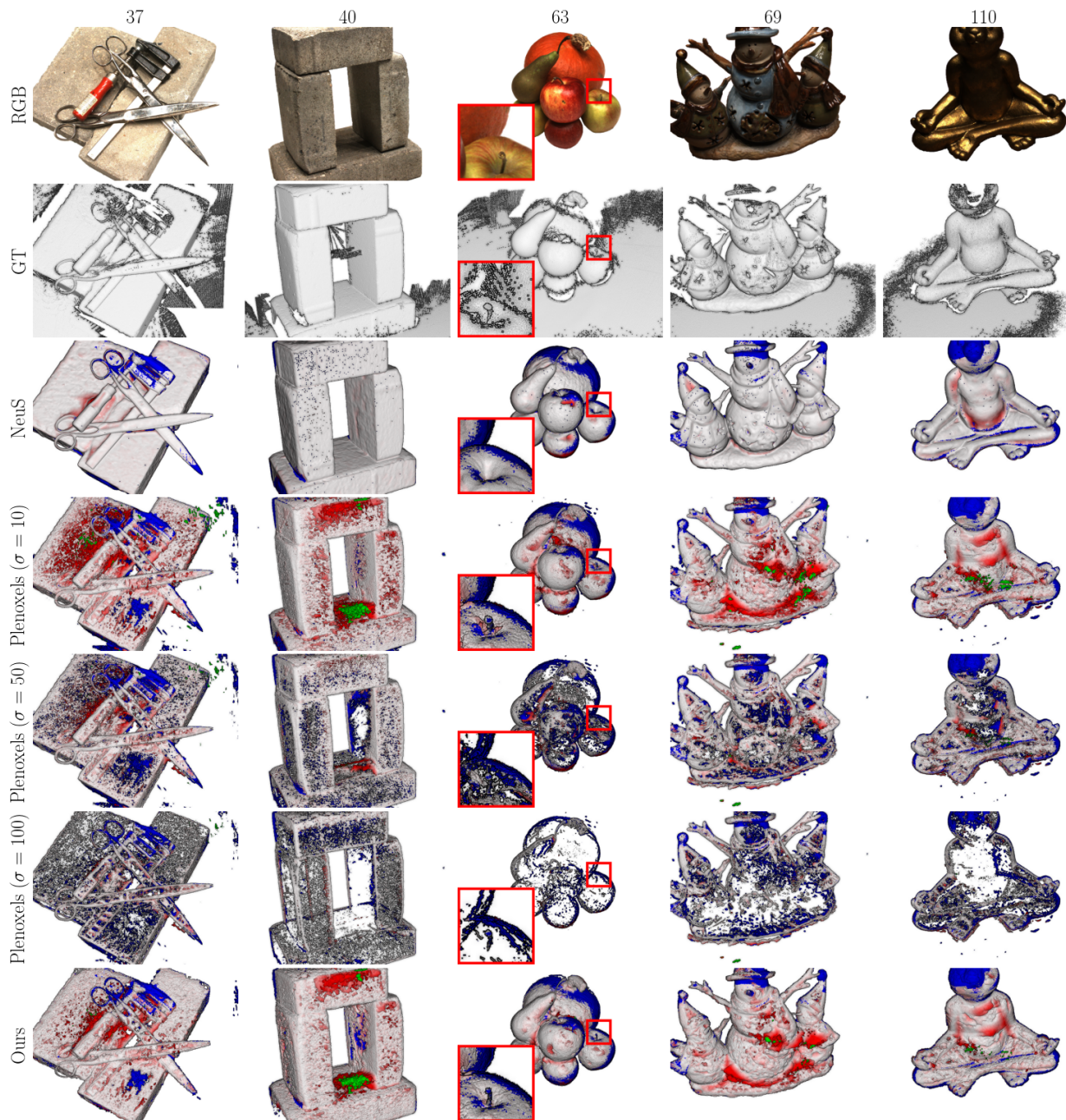


Figure 17. **Qualitative results on DTU [17] dataset.** As the DTU scenes mainly contain smooth surfaces without any semi-transparent materials, our method does achieve state-of-the-art performance on this dataset. However, note that our method can still accurately capture the thin structure that is 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.