A. Additional Training Details

A.1. Misc

We downsample the original 2048×2048 images to 400×400 for all methods but COLMAP. Both NeRF baselines are trained for 100k iterations and take ~ 10 hours to run on a 32GB NVIDIA V100. DS is trained for 50k iterations and takes ~ 9 hours to run on the same GPU.

A.2. Bi-directional distance transform loss

This section describes the bi-directional distance transform loss $L_{\text{bi-dt}}$ used in the mask reconstruction loss in Equation 6 of the main text. Our novel loss is a differentiable version of a non-differentiable naive bi-directional distance transform loss. We first define the naive non-differentiable loss and then describe our differentiable adaptation to it. For convenience, we misuse A (or A^r) to refer to the set of pixels where GT mask A (or the rendered mask A^r) is 1.

A naive non-differentiable bi-directional distance transform loss $L_{\text{bi-dt-naive}}(A^r, A)$ between the rendered mask A^r and GT mask A consists of 2 components. The first penalizes pixels $p \in A^r \setminus A$, where the rendered mask is 1 but GT mask is 0, by the distance of p from $\texttt{NN}(p, A)$ - the closest occupied pixel in the GT mask. $\texttt{NN}$ is the nearest neighbor operation in euclidean space. The second component penalizes pixels $q \in A \setminus A^r$, where the GT mask is 1 but rendered mask is 0, by the distance of q from $\texttt{NN}(q, A^r)$ - the closest occupied pixel in the rendered mask.

$$L_{\text{bi-dt-naive}}(A^r, A) = \sum_{p \in A^r \setminus A} \texttt{NN}_d(p, A) + \sum_{q \in A \setminus A^r} \texttt{NN}_d(q, A^r)) \tag{7}$$

where $\texttt{NN}_d(p, A) = d(p, \texttt{NN}(p, A))$ is the distance between p and its nearest neighbor in A as measured by a metric d.

The operation of finding the set of pixel locations where the rendered mask A^r is occupied is the only non-differentiable one in the loss above. Our novel adaptation provides a differentiable approximation for the same. Given a pixel p in i-th image I_i, recall (from Section 3.1) that the rasterizer finds K points $x_{1..K}$ on the surface of the mesh that project to p under camera π_i. Therefore, $\pi_i(x_k)$ is a differentiable approximation to p for each x_k. We approximate $p \sim \hat{p} = \sum_k w_k \pi_i(x_k)$ as a normalized weighted sum of projections $\pi_i(x_k)$ with weights from the softmax blending that composites texels $c_{1..K}$ into a final color at p. Let $\hat{A^r} = \{\hat{p} | p \in A^r; \hat{p} = \sum_k w_k \pi_i(x_k)\}$. The full differentiable bi-directional distance transform loss is

$$L_{\text{bi-dt}}(A^r, A) = \sum_{p \in A^r \setminus A} \texttt{NN}_d(\hat{p}, A) + \sum_{q \in A \setminus A^r} \texttt{NN}_d(q, \hat{A^r})) \tag{8}$$

where $d(x, y) = \texttt{clamp}(||x - y||_2, \tau_{\text{min}}, \tau_{\text{max}})$ is the clamped L-2 euclidean distance. We set τ_{min} to 2 pixels and τ_{max} to 0.1 of the shorter image dimension.

A.3. Learning Schedule

We run our optimization for 50k iterations using SGD with a momentum of 0.9 and an initial learning rate of 0.01. We use cosine annealing for our learning rate with warm restarts [33] which we find helps avoid local minima for both shape and cameras. We clip gradient norms for stability. For mesh rasterization, we use PyTorch3D [41] with $K = 6$ and a blur radius that decays exponentially from 5×10^{-5} to 10^{-6} over the course of optimization.

A.4. Aligning meshes for benchmarking

As cameras are optimized, ground-truth and optimized shapes are not in a common coordinate space and must be aligned before benchmarking. However, different benchmarking metrics are minimized by different alignments. Therefore, for every instance, for each metric, we find 3 possible alignments and pick the best. The first is a brute force minimization of Chamfer-L2 over scale/depth in camera 0's view coordinate space. The second and third additionally optimize rotation and translation via ICP from ground-truth to predicted mesh and vice-versa.

B. Additional Results

B.1. COLMAP

In Fig. 14, we show dense pointcloud reconstructions by COLMAP on scenes from the Tanks and Temples dataset as the number of views reduces from 100 to 50 to 25 to 15. The pointcloud density drops drastically as we reduce the number of views below 50. With 15 views, the reconstruction is practically empty and the scene is unrecognizable. Our attempts to mesh these points using COLMAP's Poisson and Delaunay meshers failed. In contrast, as seen in Fig. 8 of the main text, our approach is far more robust with the same views.

In Fig. 15, we show dense pointcloud reconstructions by COLMAP on instances from the Google dataset with 8 input views and ground-truth cameras. COLMAP fails on 2/50 instances and of the 48 instances it works on, it only reconstructs parts of the scene that are textured and visible in close (narrow-baseline) views. It still misses surfaces that are visible in only 2-3 wide baseline views. For example in Fig. 15, COLMAP fails to reconstruct the back of the green backpack which is visible only in 2/8 input views.

Cameras from SfM We tried using COLMAP's Structure-from-Motion (SfM) pipeline to ameliorate the need for noisy camera inputs. It often, if not always, fails to register all cameras into a single coordinate space correctly and usually ends

up with multiple groups – each with 2-3 cameras with reasonably accurate relative orientation. The inability to merge the multiple groups into a single coordinate frame made it infeasible to use the COLMAP cameras as initializations. Note that SfM's failure is not surprising since we are working with 8 white-background views covering 360-degrees of the object. On average, any surface is visible from ~ 3 views only. Furthermore, white background images are harder to calibrate than in-context images with background because of the lack of visual overlap in the background.

B.2. Further comparisons

In addition to comparisons with NeRF [35] and NeRF-opt in the main paper, we compare to COLMAP [43], IDR [51], and DS-naive. Tab. 2 shows results on GSO following the evaluation protocol of the main paper. All methods use 8 views and $\sigma = 30°$ camera noise, except COLMAP, which uses ground-truth cameras. For COLMAP, we compute metrics on the reconstructed dense point cloud via uniform random sampling.

DS outperforms all baselines across all metrics. IDR gets slightly better shapes but worse cameras than NeRF-opt. However, neither can recover accurate geometry under camera noise. Surprisingly, DS-naive performs similar to NeRF-opt but with better Chamfer and Normal Consistency, suggesting the mesh representation is robust in noisy settings. We note that COLMAP has extremely high precision (99.8%, compared to DS's 88.4%) but very poor recall (35.5%, compared to DS's 78.2%) at a threshold of 0.2. This suggests that COLMAP reconstructs accurate point clouds but misses large parts of the shape, supporting the qualitative claim in Appendix B.1. The comparison with COLMAP, a method that finds view correspondences followed by triangulation, suggests that such methods have a hard time reconstructing shapes from limited views.

Tanks and Temples Fig. 10 compares on Tanks and Temples with 15 views and no camera noise (to be compared to Fig. 8 in the main paper). Tab. 1 shows the corresponding quantitative comparison following the evaluation protocol of the main paper. However, since the ground-truth pointclouds are hollow (without the bottom), the reported numbers only approximate the quality of the shape. NeRF-opt produces lower quality shapes than DS. With more views and no camera noise, IDR performs slightly better than DS. This is not a surprise. Our approach has an advantage when few views (≤ 12) are available and cameras are noisy. But our approach, even with more views ($= 15$) and no camera noise, reconstructs shapes well, performing better than NeRF-opt and similar to IDR.

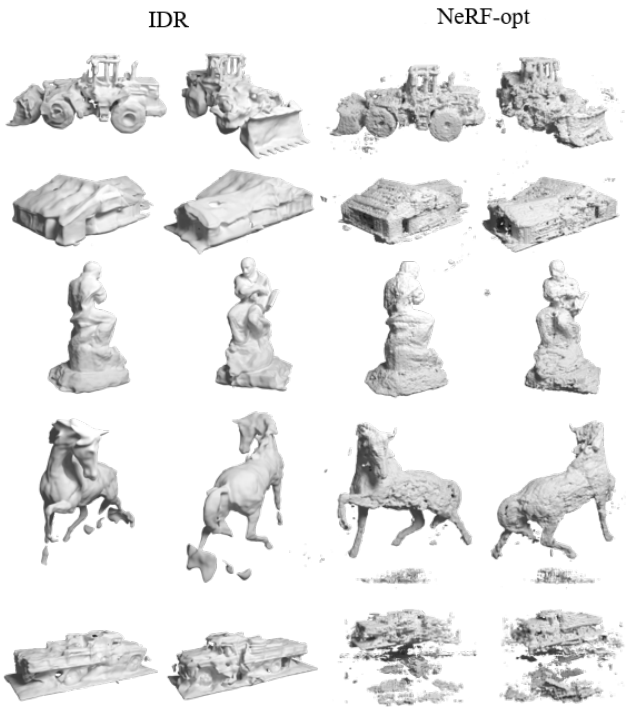

Figure 10. IDR and NeRF-opt on Tanks and Temples with 15 views.

Scene	Method	Chamfer↓	F1@.1↑	F1@.2↑
Barn	IDR	**0.14**	**47.6**	**70.2**
	NeRF-opt	0.98	33.4	48.2
	DS	0.39	35.1	. 56.0
Caterpillar	IDR	**0.06**	**59.6**	**81.9**
	NeRF-opt	0.15	54.5	77.7
	DS	0.07	55.8	76.4
Ignatius	IDR	**0.18**	**68.2**	**86.9**
	NeRF-opt	0.20	54.6	74.5
	DS	0.30	53.4	75.8
Truck	IDR	**0.06**	**54.8**	**79.7**
	NeRF-opt	1.40	23.9	40.8
	DS	0.14	52.0	70.7

Table 1. Results on Tanks and Temples scenes with 15 input views.

Method	Chamfer↓	F1@.1↑	F1@.2↑	NC↑	Rot↓
COLMAP	0.38	35.2	52.3	-	-
IDR	0.52	22.0	40.5	0.23	22.8
NeRF-opt	0.31	33.1	58.6	0.28	13.4
DS-naive	0.22	29.5	53.0	0.54	14.7
DS	**0.10**	**63.5**	**80.6**	**0.68**	**0.54**

Table 2. Results on GSO with 8 views and $\sigma = 30°$ camera noise. NC=Normal Consistency; Rot=Rotation Error (in degrees).

DTU dataset Fig. 11 compares on scenes from the DTU dataset with 6 views and linear camera noise following IDR [51]. Note that this data is easier to reconstruct because the 6 views span only $1/8^{\text{th}}$ of the entire viewing sphere and the cameras have an average camera noise of only $\sim 1°$, which is much lesser than most of our experiments on GSO. We observe that both DS and IDR outperform NeRF-opt, which has cloudy artifacts. IDR works well in this setting with few views and almost-correct cameras. This is not a surprise as DS has an advantage with few views and noisy cameras. IDR reconstructions are slightly more detailed than DS at some places (*e.g.* at the windows of the house and the feet of the bird). However, IDR shapes are less constrained and contain erroneous blobs (*e.g.* at the belly of the teddy bear, the middle and side of the house, and the head of the boy) because of the lack of a sufficient number of views.

B.3. DS

In Fig. 12 , we show shape and texture reconstruction from DS on more instances in the GSO Dataset.

In Fig. 13, we show some failure modes of DS. Thin structures and small objects are particularly hard to reconstruct. This is because we use a mesh representation with surface smoothening regularization. DS does not model surface specularities and lighting and subsequently also struggles with specular surfaces.

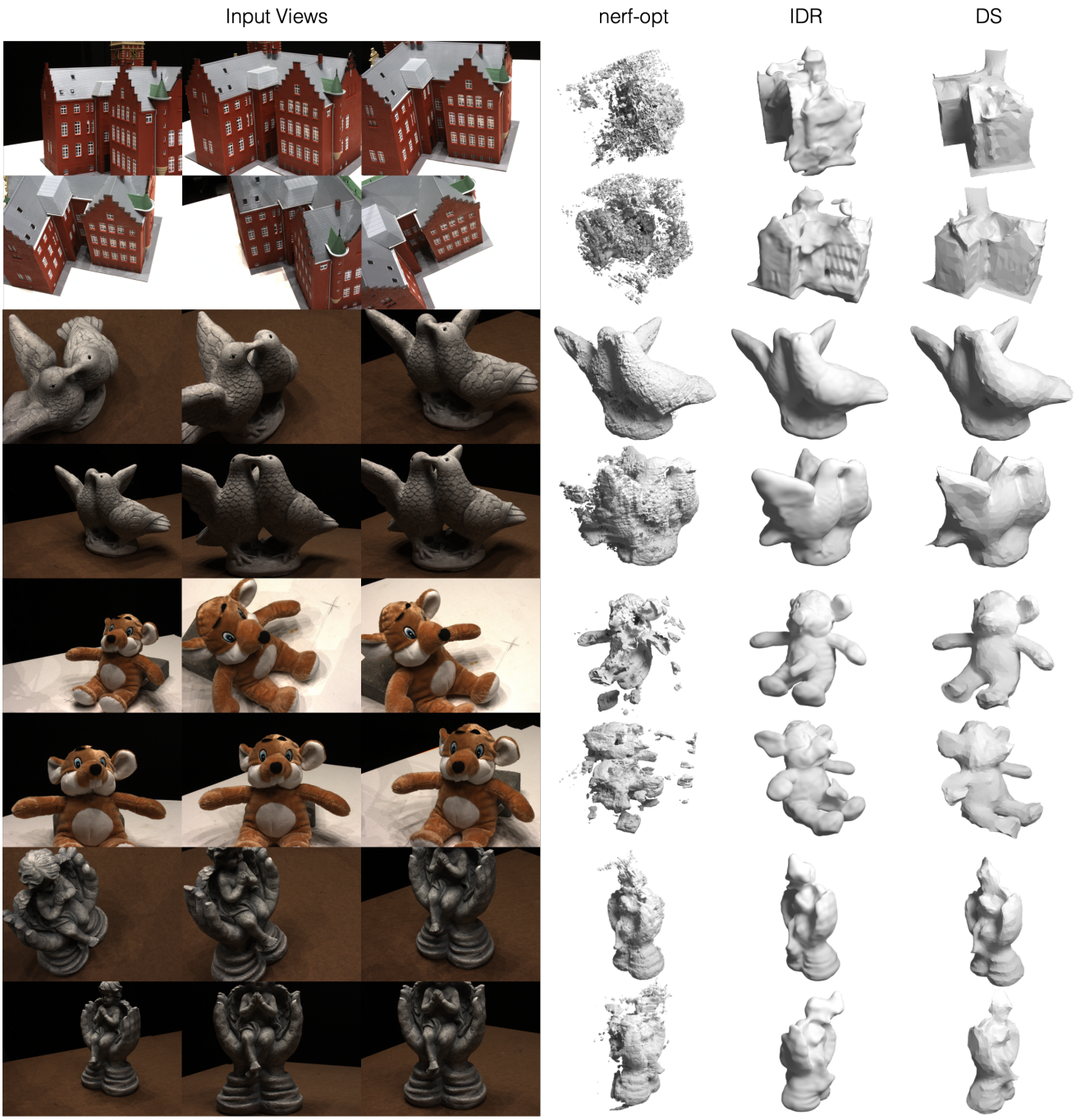

Figure 11. Results on DTU with 6 views and linear camera noise. Figure compares DS to IDR and NeRF-opt.

Figure 12. Qualitative shape and texture reconstructions by DS using 8 views and 20° camera noise for more instances from Google's Scanned Objects Dataset

Figure 13. Failure modes for two instances from Google's Scanned Objects. For the leftmost example, we show the input views (left) and reconstructions (middle) and ground truth shape (right). For the rightmost example, we show input views (left), reconstructions (right top) and ground truth shape (right bottom).

Figure 14. COLMAP-reconstructed dense pointclouds with varying number of input views from scenes in the Tanks and Temples dataset.

Figure 15. COLMAP-reconstructed dense pointclouds with 8 input views and ground-truth cameras for objects in the Google Scanned Objects Dataset.