Supplementary Material for Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision

Michael Niemeyer[1,2]   Lars Mescheder[1,2,3†]   Michael Oechsle[1,2,4]   Andreas Geiger[1,2]
[1]Max Planck Institute for Intelligent Systems, Tübingen   [2]University of Tübingen
[3]Amazon, Tübingen   [4]ETAS GmbH, Bosch Group, Stuttgart
{firstname.lastname}@tue.mpg.de

Abstract

In this ***supplementary document****, we first discuss our network architecture, implementation details, the metrics we used and the detailed training protocol in Section 1. More information regarding data pre- and post-processing steps can be found in Section 2. Finally, we provide additional experimental results, both qualitatively and quantitatively in Section 3.*

1. Implementation

In this section, we provide additional implementation details. We describe our network architecture, details regarding the forward and backward pass in our model, the reported metrics, our training protocol, the normal loss, and the loss weighting.

1.1. Architecture

In contrast to previous works [14, 15], We implement the occupancy network f_θ and texture field $\mathbf{t}_\theta$ in a *single network* with shared weights and two shallow heads as illustrated in Fig. 1. The network takes as input a batch of N_p points and (optionally) an input image $\mathbf{x}$. It outputs N_p one-dimensional occupancy probabilities and N_p three-dimensional RGB color values. We pass the point coordinates (p_1, p_2, p_3) through a fully connected layer with ReLU [7] activation. This output is then passed through five ResNet [8] blocks each with ReLU activation and a hidden dimension of 512. For our 2D supervised model we use a hidden dimension of 128. If we condition our network on input $\mathbf{x}$, we use an encoder network $\mathbf{g}_\phi$ to encode the input into a latent code $\mathbf{z} = \mathbf{g}_\phi(\mathbf{x})$. We pass this latent code $\mathbf{z}$ through a fully connected layer and add it before every ResNet block to the output of the previous block. The output of the final ResNet block is then passed through one fully connected layer with output dimension 1 for the occupancy probabilities and one fully connected layer with output dimension 3 for the RGB color values.

1.2. Forward Pass

During training, we randomly sample an image $\mathbf{I}_k$ with camera origin $\mathbf{r}_0$ and N_p points on the image plane denoted by $\mathbf{u}_i$ (see Sec. 3.4 of the main publication). For clarity, let's consider a single image pixel $\mathbf{u}$ in the following. Remember that the ray $\mathbf{r}$ cast from $\mathbf{r}_0$ towards $\mathbf{u}$ is defined as

$$\mathbf{r}(d) = \mathbf{r}_0 + d\mathbf{w} \tag{1}$$

where $\mathbf{w}$ denotes the vector connecting $\mathbf{r}_0$ and $\mathbf{u}$ (see Sec. 3.2 of the main publication). To efficiently perform the forward pass, we do not want to evaluate occupancy values along the whole ray but only in the volume of interest. In our implementation, we operate on a unit cube which is aligned with the visual hull of the target object. From this cube, we calculate (per pixel) two scalar values s_0 and s_n which are the minimum and maximum depth values to cover the volume of interest (note

†This work was done prior to joining Amazon.

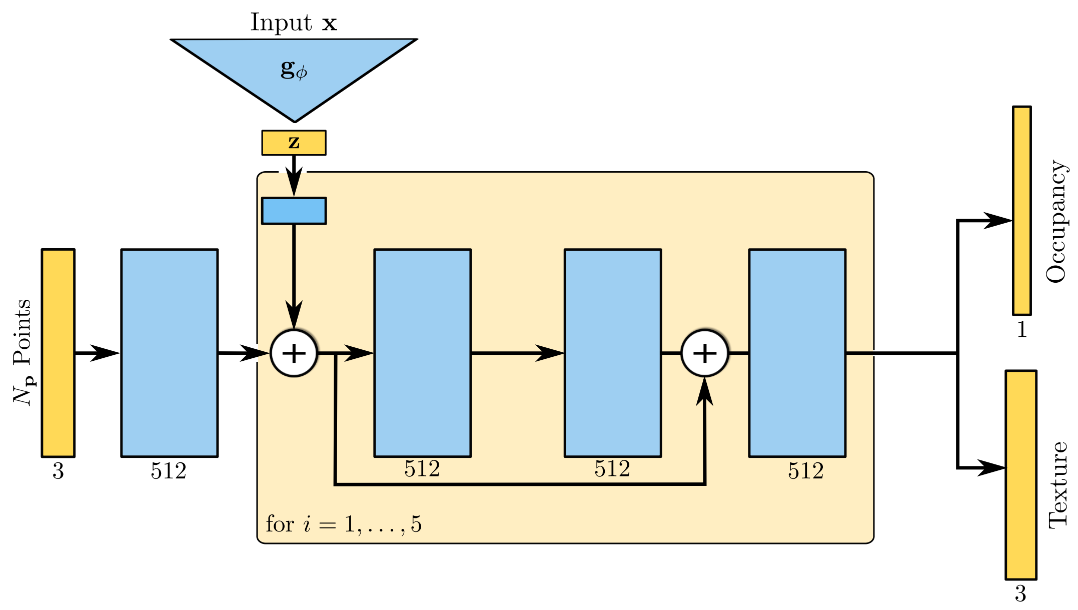


Figure 1: **Architecture.** We implement the occupancy network and the texture field as a single network that takes a batch of N_p 3D points as well as (optionally) an image $\mathbf{x}$ as input and outputs both one-dimensional occupancy probabilities and three-dimensional texture vectors in RGB space. We first pass the point coordinates (p_1, p_2, p_3) through a fully connected layer with ReLU [7] activation. We then pass the output to five consecutive ResNet [8] blocks with ReLU activation and a hidden dimension of 128 (2D supervised model) or 512 (2.5D supervised model). Optionally, we can condition our model on latent code $\mathbf{z}$ by passing it through a fully connected layer and adding it before every ResNet block. In this case, We use a ResNet-18 [8] as image encoder $\mathbf{g}_\phi$. The final output is a batch of N_p one-dimensional occupancy values and three-dimensional RGB color values.

that rays that leave the unit cube at its "sides" have a smaller s_n than rays that leave the unit cube at its "back face"). Using these entities, we can express the step size as $\Delta s := \frac{s_n - s_0}{n}$ and define the j evaluation points in world-coordinates as

$$\mathbf{p}_j^{\text{ray}} = \mathbf{r}(j\Delta s + s_0) \tag{2}$$

As indicated in the main publication, we first find the smallest j for which the occupancy prediction f_θ changes from free space ($f_\theta < \tau$) to occupied space ($f_\theta \geq \tau$):

$$j = \underset{j'}{\operatorname{argmin}} \left(f_\theta(\mathbf{p}_{j'+1}^{\text{ray}}) \geq \tau > f_\theta(\mathbf{p}_{j'}^{\text{ray}}) \right) \tag{3}$$

We then obtain the approximation to our surface depth $\hat{d}$ by iteratively applying the secant method on the interval $[j\Delta s + s_0, (j+1)\Delta s + s_0]$.

1.3. Backward Pass

Remember that in automatic differentiation, the input of the backward pass is the gradient $\lambda = \frac{\partial \mathcal{L}}{\partial \hat{d}}$ and the output is $\lambda \frac{\partial \hat{d}}{\partial \theta}$. In the main publication, we derive an analytic formula for the right part of the output which is given by

$$\frac{\partial \hat{d}}{\partial \theta} = - \left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \right)^{-1} \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} \tag{4}$$

In practice, we do not calculate the gradients wrt. a single surface depth prediction, but a whole batch of predictions in parallel. If we would calculate all entities individually, storing in particular $\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta}$ would result in an excessive memory consumption. To make batch processing feasible, we rewrite the output as

$$\lambda \frac{\partial \hat{d}}{\partial \theta} = \mu \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} \quad \text{with} \quad \mu = - \left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \right)^{-1} \lambda \tag{5}$$

We observe that $\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta}$ is a standard backward operation applied to neural network f_θ, and $-\left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w}\right)^{-1} \lambda$ is an element-wise multiplication of the inverse of the calculated dot product and the input λ. We can therefore first calculate μ and then call the standard backward operation $\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta}$ with μ as *input*. As a result, we **do not need to store** $\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta}$ and arrive at a **computationally efficient** backward pass.

1.4. Metrics

In 3D reconstruction tasks, it is common to report *accuracy* and *completeness* values which are the (mean) Euclidean distances from the prediction to the ground truth and from the ground truth to the prediction, respectively. More formally, let's define the ground truth and predicted surface of a 3D shape as $\mathcal{S}_{\text{GT}}$ and $\mathcal{S}_{\text{pred}}$, respectively. We define accuracy and completeness of $\mathcal{S}_{\text{pred}}$ wrt. $\mathcal{S}_{\text{GT}}$ as

$$\text{Accuracy}(\mathcal{S}_{\text{pred}}|\mathcal{S}_{\text{GT}}) := \frac{1}{|\mathcal{S}_{\text{pred}}|} \int_{\mathbf{p} \in \mathcal{S}_{\text{pred}}} \min_{\mathbf{q} \in \mathcal{S}_{\text{GT}}} \|\mathbf{p} - \mathbf{q}\|_2 \, d\mathbf{p} \tag{6}$$

$$\text{Completeness}(\mathcal{S}_{\text{pred}}|\mathcal{S}_{\text{GT}}) := \frac{1}{|\mathcal{S}_{\text{GT}}|} \int_{\mathbf{q} \in \mathcal{S}_{\text{GT}}} \min_{\mathbf{p} \in \mathcal{S}_{\text{pred}}} \|\mathbf{p} - \mathbf{q}\|_2 \, d\mathbf{q} \tag{7}$$

Since there is an inherent trade-off between accuracy and completeness, analyzing them separately provides only little useful information. For instance, it is possible to achieve a good completeness score (at the expense of accuracy) by predicting a lot of spurious geometry. Similarly, a good accuracy score can be achieved (at the expense of completeness) by predicting only a small fraction of the object where the method is most certain. For all experiments, we therefore report the Chamfer-L_1 [13] distance which is the mean of the two entities:

$$\text{Chamfer-}L_1(\mathcal{S}_{\text{pred}}, \mathcal{S}_{\text{GT}}) := \frac{1}{2} \left(\text{Accuracy}(\mathcal{S}_{\text{pred}}|\mathcal{S}_{\text{GT}}) + \text{Completeness}(\mathcal{S}_{\text{pred}}|\mathcal{S}_{\text{GT}})\right) \tag{8}$$

In practice, we sample $100{,}000$ points from both the prediction and the ground truth mesh which gives estimates of the true values. For the multi-view reconstruction experiments, we use the official DTU [1] evaluation script in "surface mode" and use the provided mean accuracy and completeness values to calculate the Chamfer-L_1 distance using Eq. (8).

1.5. Training Protocol

For the single-view reconstruction experiments, we use an object batch size of 64 and a point batch size of 1024 sampled image points per object. In other words, we sample 64 random objects with 1024 random points each to form a mini-batch. We start with a ray sampling resolution of 16 and double it after 50,000, 100,000, and 250,000 iterations. We train our model using Adam [10] with an initial learning rate of $\gamma = 10^{-4}$ which we decrease by a factor of 5 after 750 and 1000 epochs, respectively.

For the multi-view experiments, we do not condition our networks and hence train a single model per object. We train with an object batch size of 1 and a point batch size of 2048. Similar to before, we start with a ray sampling resolution of 16 which we double after 50,000, 100,000, 250,000, and 500,000 iterations. Again, we use Adam [10] with an initial learning rate of $\gamma = 10^{-4}$ which we decrease by a factor of 5 after 3000 and 5000 epochs, respectively. Here, we define an epoch as one iteration over all views.

To determine when to stop training, we follow common practice and evaluate the validation metric every 2000 iterations. We use the checkpoint which performs best on the validation set as the final model.

1.6. Normal Loss

We can use a normal loss on the surface points to incorporate the prior belief that surfaces are predominantly smooth. It acts as a geometric regularizer and can be in particular useful for real-world scenarios where the reconstruction problem might be less constrained and noisier.

In our implicit shape representation f_θ the unit normal vector for a point $\mathbf{p} \in \mathbb{R}^3$ is given by

$$\mathbf{n}_\theta(\mathbf{p}) = \frac{\nabla_{\mathbf{p}} f_\theta(\mathbf{p})}{||\nabla_{\mathbf{p}} f_\theta(\mathbf{p})||_2}. \tag{9}$$

While we would be able to calculate the gradient $\nabla_{\mathbf{p}} f_\theta(\mathbf{p})$ directly, in practice we use the central difference approximation to reduce GPU memory requirements

$$\nabla_{\mathbf{p}} f_\theta(\mathbf{p}) \approx \frac{1}{\Delta h} \cdot \begin{pmatrix} f_\theta(\mathbf{p}_x + \Delta h) - f_\theta(\mathbf{p}_x - \Delta h) \\ f_\theta(\mathbf{p}_y + \Delta h) - f_\theta(\mathbf{p}_y - \Delta h) \\ f_\theta(\mathbf{p}_z + \Delta h) - f_\theta(\mathbf{p}_z - \Delta h) \end{pmatrix} \tag{10}$$

where Δh is a small step size. We want to formulate a local smoothness constraint for points that lie on the surface of our representation. Towards this goal, we apply an ℓ_2 loss on the normals for surface points and corresponding randomly sampled neighboring points.

More formally, let $\mathcal{P}_0$ denote the set of pixels $\mathbf{u}$ which lie inside the object mask and for which the occupancy network f_θ predicts a surface depth $\hat{d}$. The corresponding surface points are $\mathbf{r}_{\mathbf{u}}(\hat{d_{\mathbf{u}}}) = \hat{\mathbf{p}_{\mathbf{u}}}$ (see Sec. 4.3 of the main publication). We define the surface loss as

$$\mathcal{L}_{\text{normal}}(\theta) = \sum_{\mathbf{u} \in \mathcal{P}_0} ||\mathbf{n}(\hat{\mathbf{p}_{\mathbf{u}}}) - \mathbf{n}(\mathbf{q}_{\mathbf{u}})||_2 \tag{11}$$

where $\mathbf{q}_{\mathbf{u}}$ is a neighboring point of $\hat{\mathbf{p}_{\mathbf{u}}}$ sampled uniform randomly in a fixed neighborhood around $\hat{\mathbf{p}_{\mathbf{u}}}$.

1.7. Loss Weighting

We define five different types of losses (see Sec. 3.3 of the main publication): the photometric loss $\mathcal{L}_{\text{RGB}}$, the depth loss $\mathcal{L}_{\text{Depth}}$, the normal loss $\mathcal{L}_{\text{normal}}$, the freespace loss $\mathcal{L}_{\text{freespace}}$ and the occupancy loss $\mathcal{L}_{\text{occupancy}}$. We apply $\mathcal{L}_{\text{RGB}}$, $\mathcal{L}_{\text{Depth}}$, and $\mathcal{L}_{\text{normal}}$ to points which lie inside the ground truth object mask and where our model predicts a depth. We use $\mathcal{L}_{\text{freespace}}$ for points where our model falsely predicts a depth, and $\mathcal{L}_{\text{occupancy}}$ for points where it falsely does not predict a depth. The loss for a sampled view is given by the sum of the components

$$\mathcal{L}(\theta) = \lambda_0 \mathcal{L}_{\text{RGB}}(\theta) + \lambda_1 \mathcal{L}_{\text{Depth}}(\theta) + \lambda_2 \mathcal{L}_{\text{normal}}(\theta) + \lambda_3 \mathcal{L}_{\text{freespace}}(\theta) + \lambda_4 \mathcal{L}_{\text{occupancy}}(\theta) \tag{12}$$

In practice, we did not perform hyperparameter optimization and set set $\lambda_3 = \lambda_4 = 1$. The hyperparameter λ_2 controls the strength of the smoothness prior and should be adjusted depending on the context. For the single-view experiments, we always use $\lambda_2 = 0.05$ and for the multi-view experiments, we always use $\lambda_2 = 0.1$. We set λ_0 and λ_1 to 0 or 1 depending on the model and supervision type. When training with both losses, we set $\lambda_0 = 1$ and $\lambda_1 = 10$.

These choices are only valid if all loss values are in the same order of magnitude which is always given for $\mathcal{L}_{\text{RGB}}$, $\mathcal{L}_{\text{freespace}}$, and $\mathcal{L}_{\text{occupancy}}$. The depth values, however, depend on the camera intrinsics as they are inverse proportional to the focal length f: $d = \frac{x_{\text{image}}}{f \cdot x_{\text{3D}}}$. While the depth values are in the correct order of magnitude for the experiments on the ShapeNet dataset, they are not for the experiments on the DTU dataset where we hence would have to adapt λ_1 accordingly. Optionally, one can also define the loss on the predicted 3D points where no adjustment has to be done as we always use the same volume of interest (unit cube):

$$\sum_{\mathbf{u} \in \mathcal{P}_0} ||\hat{\mathbf{p}}_{\mathbf{u}} - \mathbf{p}_{\mathbf{u}}||_2 = \sum_{\mathbf{u} \in \mathcal{P}_0} \left|\left| \mathbf{r}_0 + \mathbf{w}_{\mathbf{u}} d_{\mathbf{u}} - (\mathbf{r}_0 + \mathbf{w}_{\mathbf{u}} \hat{d}_{\mathbf{u}}) \right|\right|_2 = \sum_{\mathbf{u} \in \mathcal{P}_0} ||\mathbf{w}_{\mathbf{u}}||_2 \left| d_{\mathbf{u}} - \hat{d}_{\mathbf{u}} \right|_1 \approx c_{\mathbf{w}} \sum_{\mathbf{u} \in \mathcal{P}_0} \left| \hat{d}_{\mathbf{u}} - d_{\mathbf{u}} \right|_1 = \mathcal{L}_{\text{Depth}} \tag{13}$$

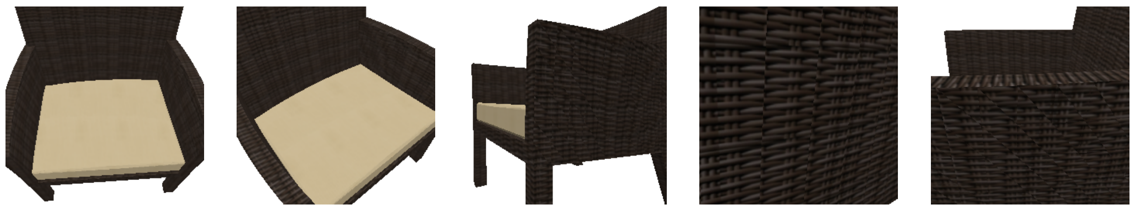
where $c_{\mathbf{w}}$ is a constant. We use the approximate symbol as $||\mathbf{w}_{\mathbf{u}}||_2$ can only be approximated by constant $c_{\mathbf{w}}$ because it slightly varies with different $\mathbf{u}$. However, the differences are negligible and we find that it works equally well in practice.

2. Data

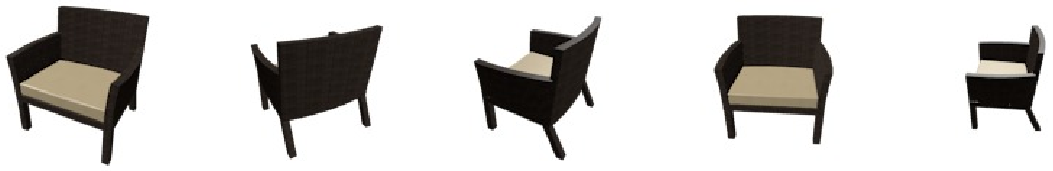
In this section, we provide additional information regarding the data pre- and post-processing steps which we perform for the single- and multi-view experiments reported in the paper.

2.1. Pre-Processing for the ShapeNet dataset

We render 24 RGB images with hemispheric lighting and depth maps for all objects from the Choy et al. [4] subset (13 classes) of the ShapeNet dataset [3]. We render all images at a resolution of 256^2 and additionally save the camera intrinsics and extrinsics to files. We follow Mescheder et al. [13] and resize the objects to the unit cube to avoid scale ambiguity. In contrast to previous works, we first sample the viewpoint randomly on the northern hemisphere and next, we sample the distance of the camera origin to the center of the object uniformly from the interval $[0.5, 1.5]$. As Fig. 2 shows, compared to the renderings from Choy et al. [4], our renderings are more diverse in terms of viewpoints.

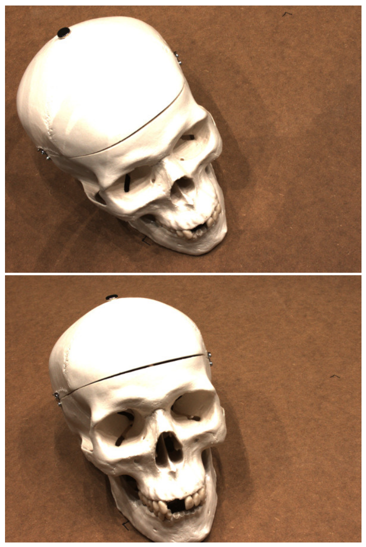


(a) Our Renderings.

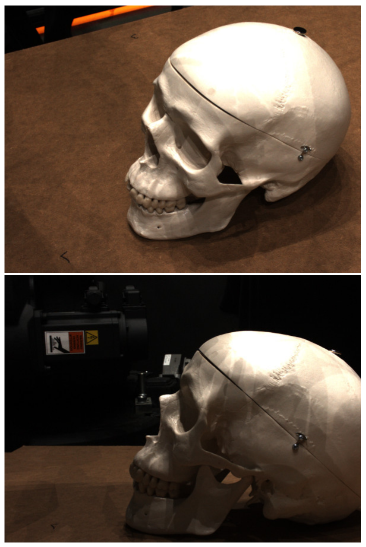


(b) Choy et al. [4] Renderings.

Figure 2: **Comparison of Renderings.** We show five random samples from our renderings and the Choy et al. [4] renderings for a randomly sampled object from the "chair" category of the ShapeNet dataset [3]. We sample not only the camera origin on the northern hemisphere but also the distance to the object, resulting in more diverse views.



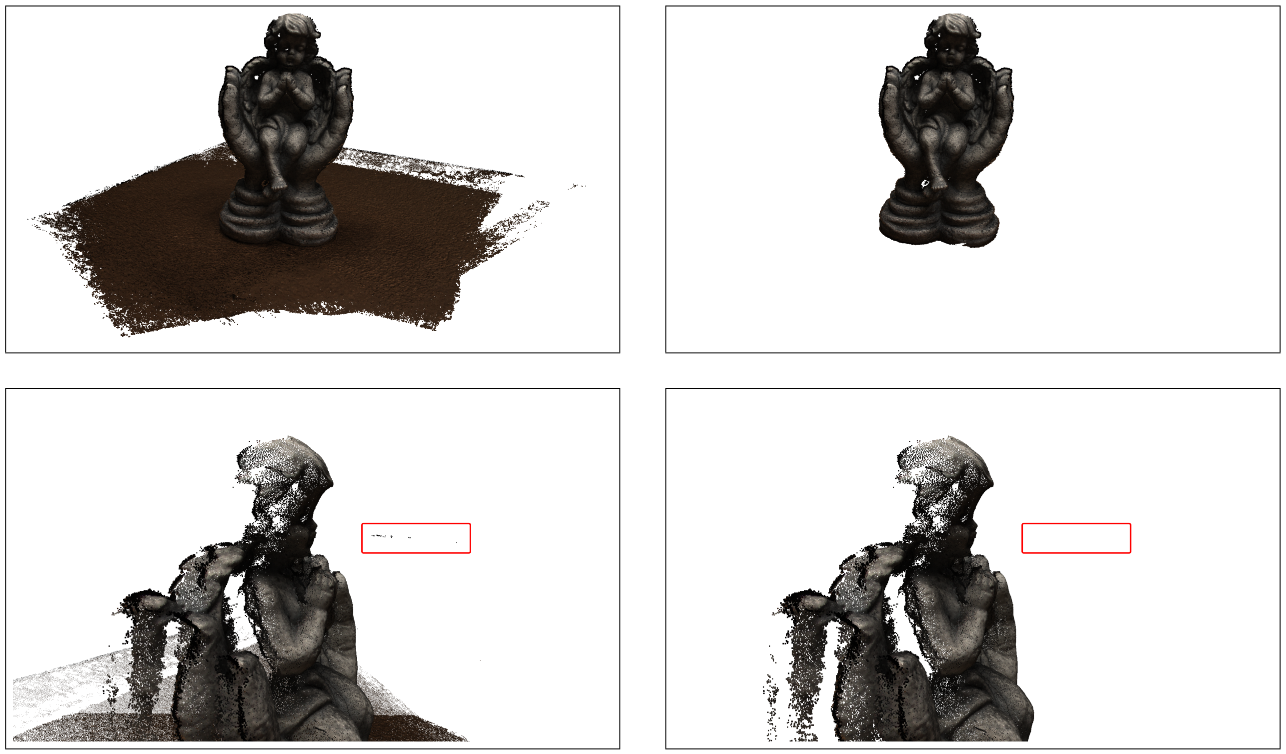
(a) Input Images to our Method.



(b) Removed Images.

Figure 3: **DTU Image Filtering.** We show examples for images used as input to our method and images with profound changes in lighting conditions for scan 65 of the DTU dataset.

In the single-view reconstruction experiments, we use the first point on a ray which lies inside all object masks as the depth value for the $\mathcal{L}_{\text{occupancy}}$ loss when training only with 2D supervision (see 3.4 of the main publication). We generate the visual hulls for all objects in the training and validation split as follows. We first project points on a regular 3D grid into all views and evaluate if they are inside or outside the mesh. They are only inside if they are contained in at least one view and if they lie inside all object masks. Otherwise, they are identified as outside. Finally, we can run the marching cubes algorithm [12] to extract the mesh. We render the visual hulls from the training views to obtain depth maps which we only use for encouraging occupancy along the ray with the $\mathcal{L}_{\text{occupancy}}$ loss.



(a) Without Post-Processing.

(b) With Post-Processing.

Figure 4: **DTU Post-Processing.** We show the predicted point cloud of Colmap [16] before (Fig. 4a) and after foreground mask-based (Fig. 4b) post-processing for scan 118 of the DTU dataset [1]. This post-processing step removes the ground plane as well as outlier points (indicated by the red box) and allows for a fairer comparison to our method.

2.2. Pre-Processing for the DTU Dataset

We observe that the multi-view images contained in the DTU dataset exhibit profound changes in lighting conditions (see Fig. 3). As the image caption process was automated for the DTU dataset, the same image IDs correspond to the same camera locations for all object scans. It allows us to identify the IDs of images for which the Lambertian assumption of our model is strongly violated and to remove the corresponding images. The IDs of the removed images are $3-8$, $16-22$, and $36-40$.

One of the fundamental steps of the classic multi-view stereo pipeline is to obtain sparse depth maps which are later fused in a post-processing step, e.g., using volumetric fusion [5]. We want to investigate to which degree our method is able to incorporate this sparse depth information in addition to the RGB images to reconstruct 3D shape and texture. Towards this goal, we obtain sparse depth maps by running MVS on the objects using Colmap [17] with the provided camera intrinsics and extrinsics from DTU. To investigate different types of depth supervision (see Section 3.5.3), we further run Structure-from-Motion (SfM) with Colmap [17] to obtain even sparser depth supervision.

2.3. Post-Processing for Baselines on the DTU Dataset

For the multi-view reconstruction experiment, our method is trained with multi-view images, corresponding camera information, and object masks as input. To provide a fair comparison, we use the object masks to filter the predictions of the baselines before we run screened Poisson surface reconstruction (sPSR) [9] to obtain the final output.

Towards this goal, we project all points into all views, and keep only points which are visible in at least one view and which are never outside an object mask. This removes the ground plane as well as outlier points from the predictions (see Fig. 4). To ensure that we do not remove non-outlier points due to inaccuracies in the object mask annotations, we apply binary dilation with a disk size of 12 before evaluating whether a point lies inside or outside an object mask. Intuitively, this "thickens" the inside / outside boundaries of the object masks.

	Ray Sampling Resolution	Fine-Tune Resolution	Chamfer-L_1
$\mathcal{L}_{\text{Depth}}$	16	-	0.257
$\mathcal{L}_{\text{Depth}}$	32	-	0.241
$\mathcal{L}_{\text{Depth}}$	64	-	0.232
$\mathcal{L}_{\text{Depth}}$	128	-	**0.229**
$\mathcal{L}_{\text{Depth}}$	16	128	**0.229**

Table 1: **Ray Sampling Resolution.** We show quantitative results for our method trained with the depth loss $\mathcal{L}_{\text{Depth}}$, using different ray sampling resolutions on the ShapeNet "chair" category. The results indicate that surprisingly a sampling resolution of 16 is already enough to learn an accurate representation. Training or fine-tuning on a higher resolution further leads to small improvements.

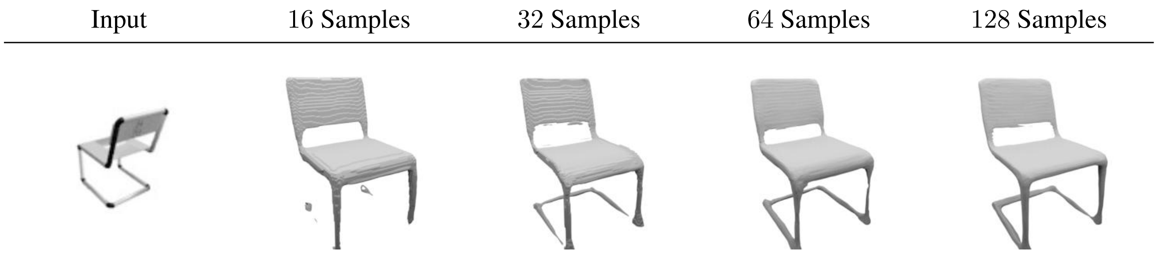


Figure 5: **Ray Sampling Resolution.** We show the output of our model trained with the depth loss $\mathcal{L}_{\text{Depth}}$, using different ray sampling resolutions for the single-view reconstruction experiment on the ShapeNet "chair" category. Our method is able to represent the overall shape approximately for a sampling resolution of 16. However, it struggles to represent finer details which are better preserved when using high sampling resolutions.

3. Additional Experimental Results

In this section, we provide additional experimental results regarding the ray sampling resolution (Section 3.1), the smoothness prior (Section 3.2), and discuss failure cases (Section 3.3). Additional single- (Section 3.4) and multi-view reconstruction results can be found in Section 3.5.

3.1. Ray Sampling Resolution

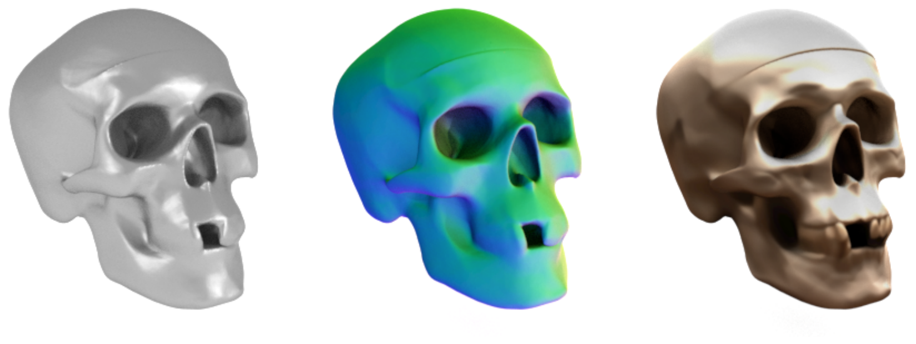
We determine the approximate surface location by evaluating the occupancy network f_θ at n equally spaced samples on the ray cast from the camera origin towards the sampled image point (see Sec. 3.3 of the main publication). This raises the question of how many samples are needed during training. To investigate this further, we train our model with $\mathcal{L}_{\text{Depth}}$ on the ShapeNet [3] "chair" category with sampling resolutions 16, 32, 64, and 128. We also investigate training with a low resolution of 16 and fine-tuning it with 128 samples per ray. We choose the "chair" category as it is challenging due to large intra-class variations and fine structures.

In Table 1 and Fig. 5 we show quantitative and qualitative results. Surprisingly, a resolution of 16 is already enough to capture the overall shape. However, both the quantitative and qualitative results improve when increasing the sampling resolution. In particular, fine details (e.g. the thin parts of the chair in Fig. 5) are better preserved for resolutions larger than 32. Importantly, first training with a low resolution and then fine-tuning at a higher resolution results in the same reconstruction quality as training directly at high resolution. We use this observation in our experiments to speed up training using a coarse-to-fine scheme (see Section 1.5).

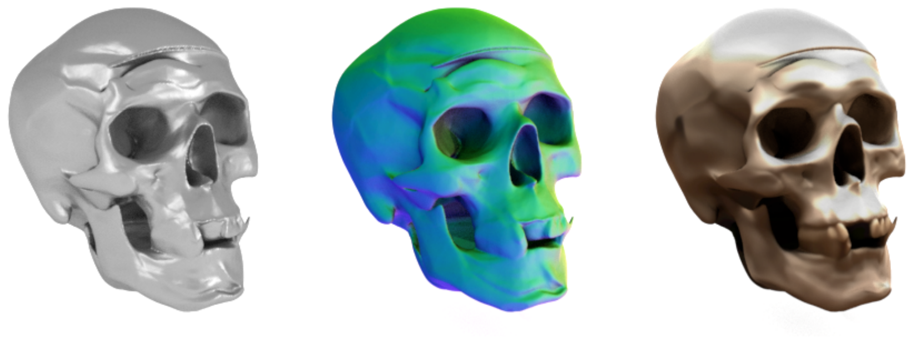
3.2. Smoothness Prior

To investigate the effect of the normal loss which acts as a smoothness prior, we train our model in the multi-view reconstruction experiment with the three different weightings $\lambda_2 \in [1., 0.1, 0.]$ for scan 65 ("skull" object) from the DTU dataset.

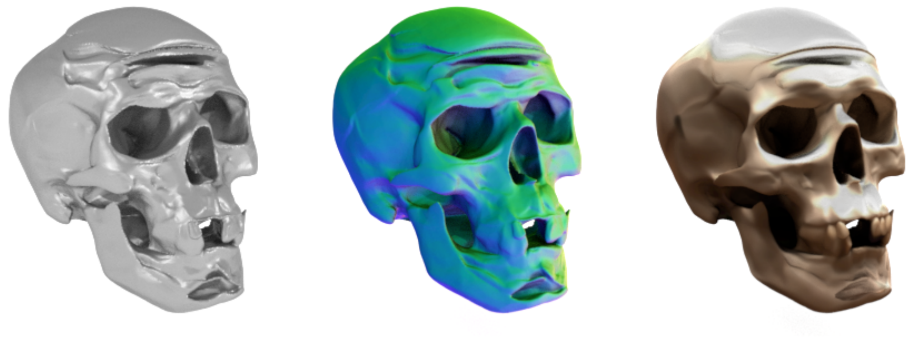
Fig. 6 and Fig. 7 show results for the different weightings of λ_2. As expected, it acts as a smoothness prior and the weighting offers control over how strongly the 3D geometry should be regulated. While we use a small value of $\lambda_1 = 0.05$ in our single-view experiments, higher regularization might be preferable for applications where smooth shapes are required.



(a) Our model ($\mathcal{L}_{\text{RGB}}$) with $\lambda_2 = 1$



(b) Our model ($\mathcal{L}_{\text{RGB}}$) with $\lambda_2 = 0.1$



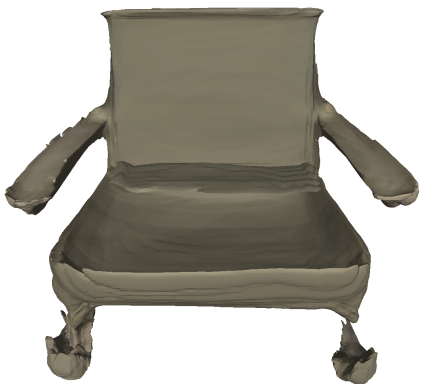
(c) Our model ($\mathcal{L}_{\text{RGB}}$) with $\lambda_2 = 0$.

Figure 6: **Effect of Smoothness Prior.** We show the output of our model trained with $\mathcal{L}_{\text{RGB}}$ and the three different smoothness prior weights $\lambda_2 \in [1, 0.1, 0]$.

3.3. Failure Case Analysis

As we focus on 3D reconstruction in this work, we always show the extracted meshes rendered with a physically-based renderer for showing qualitative results. However, our method also allows to directly render the representation from arbitrary views at arbitrary resolution. In Fig. 8a we show a comparison of physically-based renderings of the extracted geometry and the predicted renderings of our model trained with $\mathcal{L}_{\text{RGB}}$. We selected failure cases to highlight the differences. Training with only 2D supervision is less constrained and the model sometimes predicts geometry which the predicted renderings do not reveal. For example, the tabletop and the seat plate of the bench in Fig. 8a are not completely planar. Besides, very fine geometry can get lost during the mesh extraction step.

We further observe that sometimes our models struggle to reconstruct very fine structures (see Fig. 8b). We identify the simple fully-connected architecture in combination with a global latent code as the main reason and plan to investigate novel architectures in the future.


(a) Without Smoothness Prior.

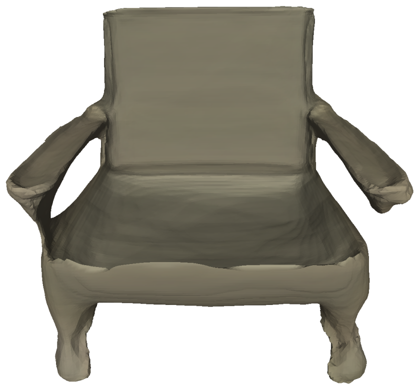
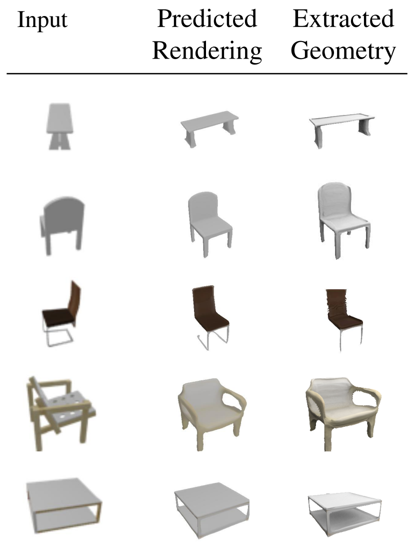
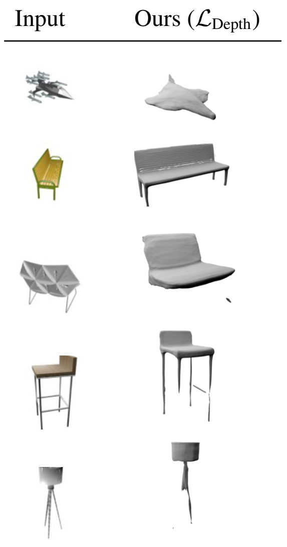

(b) With Smoothness Prior.

Figure 7: **Effect of Smoothness Prior.** We show the output of our single-view reconstruction model trained with 2D supervision with and without the smoothness prior.



(a) **Volumetric Rendering without Mesh Extraction.** We show the input, the rendering predicted by our model trained with $\mathcal{L}_{\text{RGB}}$, and the extracted geometry rendered with a physically-based renderer.



(b) **Thin Geometric Structures.** We show failure cases for our model trained with $\mathcal{L}_{\text{Depth}}$. We observe that it sometimes struggles to reconstruct very fine geometric details.

Figure 8: **Failure Case Analysis.** Fig. 8a shows that our model ($\mathcal{L}_{\text{RGB}}$) sometimes predicts geometry details which the predicted renderings do not reveal and that fine details can get lost during mesh extraction. In Fig. 8b we show examples where our model ($\mathcal{L}_{\text{Depth}}$) is not able to reconstruct very fine geometric details.

3.4. Single-View Reconstruction

In the following, we provide additional experimental results for the single-view reconstruction experiments.

3.4.1 Multi-View Supervision

In addition to the Chamfer distance reported in the main publication (see Section 1.4), we show separate accuracy (Table 2) and completeness (Table 3) values for all methods. We also present additional qualitative results in Fig. 9 and Fig. 10.

The results show that while all methods are able to predict accurate shapes, both our 2D and our 2.5D supervised methods are not restricted to certain topologies and produce smooth results. When trained with the photometric loss $\mathcal{L}_{\text{RGB}}$, we find

category	2D Supervision DRC (Mask) [19]	 SoftRas [11]	 Ours ($\mathcal{L}_{\text{RGB}}$)	2.5D Supervision DRC (Depth) [19]	 Ours ($\mathcal{L}_{\text{Depth}}$)	3D Supervision 3D R2N2 [4]	 ONet [13]	 Pixel2Mesh [20]
airplane	0.922	**0.140**	0.202	0.499	**0.148**	0.226	**0.148**	0.186
bench	-	0.246	**0.221**	-	**0.176**	0.219	**0.183**	0.210
cabinet	-	**0.182**	0.184	-	**0.146**	0.223	**0.157**	0.202
car	0.317	0.151	**0.134**	0.296	**0.119**	0.200	**0.123**	0.154
chair	0.764	0.297	**0.257**	0.631	**0.218**	0.262	**0.223**	0.291
display	-	0.276	**0.257**	-	**0.240**	0.311	0.282	**0.232**
lamp	-	0.336	**0.333**	-	**0.315**	0.598	**0.337**	0.355
loudspeaker	-	0.260	**0.252**	-	**0.249**	0.325	**0.268**	0.310
rifle	-	**0.150**	0.156	-	**0.169**	0.214	0.172	**0.163**
sofa	-	0.351	**0.205**	-	**0.191**	0.246	**0.192**	0.208
table	-	0.358	**0.289**	-	**0.202**	0.246	**0.201**	0.253
telephone	-	**0.127**	0.150	-	**0.129**	0.217	0.156	**0.146**
vessel	-	**0.213**	0.234	-	**0.180**	0.240	**0.202**	0.217
mean	0.668	0.237	**0.221**	0.475	**0.191**	0.271	**0.203**	0.225

Table 2: **Accuracy for Single-View Reconstruction.** We show the accuracy for the single-view reconstruction experiment on the ShapeNet dataset.

category	2D Supervision DRC (Mask) [19]	 SoftRas [11]	 Ours ($\mathcal{L}_{\text{RGB}}$)	2.5D Supervision DRC (Depth) [19]	 Ours ($\mathcal{L}_{\text{Depth}}$)	3D Supervision 3D R2N2 [4]	 ONet [13]	 Pixel2Mesh [20]
airplane	0.396	**0.158**	0.179	0.254	**0.138**	0.246	**0.154**	0.179
bench	-	0.236	**0.199**	-	**0.154**	0.201	**0.159**	0.172
cabinet	-	0.281	**0.256**	-	**0.219**	0.269	0.221	**0.186**
car	0.364	0.291	**0.258**	0.335	**0.240**	0.300	0.240	**0.154**
chair	0.556	0.380	**0.270**	0.389	**0.233**	0.302	**0.226**	0.228
display	-	0.292	**0.254**	-	**0.253**	0.336	0.267	**0.231**
lamp	-	**0.425**	0.492	-	**0.409**	0.969	0.423	**0.263**
loudspeaker	-	0.379	**0.327**	-	**0.341**	0.341	0.313	**0.259**
rifle	-	**0.159**	0.193	-	**0.117**	0.184	0.148	**0.139**
sofa	-	0.464	**0.243**	-	**0.250**	0.282	0.241	**0.214**
table	-	0.389	**0.271**	-	**0.198**	0.249	0.189	**0.177**
telephone	-	**0.134**	0.146	-	**0.132**	0.225	0.153	**0.145**
vessel	-	**0.252**	0.256	-	**0.232**	0.257	0.238	**0.186**
mean	0.438	0.295	**0.257**	0.326	**0.224**	0.320	0.229	**0.195**

Table 3: **Completeness for Single-View Reconstruction.** We show completeness for the single-view reconstruction experiment on the ShapeNet dataset.

	Supervision Views	Chamfer-L_1
$\mathcal{L}_{\text{Depth}}$	1	0.431
$\mathcal{L}_{\text{Depth}} + \mathcal{L}_{\text{RGB}}$	1	0.410
$\mathcal{L}_{\text{Depth}}$	24	**0.383**

Table 4: **Single-View Reconstruction with Single-View Supervision.** Our quantitative results indicate that the predictions of our method when only a single-view per object is available during training are comparable to the case of supervision from 24 images. Note that in contrast to the multi-view supervision experiment from before, we now condition the models on our renderings, not the ones provided by Choy et al. [4].

that our approach is able to predict sensible texture in addition to the 3D shape.

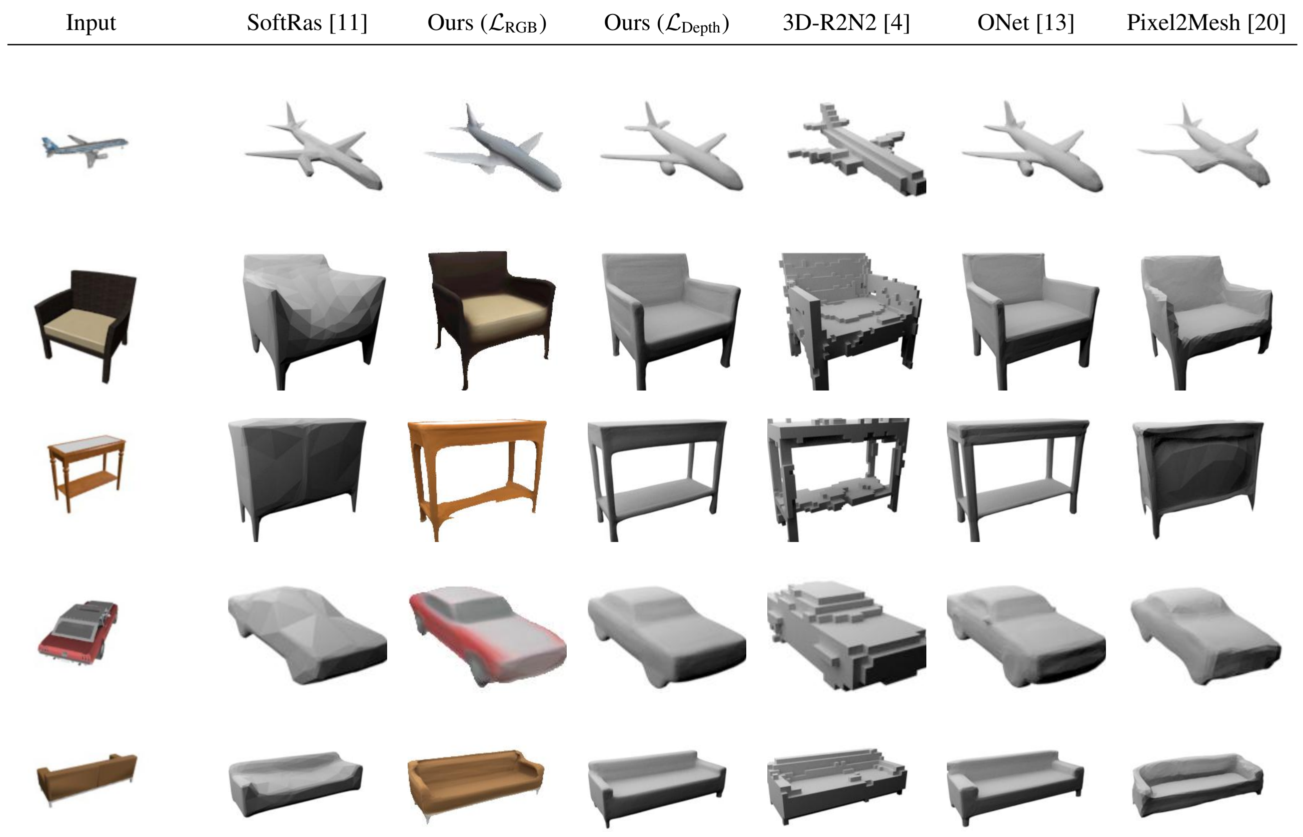

Input
SoftRas [11]
Ours ($\mathcal{L}_{\text{RGB}}$)
Ours ($\mathcal{L}_{\text{Depth}}$)
3D-R2N2 [4]
ONet [13]
Pixel2Mesh [20]

Figure 9: **Single-View Reconstruction.** We show qualitative results for our 2D supervised ($\mathcal{L}_{\text{RGB}}$) and 2.5D supervised ($\mathcal{L}_{\text{Depth}}$) methods as well as the state-of-the-art baselines Soft Rasterizer [11] (2D Supervision), 3D-R2N2 [4], ONet [13], and Pixel2Mesh [20] (all 3D Supervision) for the single-view reconstruction experiment. While all methods are able to predict accurate shapes, our method and ONet are not restricted to certain topologies and produce smoother results. In contrast to ONet [13], our method does not require ground truth watertight meshes during training but can use 2D or 2.5D supervision.

Iterations	100k	200k	300k	400k	500k
Epochs	209	417	626	846	1076
Ours	0.261	0.258	0.242	**0.221**	**0.211**
ONet [45]	**0.252**	**0.244**	**0.234**	0.233	0.230

Table 5: **Training Time Comparison.** We show the Chamfer-L_1 distances for our model and ONet over the number of training iterations for the first $500{,}000$ iterations.

3.4.2 Single-View Supervision

In Table 4 and Fig. 11 we show quantitative and qualitative results for the single-view reconstruction experiment with only single RGB-D images as supervision. Note that in contrast to before, we now condition on our renderings instead of the ones provided by Choy et al. [4] which results in a harder task (see Section 2.1).

As discussed in the main publication, our hypothesis that our model can aggregate information over multiple training instances is confirmed. Surprisingly, the models trained with only a single view per object achieve results comparable to the model trained with all 24 views.

3.4.3 Training Time

Tab. 5 shows results for our 2.5D supervised model and 3D supervised ONet [13] wrt. the training iterations. While training with the same batch size, ours receives half the number of training points per object (1024) compared to ONet (2048). Our

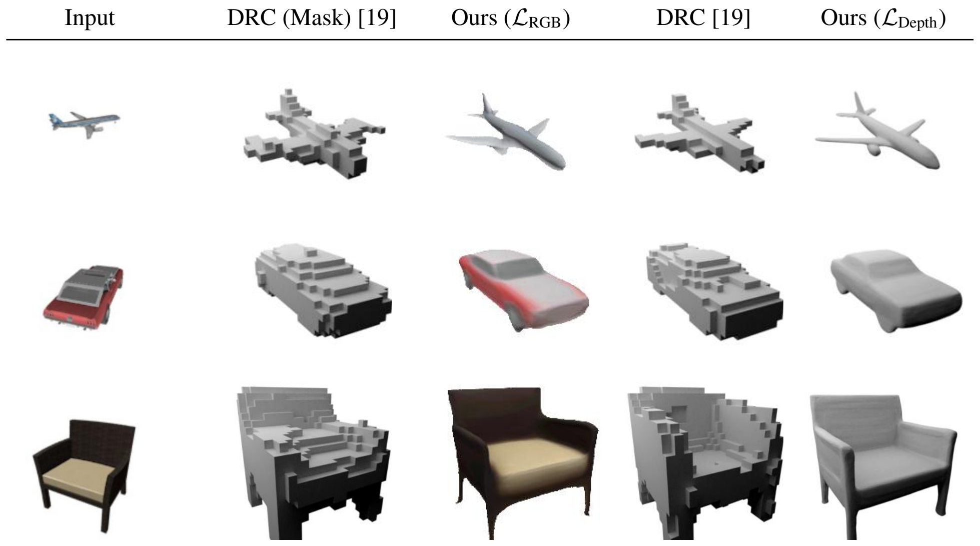

Input
DRC (Mask) [19]
Ours ($\mathcal{L}_{\text{RGB}}$)
DRC [19]
Ours ($\mathcal{L}_{\text{Depth}}$)

Figure 10: **Single-View Reconstruction.** We show qualitative results for our 2D supervised ($\mathcal{L}_{\text{RGB}}$) and 2.5D supervised ($\mathcal{L}_{\text{Depth}}$) methods as well as the 2D supervised and the 2.5D supervised version of Differentiable Ray Consistency (DRC) [19]. While all approaches are able to predict an overall correct shape, our methods do not suffer from discretization.

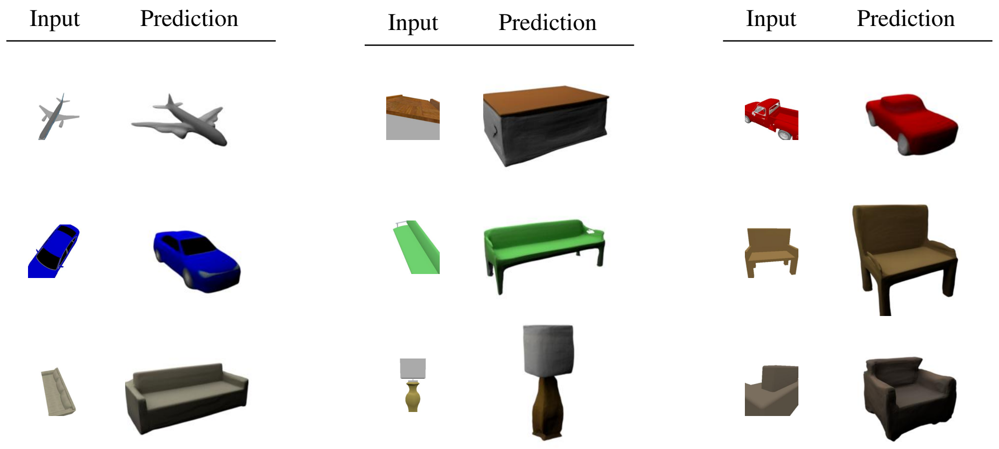


Figure 11: **Single-View Reconstruction with Single-View Supervision.** We show qualitative results for our method trained with $\mathcal{L}_{\text{RGB}}$ and $\mathcal{L}_{\text{Depth}}$ for the single-view reconstruction experiment with single-view supervision. Our model learns to predict accurate 3D geometry and texture although every object was only observed from a single view during training.

model converges faster while ONet reaches better results in the early stages of training. The latter can be a result of the larger number of training points.

3.5. Multi-View Reconstruction

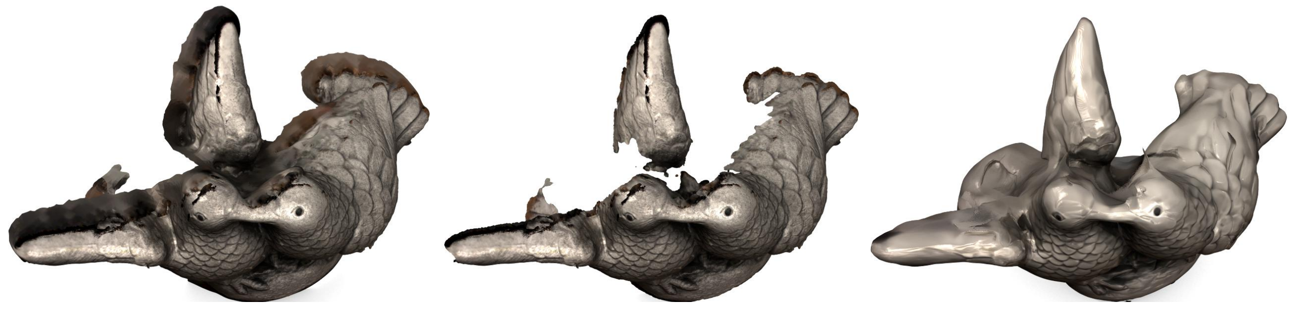
We analyze the effect of the trim parameter for the baseline methods in Section 3.5.1, give additional insights into the training progression of our method in Section 3.5.2, and investigate different types of depth supervision in Section 3.5.3. Additional results for the multi-view reconstruction experiment are provided in Section 3.5.4.

3.5.1 Effect of Trim Parameter for Classic MVS Methods

In the multi-view reconstruction experiment, we compare against classic MVS methods which produce 3D meshes as output (see Sec. 4.2 of the main publication). Towards this goal, we run screened Poisson surface reconstruction (sPSR) [9] on the

	Trim Param.	Accuracy	Completeness	Chamfer-L_1
Colmap [16] + sPSR	0	1.865	**0.696**	1.280
Colmap [16] + sPSR	1	1.864	**0.696**	1.280
Colmap [16] + sPSR	2	1.864	**0.696**	1.280
Colmap [16] + sPSR	3	1.865	**0.696**	1.280
Colmap [16] + sPSR	4	1.822	**0.696**	1.259
Colmap [16] + sPSR	5	1.453	**0.696**	1.075
Colmap [16] + sPSR	6	0.664	0.703	0.683
Colmap [16] + sPSR	7	0.383	0.763	**0.573**
Colmap [16] + sPSR	8	0.296	0.889	0.592
Colmap [16] + sPSR	9	0.245	1.167	0.706
Colmap [16] + sPSR	10	**0.221**	7.026	3.624

Table 6: **Accuracy and Completeness Tradeoff for Colmap.** We show the accuracy and completeness tradeoff for Colmap on scan 106 of the DTU dataset. By increasing the trim parameter in the post-processing pipeline, the accuracy is increased why the completeness is decreased. In this case, 7 results in the best Chamfer distance.



(a) Colmap 5 (b) Colmap 7 (c) Ours

Figure 12: **Effect of Trim Parameter.** We show screened Poisson surface reconstructions [9] with trim parameters 5 and 7 for Colmap [16] and the prediction of our model trained with $\mathcal{L}_{\text{RGB}} + \mathcal{L}_{\text{Depth}}$ for scan 106 of the DTU dataset.

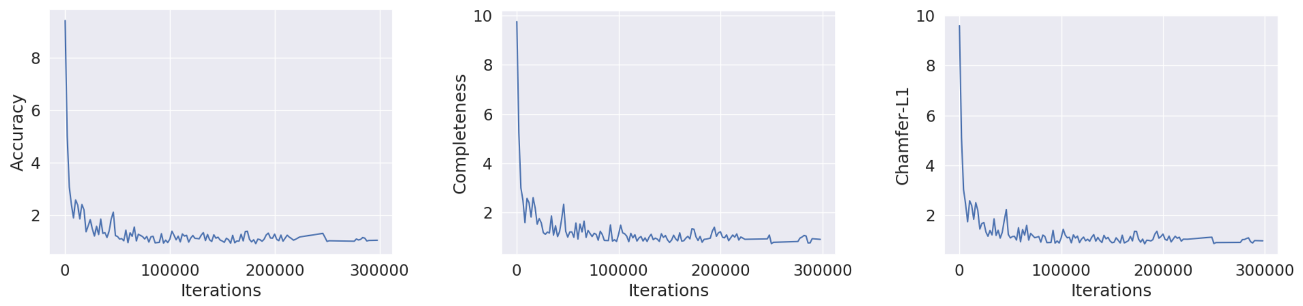
output of the baselines. As reported in the main publication, we found that the DTU surface evaluation is highly sensitive to the trim parameter of sPSR. To provide a thorough investigation, we run sPSR on the output of Colmap [16] with 11 different trim parameters and report the accuracy, completeness, and Chamfer-L_1 for scan 106 of the DTU dataset [1].

The quantitative results are summarized in Table 6. As the ground truth itself is sparse, we observe that methods are rewarded for trading off completeness for accuracy. More specifically, we see that the completeness score is saturated quickly using a trim parameter of 5. While the trim parameter 7 provides the best quantitative results, it is not the most complete one and might not be the ideal choice depending on the application (see Fig. 12). Our method, in contrast, directly results in watertight meshes without the need for tuning additional hyperparameters.

3.5.2 Multi-View Reconstruction Training Progression

The iterative progression of an algorithm often gives valuable insights into its behavior and how it can be improved. We hypothesize that our method depends on the freespace and occupancy loss in the early stages of training to predict a rough shape which is similar to the visual hull. Only later it can refine this initial estimate using RGB and (sparse) depth cues. To test our hypothesis, we train our method using $\mathcal{L}_{\text{RGB}}$ and $\mathcal{L}_{\text{Depth}}$ on scan 106 of the DTU dataset. We extract a mesh every 250 iterations and evaluate accuracy, completeness, and Chamfer-L_1 distance every 2000 iterations.

The quantitative and qualitative results summarized in Fig. 13, Fig. 14, and Fig. 15 show that our hypothesis can be confirmed. They further reveal that our method first improves the overall shape and texture prediction before it starts to add high-frequency details. This is in particular visible for the texture progression in Fig. 15.



(a) Accuracy (b) Completeness (c) Chamfer-L_1

Figure 13: **Multi-View Reconstruction Training Progression.** We show the training progression of the accuracy, completeness, and Chamfer-L_1 values against the number of iterations for our method trained with RGB images and sparse depth maps for scan "106" of the DTU dataset. While number of iterations cannot directly be converted to time as we increase the ray sampling resolution iteratively (see Section 1.5), one iteration takes roughly $0.16 - 0.25$ seconds.

	Trim Param.	Accuracy	Completeness	Chamfer-L_1
Ours ($\mathcal{L}_{\text{RGB}}$)	-	1.054	**0.760**	0.907
Ours ($\mathcal{L}_{\text{RGB}}$ + $\mathcal{L}_{\text{Depth}}$ (SfM))	-	0.940	0.821	0.881
Ours ($\mathcal{L}_{\text{RGB}}$ + $\mathcal{L}_{\text{Depth}}$ (MVS))	-	**0.789**	0.775	**0.782**

Table 7: **Different Types of Depth Supervision.** We report accuracy, completeness, and Chamfer-L_1 for our method trained only with L_{RGB}, with L_{RGB} and L_{Depth} on sparse depth maps from MVS, or with L_{RGB} and L_{Depth} on even sparser depth supervision from Structure-from-Motion (SfM). Our model can incorporate various types of depth supervision and results are improved with increased density of supervision.

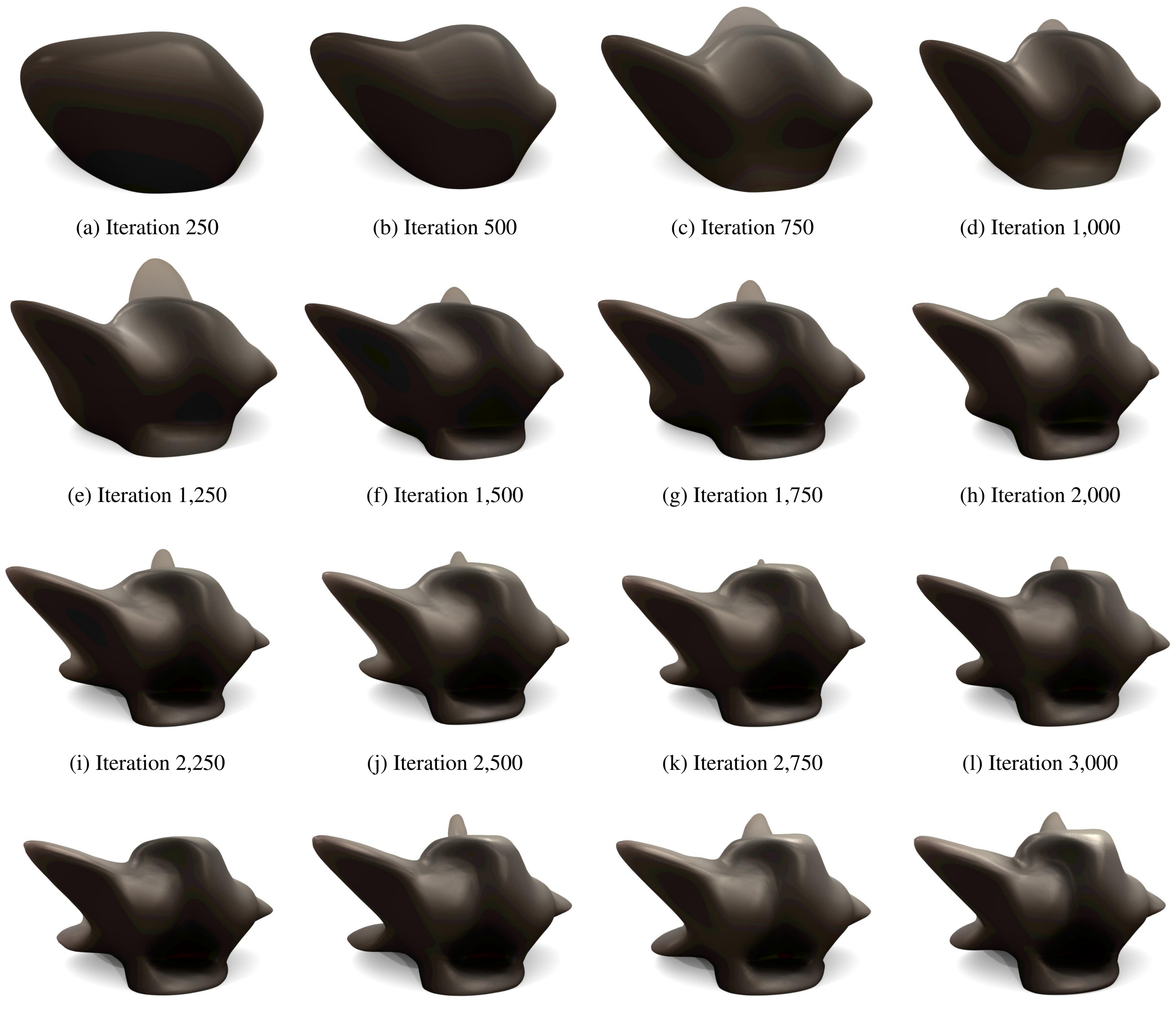
3.5.3 Different Types of Depth Supervision

In the multi-view reconstruction experiments, we investigate two cases: a.) where RGB images with corresponding object masks and camera information are given and b.) where additional sparse depth maps are given which we obtain by running MVS with Colmap [17]. Another type of supervision which one encounters often in practice is the even sparser output of Structure-from-Motion (SfM). In particular, this is a small set of 3D keypoints with visibility masks for each view mainly used for camera pose estimation. We further investigate the performance of our method when incorporating these even sparser depth maps which we obtain by running SfM with Colmap [17].

Table 7 shows quantitative results for our models trained with different supervision types. Our model improves performance when additional depth supervision is incorporated, and the improvement is larger with higher supervision density.

3.5.4 Additional Multi-View Reconstruction Results

We report accuracy, completeness, and Chamfer-L_1 for three scans from the DTU dataset [1] individually in Table 8, Table 9, and Table 10. We show additional qualitative results in Fig. 16, Fig. 17, and Fig. 18. Our method is able to predict a watertight mesh with accurate shape, normal, and texture information. Our sampling-based training strategy allows us to incorporate sparse depth information which further improves performance. Quantitatively and qualitatively our method performs comparable to highly optimized MVS methods. While the accuracy of our method is slightly lower than that of sophisticated MVS pipelines, we do not require any pre- or post-processing steps (except for mesh extraction) and directly output a more complete and watertight mesh.



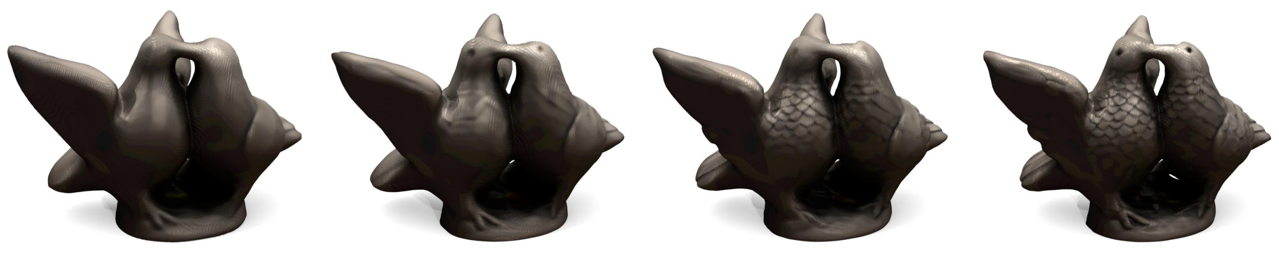
(a) Iteration 250 (b) Iteration 500 (c) Iteration 750 (d) Iteration 1,000

(e) Iteration 1,250 (f) Iteration 1,500 (g) Iteration 1,750 (h) Iteration 2,000

(i) Iteration 2,250 (j) Iteration 2,500 (k) Iteration 2,750 (l) Iteration 3,000

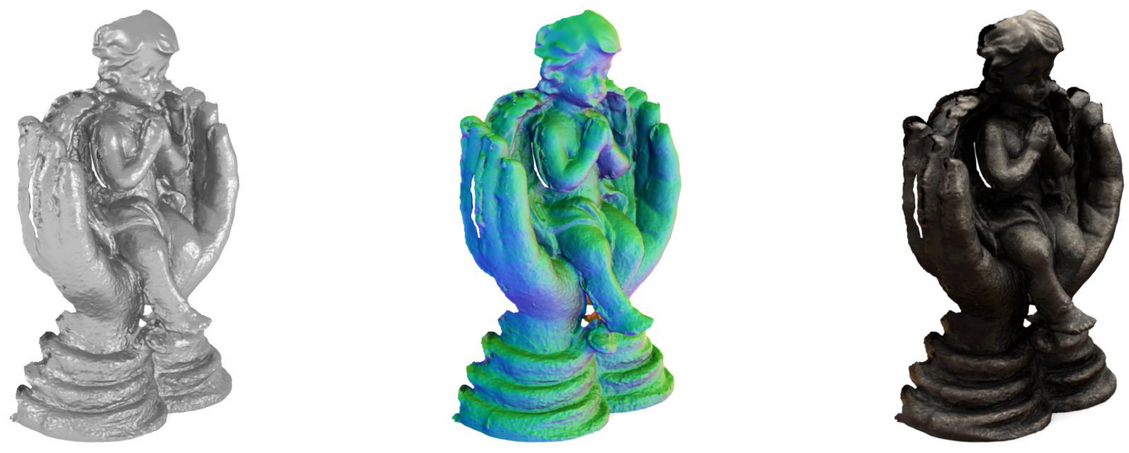
(m) Iteration 3,250 (n) Iteration 3,500 (o) Iteration 3,750 (p) Iteration 4,000

Figure 14: **Early Training Progression.** We show the early training progression of the prediction of our model trained with $\mathcal{L}_{\text{RGB}}$ and $\mathcal{L}_{\text{Depth}}$ for scan 106 of the DTU dataset.

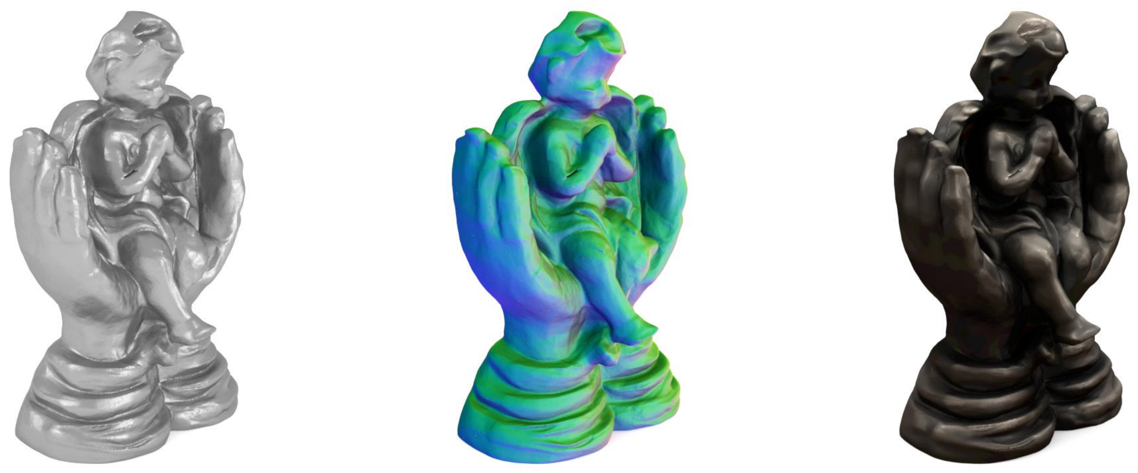


(a) 50,000 Iterations (b) 100,000 Iterations (c) 200,000 Iterations (d) 300,000 Iterations

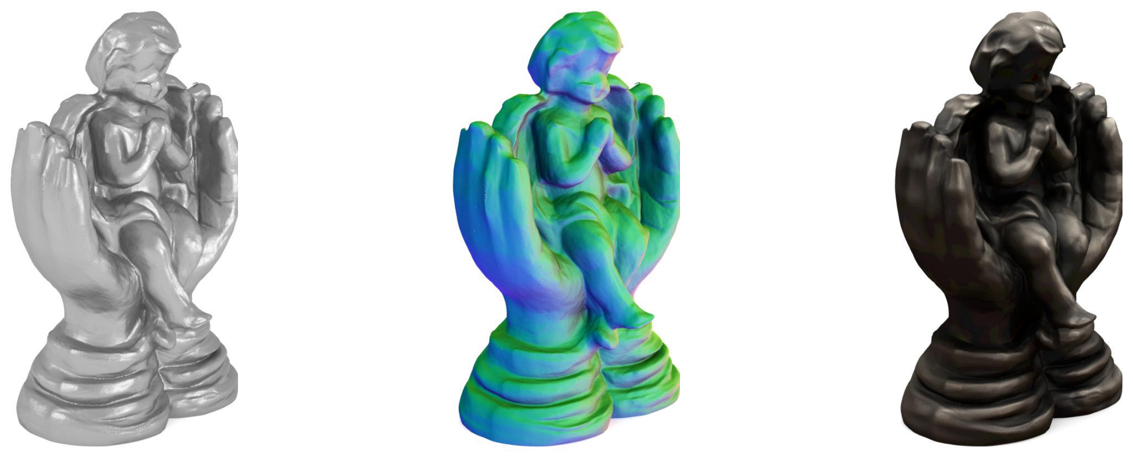
Figure 15: **Long-Term Training Progression** We show the long-term training progression of the prediction of our model trained with $\mathcal{L}_{\text{RGB}}$ and $\mathcal{L}_{\text{Depth}}$ for scan 106 of the DTU dataset. In later stages of training, mainly high frequency details, in particular texture information, are refined.



(a) Colmap [17] + sPSR



(b) Ours ($\mathcal{L}_{\text{RGB}}$)



(c) Ours ($\mathcal{L}_{\text{RGB}} + \mathcal{L}_{\text{Depth}}$)

Figure 16: **Multi-View Stereo.** We show shape, normals, and the textured shape for the screened Poisson surface reconstruction (sPSR) [9] of the output of Colmap [17], our method trained with $\mathcal{L}_{\text{RGB}}$, and our method trained with $\mathcal{L}_{\text{RGB}} + \mathcal{L}_{\text{Depth}}$ for scan 118 of the DTU dataset [1].

	Trim Param.	Accuracy	Completeness	Chamfer-L_1
Tola [18] + sPSR	0	3.165	2.020	2.592
Furu [6] + sPSR	0	2.838	1.283	2.061
Colmap [16] + sPSR	0	2.541	0.994	1.768
Camp [2] + sPSR	0	**1.834**	**0.991**	**1.412**
Tola [18] + sPSR	5	1.692	2.095	1.893
Furu [6] + sPSR	5	2.193	1.285	1.739
Colmap [16] + sPSR	5	1.642	1.163	1.402
Camp [2] + sPSR	5	**1.425**	**0.991**	**1.208**
Tola [18] + sPSR	7	**0.473**	2.420	1.446
Furu [6] + sPSR	7	0.970	1.475	1.222
Colmap [16] + sPSR	7	0.519	1.789	1.154
Camp [2] + sPSR	7	0.685	**1.129**	**0.907**
Ours ($\mathcal{L}_{\text{RGB}}$)	-	1.270	**0.849**	1.060
Ours ($\mathcal{L}_{\text{RGB}}$ + $\mathcal{L}_{\text{Depth}}$)	-	**1.049**	0.974	**1.012**

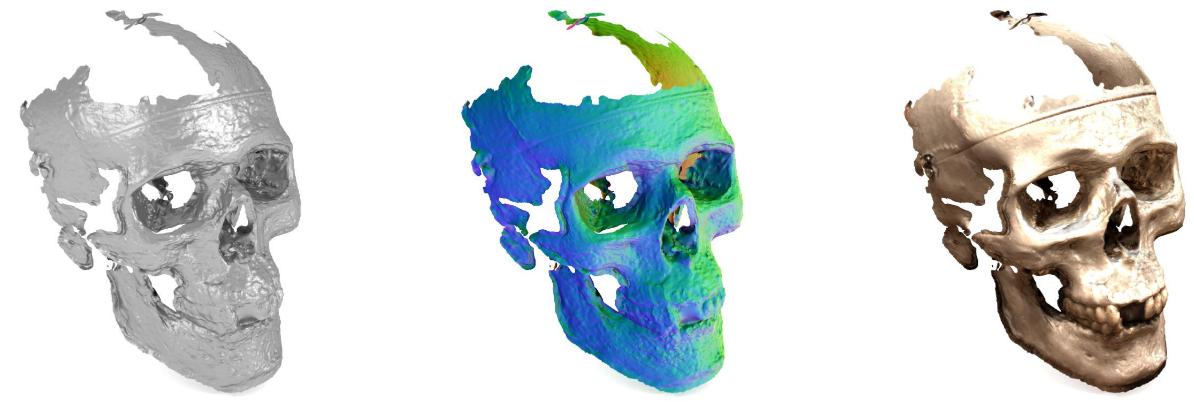
Table 8: **Multi-View Reconstruction for Scan 65.** We report accuracy, completeness, and Chamfer-L_1 for the MVS baselines and our method for scan 65 (“skull” object) of the DTU dataset.

	Trim Param.	Accuracy	Completeness	Chamfer-L_1
Tola [18] + sPSR	0	2.406	1.004	1.705
Furu [6] + sPSR	0	1.973	0.807	1.390
Colmap [16] + sPSR	0	1.865	0.696	1.280
Camp [2] + sPSR	0	**1.309**	**0.691**	**1.000**
Tola [18] + sPSR	5	1.700	1.003	1.352
Furu [6] + sPSR	5	1.630	0.807	1.218
Colmap [16] + sPSR	5	1.454	0.696	1.075
Camp [2] + sPSR	5	**1.103**	**0.692**	**0.897**
Tola [18] + sPSR	7	**0.333**	1.103	0.718
Furu [6] + sPSR	7	0.585	0.815	0.700
Colmap [16] + sPSR	7	0.383	0.763	0.573
Camp [2] + sPSR	7	0.395	**0.696**	**0.546**
Ours ($\mathcal{L}_{\text{RGB}}$)	-	1.155	0.743	0.949
Ours ($\mathcal{L}_{\text{RGB}}$ + $\mathcal{L}_{\text{Depth}}$)	-	**0.722**	**0.718**	**0.720**

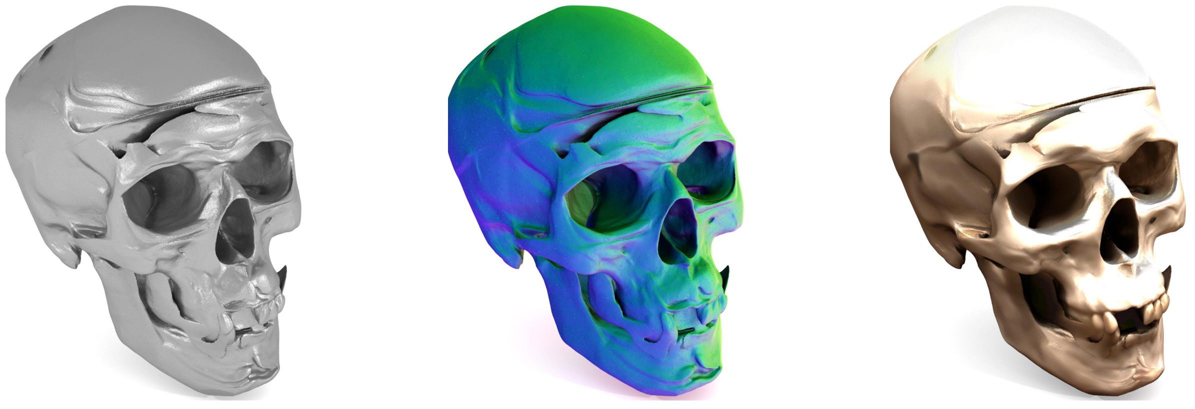
Table 9: **Multi-View Reconstruction for Scan 106.** We report accuracy, completeness, and Chamfer-L_1 for the MVS baselines and our method for scan 106 (“bird” object) of the DTU dataset.

	Trim Param.	Accuracy	Completeness	Chamfer-L_1
Tola [18] + sPSR	0	1.657	0.703	1.180
Furu [6] + sPSR	0	1.628	0.574	1.101
Colmap [16] + sPSR	0	**1.236**	0.488	**0.862**
Camp [2] + sPSR	0	3.496	**0.327**	1.911
Tola [18] + sPSR	5	1.202	0.704	0.953
Furu [6] + sPSR	5	1.376	0.574	0.975
Colmap [16] + sPSR	5	**1.103**	0.488	**0.795**
Camp [2] + sPSR	5	3.445	**0.327**	1.886
Tola [18] + sPSR	7	**0.383**	0.749	0.566
Furu [6] + sPSR	7	0.613	0.577	0.595
Colmap [16] + sPSR	7	0.436	0.507	**0.471**
Camp [2] + sPSR	7	3.319	**0.331**	1.825
Ours ($\mathcal{L}_{\text{RGB}}$)	-	0.738	0.687	0.712
Ours ($\mathcal{L}_{\text{RGB}}$ + $\mathcal{L}_{\text{Depth}}$)	-	**0.595**	**0.633**	**0.614**

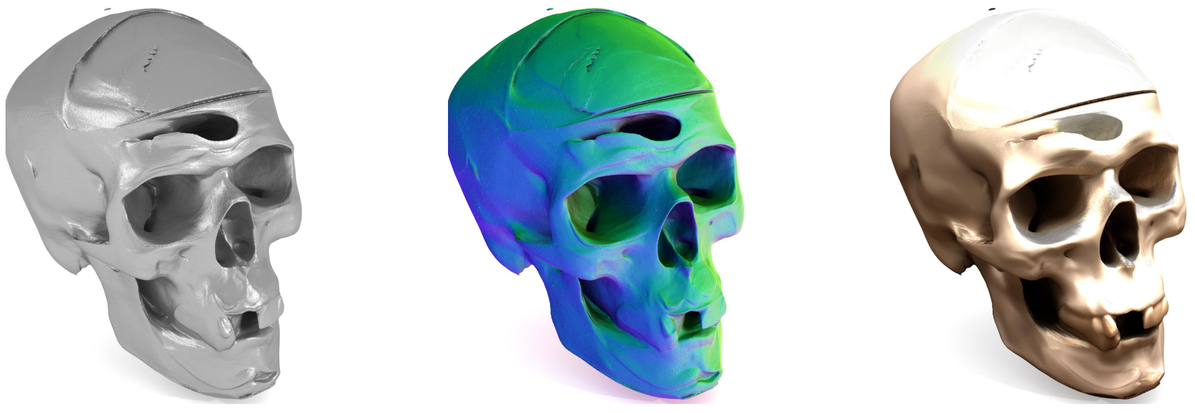
Table 10: **Multi-View Reconstruction for Scan 118.** We report accuracy, completeness, and Chamfer-L_1 for the MVS baselines and our method for scan 118 (“angel” object) of the DTU dataset.



(a) Colmap [16] + sPSR

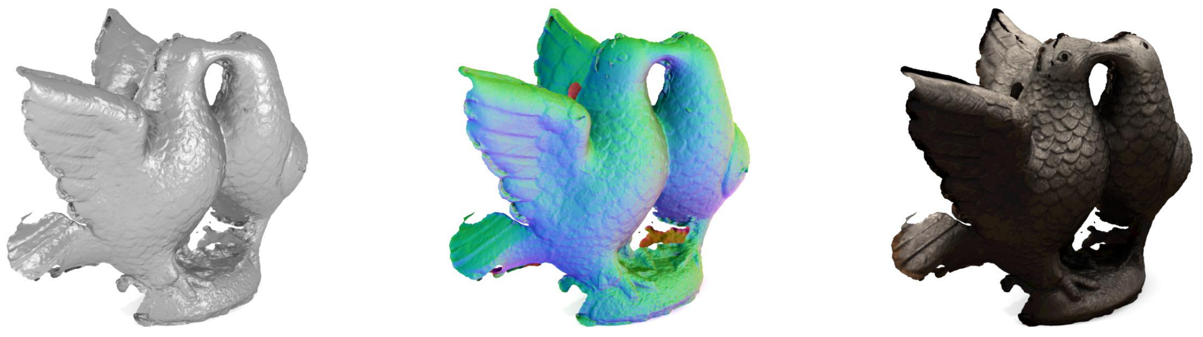


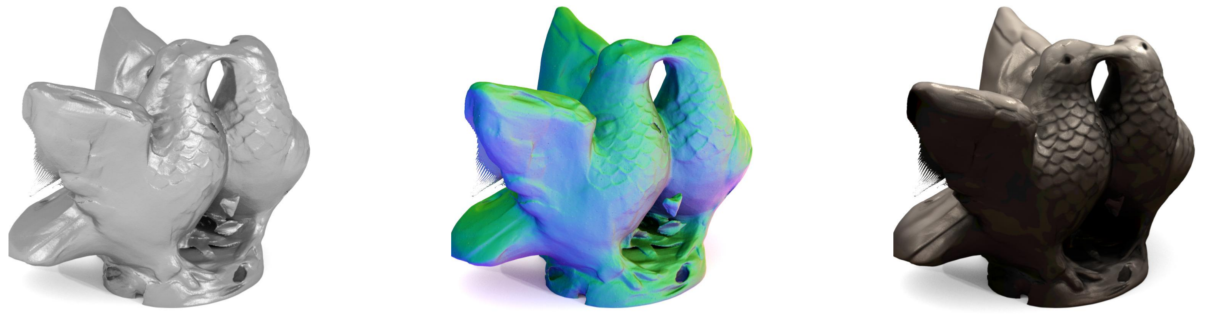
(b) Ours ($\mathcal{L}_{\text{RGB}}$)



(c) Ours ($\mathcal{L}_{\text{RGB}} + \mathcal{L}_{\text{Depth}}$)

Figure 17: **Multi-View Stereo.** We show shape, normals, and the textured shape for the screened Poisson surface reconstruction (sPSR) [9] of the output of Colmap [17], our method trained with $\mathcal{L}_{\text{RGB}}$, and our method trained with $\mathcal{L}_{\text{RGB}} + \mathcal{L}_{\text{Depth}}$ for scan 65 of the DTU dataset [1]. All methods are able to predict an overall correct shape with minor artifacts, especially near the skullcap. This area is particularly challenging because of strong specularities and overexposure in some of the views.


(a) Colmap [17] + sPSR


(b) Ours ($\mathcal{L}_{\text{RGB}}$)

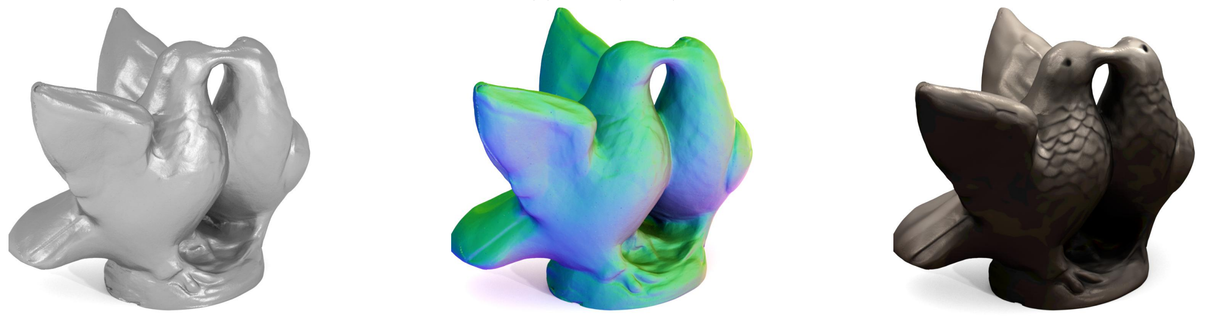

(c) Ours ($\mathcal{L}_{\text{RGB}} + \mathcal{L}_{\text{Depth}}$)

Figure 18: **Multi-View Stereo.** We show shape, normals, and the textured shape for the screened Poisson surface reconstruction (sPSR) [9] of the output of Colmap [17], our method trained with $\mathcal{L}_{\text{RGB}}$, and our method trained with $\mathcal{L}_{\text{RGB}} + \mathcal{L}_{\text{Depth}}$ for scan 106 of the DTU dataset [1]. Fig. 18b shows that our method trained with $\mathcal{L}_{\text{RGB}}$ sometimes introduces thin artifacts to cope with specularities and errors in the mask. This motivates the investigation of more complex appearance properties as well as predicting soft masks in future work.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision (IJCV)*, 120(2):153–168, 2016.

[2] Neill D.F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2008.

[3] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015.

[4] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[5] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *ACM Trans. on Graphics*, 1996.

[6] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 32(8):1362–1376, 2010.

[7] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] Michael M. Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. on Graphics*, 32(3):29, 2013.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

[11] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[12] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Trans. on Graphics*, 1987.

[13] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[15] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[16] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications (MVA)*, 23(5):903–920, 2012.

[19] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[20] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.