Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction

Supplementary material

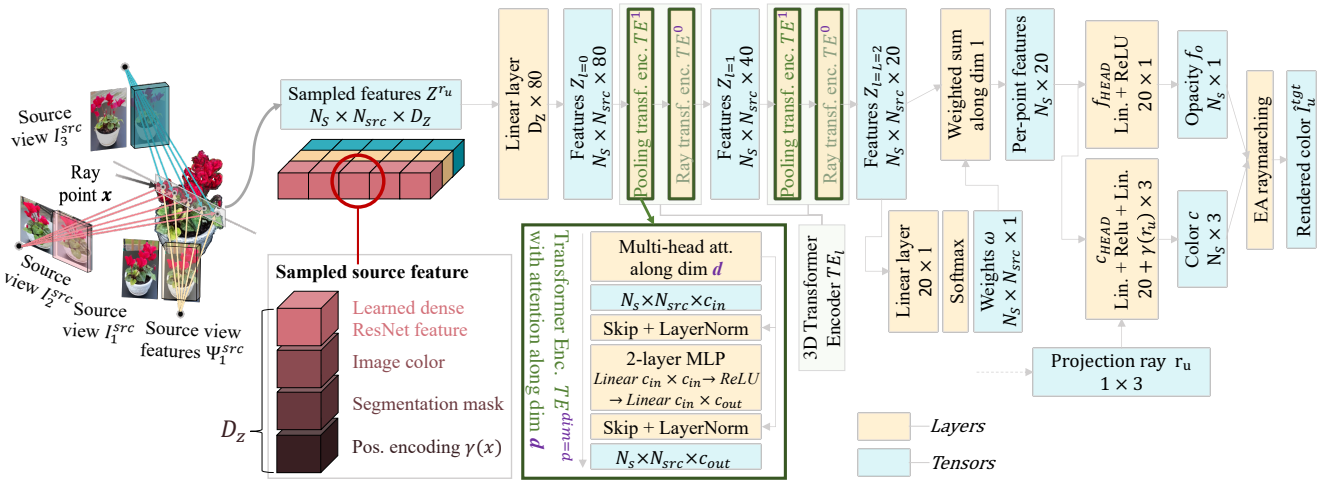

Figure I: A detailed illustration of the architecture of NerFormer .

In what follows, we provide additional quantitative results (sec. A), technical details of NerFormer and the baselines (sec. B), and details of the Human-in-the-loop 3D annotation process (sec. C).

A. Additional results

A.1. Results on all 50 categories

While tab. 3 in the main paper provides results on a subset of 10 object categories for all baselines, for completeness, in tab. I, we provide evaluation on all 50 object classes for 4 best-performing baselines according to results reported in tab. 3: NerFormer, SRN+WCE, SRN+γ+WCE, and NeRF+WCE.

Similar to tab. 3, on the `test-unseen` set, NerFormer is the best in all color-based metrics, suggesting that SRN and NeRF+WCE are prone to overfitting to the training scenes. While SRN outperforms NerFormer in some cases on `train-unseen`, we note that the autodecoders would likely yield superior performance on `train-unseen` due to their ability to capture the information from all views of a training scene in the latent scene-specific encoding.

A.2. Convergence speed

Fig. II further analyzes training convergence on the single-scene new-view synthesis task. For each method and epoch, we plot the average and standard deviation over of the per-epoch mean PSNRs of each of the 40 test scenes. The fastest converging methods are SRN, IDR, NerFormer , NeRF which also top the performance in tab. 2, indicating a significant correlation between convergence rate and performance. Furthermore, there is a large discrepancy between the train and test PSNR of γ/WCE-endowed SRN. This shows that, for the single-scene setting, increasing the model expressivity with WCE or γ can lead to overfitting to the training views.

A.3. Execution speed

Tab. II contains an evaluation of execution times for all methods from tab. 2. Here, each row reports an average time to render an 800x800 pixel image on NVIDIA Tesla V100 GPU.

A.4. Test-time autodecoder optimization

In tab. III, we provide an extension of tab. 3 containing the evaluation of the best-performing autodecoding methods at test-time. First, each method is first trained on `train-known`. Then, during evaluation, the latter freezes the trained weights and optimizes the input latent code for a given set of source frames from a test sequence. The latent codes are optimized with Adam until convergence, decaying the learning rate 10-fold whenever the optimization

	(a) Average statistics								(b) PSNR @ # source views										(c) PSNR @ target view difficulty					
	train-unseen				test-unseen				train-unseen					test-unseen					train-unseen			test-unseen		
Method	PSNR	LPIPS	ℓ_1^{depth}	IoU	PSNR	LPIPS	ℓ_1^{depth}	IoU	9	7	5	3	1	9	7	5	3	1	easy	med.	hard	easy	med.	hard
NerFormer	16.5	**0.24**	3.67	0.76	**15.7**	**0.24**	1.82	0.75	**17.5**	17.3	16.9	16.3	14.8	**16.7**	**16.4**	**16.1**	**15.5**	**13.9**	17.3	14.7	12.8	**16.5**	**13.7**	11.2
SRN+WCE	16.3	0.25	0.37	**0.81**	14.2	0.27	**0.47**	**0.77**	16.6	16.6	16.5	16.2	15.6	14.4	14.3	14.3	14.2	13.5	16.6	15.5	12.9	14.4	13.4	**11.4**
SRN+WCE+γ	**17.1**	0.25	**0.35**	**0.81**	13.7	0.28	**0.47**	0.73	17.4	**17.4**	**17.3**	**17.0**	**16.3**	14.0	13.8	13.9	13.7	13.2	**17.4**	**16.3**	**14.4**	14.0	13.1	10.6
NeRF+WCE[26]	12.6	0.27	6.21	0.54	11.6	0.27	4.54	0.51	13.0	13.0	12.8	12.6	11.6	11.9	11.8	11.8	11.6	10.8	12.9	12.1	9.4	11.9	11.1	8.7

Table I: **Results on all 50 classes from CO3D** comparing the 4 best-performing methods from tab. 3.

method	time [sec]	method	time [sec]
NerFormer	178.41	SRN[56]	1.00
NeRF+WCE[26]	113.82	SRN+γ	1.19
NeRF[43]	23.82	SRN+WCE+γ	4.20
NV[41]	0.37	SRN+WCE	5.34
NV+WCE	0.41	DVR[45]	196.94
IDR[74]	69.11	DVR+γ	204.39
IPC	0.15	P3DMesh[50]	0.09
IPC+WCE	0.16		

Table II: **Average rendering time** of an 800x800 pixel image comparing all methods from tab. 2.

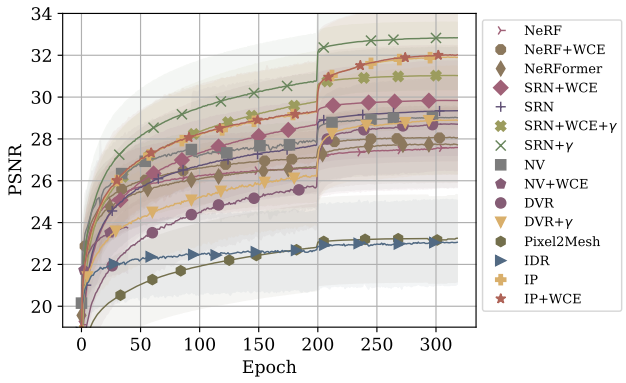

Figure II: **Convergence speed on single-scene new-view-synthesis** showing the mean and std. dev. over the per-scene training PSNRs for each method.

objective plateaus.

We observed that the latent code optimization was mostly failing for NeRF and NV. On the other hand, SRN gave slightly better performance, which we attribute to the higher smoothness of the implicit function compared to the NeRF and NV (SRN contains normalization layers while the other baselines are bare MLPs interleaving linear layers and ReLUs).

A.5. Estimating new-view difficulty

Tab. 3 contains metrics evaluated separately for three viewpoint difficulty bins. Here, we detail the process of estimating the difficulty of a testing target view.

Camera difficulty $\mathcal{D}$ Given a target camera P^{tgt} and a set of available source views $\{P_i^{\text{src}}\}_{i=1}^{N_{\text{src}}}$, the difficulty of the target view $\mathcal{D}(P^{\text{tgt}}) \in [0,1]$ is quantified as the average of the two lowest distances $d(P^{\text{tgt}}, P_i^{\text{src}})$ between the target view and each of the source views.

Camera distance d^{cam} The distance $d^{\text{cam}}(P_i, P_j) \in [0,1]$ between two cameras P_i and P_j is defined as follows. We first generate a cubical voxel grid of size 32^3 in the center of the scene with the voxel size set such that the majority of the grid is observed by all cameras in the scene. Each point $\mathbf{x}_k$, denoting the coordinates of the center of a cell in the voxel grid, is then projected to both cameras leading to a pair of projection rays $\mathbf{r}_k^i, \mathbf{r}_k^j \in \mathbb{S}^2$. We then define the similarity $s(\mathbf{r}_k^i, \mathbf{r}_k^j) = \delta[\pi_{P_i}(\mathbf{x}_k) \in \Omega_i \wedge \pi_{P_j}(\mathbf{x}_k) \in \Omega_j](1 + \mathbf{r}_k^i \cdot \mathbf{r}_k^j)$ as a dot product between the pair of rays weighted by an indicator that checks whether the projection of $\mathbf{x}_k$ simultaneously lands in the rasters $\Omega_i, \Omega_j \in [0, W] \times [0, H]$ of both cameras P_i and P_j. The camera distance d^{cam} is then defined as one minus the intersection-over-union of the similarities between all pairs of rays generated by each voxel grid point $\mathbf{x}_k$:

$$d^{\text{cam}}(P_i, P_j) = 1 - \frac{\sum_k s(\mathbf{r}_k^i, \mathbf{r}_k^j)}{\sum_k s(\mathbf{r}_k^i, \mathbf{r}_k^i) + s(\mathbf{r}_k^j, \mathbf{r}_k^j) - s(\mathbf{r}_k^i, \mathbf{r}_k^j)}.$$

Intuitively, the camera distance is proportional to the angle between the camera heading vectors adjusted by the overlap between the voxels observed by both cameras. However, merely considering the heading vectors would not take into account the intrinsics of the cameras (focal length / principal point). We thus devised d^{cam} which leverages angles between projection rays, which are a function of both the intrinsics and extrinsics.

In order to understand d^{cam}, consider the following two examples: Two cameras observing the same set of voxels at a relative angle of 0.5π would have $d^{\text{cam}}(P_i, P_j) \approx \frac{2}{3}$, while opposite-facing cameras would yield a maximum possible $d^{\text{cam}}(P_i, P_j) \approx 1$.

Camera difficulty bins Each testing target camera P^{tgt} is then assigned into one of 3 difficulty bins (*easy, medium, hard*) depending on its difficulty measure $\mathcal{D}(P^{\text{tgt}})$. More specifically, the easy cameras satisfy $0 \leq \mathcal{D}(P^{\text{tgt}}) < \frac{1}{6}$, medium $\frac{1}{6} \leq \mathcal{D}(P^{\text{tgt}}) < \frac{1}{3}$, and hard $\mathcal{D}(P^{\text{tgt}}) \geq \frac{1}{3}$.

	(a) Average statistics				(b) PSNR @ # src. views					(c) PSNR @ tgt. difficulty		
	test-unseen				test-unseen					test-unseen		
Method	PSNR	LPIPS	ℓ_1^{depth}	IoU	9	7	5	3	1	easy	medium	hard
NerFormer	**17.6**	**0.27**	0.91	**0.81**	**18.9**	**18.6**	**18.1**	**17.1**	**15.1**	**18.6**	**14.9**	**14.7**
SRN+γ+AD	13.2	0.29	0.48	0.71	13.6	13.5	13.3	13.1	12.4	13.5	11.6	11.8
SRN[56]+AD	13.8	0.28	**0.45**	0.74	14.3	14.3	14.0	13.6	12.6	14.2	12.5	11.1
NV[41]+AD	11.4	0.53	1.29	0.47	11.5	11.2	11.3	11.5	11.5	11.4	11.3	8.0
NeRF+AD	10.6	0.32	4.42	0.49	10.7	10.5	10.4	10.7	10.4	10.7	10.3	3.6

Table III: **Autodecoder latent optimization on `test-unseen`** extending the results in tab. 3. Each method labelled with +AD is first trained on `train-known`. During evaluation the latter fixes the trained weights and optimizes the input latent code for a given set of source frames from a test sequence. For context, we also compare to NerFormer , which is not an autodecoder.

B. Additional technical details

In this section we provide additional details of NerFormer and of the benchmarked baseline approaches which were outlined in sec. 5.

B.1. NerFormer

Source image features Ψ The dense pixel-wise descriptor $\Psi_{\text{CNN}}(I^{\text{src}})$ (sec. 4.2) of a source image I^{src} is a stacking of 3 types of feature tensors along the channel dimension after differentiably upsampling to a common spatial resolution $H \times W$. The feature types are: 1) Intermediate activations extracted from the source image I^{src} after "layer1", "layer2", and "layer3" layers of the ResNet34 [23] network, 2) the source segmentation mask M^{src}, 3) the raw source image I^{src}. Note that we separately map the output of each of the ResNet34 layers to a 32-dimensional feature with a 1x1 convolution followed by ℓ_2 normalization of the feature column at every spatial location.

NerFormer architecture In fig. I we provide a more detailed visualisation of the NerFormer architecture.

Rendering details Similar to NeRF [43], NerFormer optimizes loss functions for 800 randomly sampled image rays $\mathbf{r_u}$ in each training target image. Following [43], NerFormer implements a coarse and fine rendering network f_{TR}. The former, given a ray $\mathbf{r_u}$, samples 32 points $\mathbf{x}_i \in \mathbf{r_u}$ at uniform depth intervals between predefined lower and upper depth bounds. The fine rendering network then samples 16 points on $\mathbf{r_u}$ with importance sampling from the distribution proportional to the coarse rendering weights w_i.

Training details For a randomly sampled pixel $\mathbf{u}$ we thus render the color $\hat{I}^{\text{tgt}}_{\mathbf{u}}$ and an alpha value $\hat{M}^{\text{tgt}}_{\mathbf{u}} = 1 - \prod_i (1 - \exp(-\Delta f_o(\mathbf{x}, \mathbf{z}))) \in [0, 1]$, where the latter denotes the total amount of light absorbed by the implicit surface ($\hat{M}^{\text{tgt}}_{\mathbf{u}} = 1$ for complete absorption).

As noted in sec. 4.3, the optimized loss is a sum of the RGB squared error $\sum_{\mathbf{u}} \|\hat{I}^{\text{tgt}}_{\mathbf{u}} - I^{\text{tgt}}_{\mathbf{u}}\|^2$ and the segmentation binary cross entropy (BCE) $\sum_{\mathbf{u}} M^{\text{tgt}}_{\mathbf{u}} \log \hat{M}^{\text{tgt}}_{\mathbf{u}} + (1 - M^{\text{tgt}}_{\mathbf{u}}) \log(1 - \hat{M}^{\text{tgt}}_{\mathbf{u}})$. The latter ensures that rays that do not intersect the object of interest do not terminate in the scene and vice versa. Following [43], we evaluate the losses for the fine and coarse renders and optimize their sum.

B.2. NeRF

We use the implementation of NeRF [43] from PyTorch3D [50] which closely follows the original paper. Similar to NerFormer , we also add to the original losses of NeRF the BCE loss between the rendered alpha mask and the ground truth target mask. The coloring function c_{MLP} and the opacity function f_{MLP} have their architecture identical to the original implementation.

B.3. SDF methods - DVR, IDR

Here we detail the two baseline methods that represent shapes with signed distance fields (SDF). We start with introducing the SDF and a method for their rendering.

Signed distance fields While opacity functions f_o represent shapes with a measure of opaqueness of 3D spatial elements, *signed distance fields* $f_d(\mathbf{x}, \mathbf{z}) \in \mathbb{R}$, express the signed euclidean distance to the nearest point $\mathbf{x}' \in \mathcal{S}_f$ on the implicit surface $\mathcal{S}_f$.

Sphere tracing (ST) While EA is the most popular method for rendering opacity fields f_o, ST is its analogue for signed distance fields f_d. Specifically, ST renders a pixel $\mathbf{u}$ by seeking the minimum of the signed distance function f_d on the domain of 3D points belonging to the ray $\mathbf{r_u}$. ST, during its t-th iteration, refines the current estimate $\mathbf{x}^{\mathbf{r_u}}_t$ of the ray-surface intersection by moving $\Delta_t = f_d(\mathbf{x}^{\mathbf{r_u}}_t, \mathbf{z})$ units in the direction of the projection ray: $\mathbf{x}^{\mathbf{r_u}}_{t+1} = \mathbf{x}_t + \Delta_t \mathbf{r_u}$. Upon convergence at time T, the rendered color $\hat{I}^{\text{tgt}}_u = c(\mathbf{x}^{\mathbf{r_u}}_T, \mathbf{r_u}, \mathbf{z})$ comprises the response of the coloring function at the estimated ray-surface intersection.

DVR natively supports our supervisory scenario and hence no alternations were required for the training protocol of both DVR+AD and DVR. In order to implement DVR+γ, we

simply convert the input coordinates to positional embeddings and adjust the number of input channels of the first layer of DVR's implicit function accordingly. The released code [45] supports DVR+AD so no changes were required here. As mentioned in the paper, unfortunately, all our attempts to merge DVR with WCE lead to a non-converging model.

IDR Similar to DVR, IDR [74] supports our supervisory setup by default. In order to implement IDR+AD, we append the latent code $\mathbf{z}$ to the positional embeddings that are input to the implicit function, and we adjust the number of input channels of the first layer of the implicit function accordingly. IDR already takes as input the positional embeddings γ, so the extension IDR+γ does not apply here. As mentioned in the main paper, we could not obtain a converging version of IDR+WCE.

B.4. SRN

Here, we first give a brief overview of the learned SRN raymarcher, followed by describing the WCE extension of SRN.

Neural raymarching Contrasted to the explicit formulations of sphere-tracing or EA, recently, SRN [56] proposed to learn to march along the projection rays with a recurrent deep network. Similar to sphere-tracing, SRN decides at iteration t on the length of the raymarching step Δ_t by evaluating a function at the current intersection estimate $\mathbf{x}_t^{\mathbf{r}_u}$. However, instead of querying the SDF, SRN utilizes an LSTM [27] cell $f_{\text{LSTM}}(\mathbf{x}_t^{\mathbf{r_u}}, \mathbf{r_u}, \mathbf{z}, \mathbf{h}_t) = (\Delta_t, \mathbf{h}_{t+1})$ which is additionally conditioned on the ray direction $\mathbf{r_u}$ and a temporal hidden state $\mathbf{h}_t$. In this manner, the raymarcher adapts the step-size prediction based on the past marching observations.

SRN+WCE The WCE exension of SRN is straightforwardly implemented by replacing the iterative invocation of the global-encoding-conditioned implicit $f_{\text{LSTM}}(\mathbf{x}_t^{\mathbf{r_u}}, \mathbf{r_u}, \mathbf{z}_{\text{global}}, \mathbf{h}_t)$ of the SRN's raymarcher with the WCE-conditioned implicit $f_{\text{LSTM}}(\mathbf{x}_t^{\mathbf{r_u}}, \mathbf{r_u}, \mathbf{z}^{\star}_{\text{WCE}}(\mathbf{x}_t^{\mathbf{r_u}}, \{I_i^{\text{src}}\}, \{P_i^{\text{src}}\}), \mathbf{h}_t)$ This way, the learned raymarcher can "tap"" into the source views during every iteration to receive a more direct triangulation signal. As apparent from tabs. 3 and I, our WCE extension of SRN provides a very strong baseline that in fact achieves the best depth prediction performance.

Mask prediction The learned raymarcher of the original version of SRN does not render an alpha mask of the foreground object. In order to enable the latter, we extend the last layer of the SRN's coloring function c with an additional channel that is terminated with a sigmoid activation and represents the alpha value of the corresponding pixel $\mathbf{u}$. This channel is then supervised by minimizing the DICE coefficient between its output and the ground truth segmentation masks M^{tgt}.

B.5. P3DMesh

As mentioned in the paper, P3DMesh [50] deforms an initial spherical mesh template with a fixed topology with a series of convolutions on the mesh graph. As in [67], the graph convolutions accept features sampled from the source images at the 2D projections of the mesh vertices. Since P3DMesh supports conditioning only on a single-source-view, we extend to the multi-view setting by averaging over the per-vertex features sampled from each of the source views. Furthermore, note that the implementation in [50] differentiably renders the mesh with a memory-efficient version of the Soft Rasterizer [40]. The training protocol, including the employed losses and their weighting, closely follows [50].

B.6. Neural Volumes (NV)

Neural Volumes [41] is a method that represents implicit surfaces as voxel grids. In what follows, we first briefly describe voxel grids, their specific implementation in NV, and its extension with warp-conditioned embedding (NV+WCE).

Voxel grids While MLPs can label an arbitrary element of the 3D domain, a voxel grid can be seen as an implicit surface restricted to a subset of $\mathbb{R}^3$ which is uniformly subdivided to a lattice $V(\mathbf{z}) \in \mathbb{R}^{R^3}$ of $R^3; R \in \mathbb{N}_+$ cuboid elements of the same size. Note that the lattice $V(\mathbf{z})$ is a function (typically a 3D deconvnet) of $\mathbf{z}$ which allows for representing different 3D shapes. The implicit function $f_{\text{voxel}}(\mathbf{x}, \mathbf{z}) = \zeta(V(\mathbf{z}), \mathbf{x})$ is then evaluated by sampling $V(\mathbf{z})$ at the corresponding world coordinate $\mathbf{x}$, with a grid-sampling function $\zeta : \mathbb{R}^{R^3} \times \mathbb{R}^3 \mapsto \mathbb{R}^{\dim(f)}$, such as trilinear interpolation. Voxel grids also admit coloring via a volume $C(\mathbf{z}) \in \mathbb{R}^{3 \times R^3}$ which can be sampled in an analogous manner.

Neural Volumes A notable voxel-grid-based method is Neural Volumes [41], which proposed an improved sampling function $\zeta_{\text{warp}}(\zeta(W(\mathbf{z}), \mathbf{x}) + \mathbf{x}, V(\mathbf{z}))$ which refines the sampling location $\mathbf{x}$ with an offset vector $\zeta(W(\mathbf{z}), \mathbf{x}) \in \mathbb{R}^3$ sampled from a warping lattice $W(\mathbf{z}) \in \mathbb{R}^{R^3}$. Here both W and V are implemented as a 3D deconvolutional network.

NV and NV+AD NV+AD is in fact the vanilla version of [41] whose 3D deconvnets V and W accept the scene-specific latent code $\mathbf{z}_{\text{scene}}(\text{sequence}_{\text{ID}})$ (described in sec. 5.3). The "overfitting" version of NV from tab. 2 is a special case of NV+AD with a single latent code.

NV+WCE The WCE extension of Neural Volumes (NV+WCE) appends the WCE to the feature of each voxel after the second 3D deconvolution layer of the 3D convnets V and W. Here, the WCE of a voxel is generated by expressing the world coordinate $\mathbf{x}_{V_i}$ of the center of the correspoding voxel V_i and calculating the aggregate WCE

$\mathbf{z}^{\star}_{\text{WCE}}(\mathbf{x}_{V_i}, \{I_i^{\text{src}}\}, \{P_i^{\text{src}}\})$ for a set of source views $\{I_i^{\text{src}}\}$ and their cameras $\{P_i^{\text{src}}\}$. Note that a similar approach has been proposed in [31].

Training All versions of NV optimize the losses from [41] with the original weights. Furthermore, we exploit the known ground truth segmentation masks and minimize the binary cross entropy between the alpha mask returned by the raymarcher of NV and the ground truth mask M^{tgt}.

B.7. Implicit Point Cloud (IPC)

As mentioned in sec. 5, IPC represents shapes by converting a point cloud to an implicit function which is later rendered with EA raymarching.

Formally, let a point cloud $\mathcal{P}(\mathbf{z}) = \{\mathbf{x}_i\}_{i=1}^{N_{\text{pts}}}$ be an N_{pts}-sized unordered set of points, where $\mathcal{P}$ is a point cloud predictor (detailed later in this section) which accepts the latent code $\mathbf{z}$. $\mathcal{P}(\mathbf{z})$ then admits an occupancy function $f_{\mathcal{P}_\epsilon}$ defined as follows:

$$f_{\mathcal{P}_\epsilon}(\mathbf{x}', \mathbf{z}) = \delta[\|\text{NN}_{\mathcal{P}(\mathbf{z})}(\mathbf{x}') - \mathbf{x}'\| < \epsilon],$$

where $\text{NN}_{\mathcal{P}(\mathbf{z})}(\mathbf{x}') = \arg\min_{\mathbf{x}\in\mathcal{P}(\mathbf{z})} \|\mathbf{x} - \mathbf{x}'\|$ returns the nearest point from the point cloud $\mathcal{P}(\mathbf{z})$ to the query point $\mathbf{x}'$. Intuitively, $f_{\mathcal{P}_\epsilon}$ yields zero everywhere except within an ϵ neighborhood of each point cloud point $\mathbf{x}_i \in \mathcal{P}(\mathbf{z})$, where $f_{\mathcal{P}}$ yields 1. As we describe later, anchoring the implicit function on the set of cloud points allows for faster and more memory-efficient EA raymarching than in the case of the neural implicit occupancy f_{MLP} (described in sec. 4.1).

In order to color the implicit point cloud, we define its coloring function c_{IPC}:

$$c_{\text{IPC}}(\mathbf{x}', \mathbf{r}, \mathbf{z}) = c_{\text{MLP}}(\text{NN}_{\mathcal{P}(\mathbf{z})}(\mathbf{x}'), \mathbf{r}, \mathbf{z}).$$

Here c_{IPC} attaches to an arbitrary point $\mathbf{x}'$ the response of the coloring MLP c_{MLP} at $\mathbf{x}'$'s nearest point cloud neighbor $\text{NN}_{\mathcal{P}(\mathbf{z})}(\mathbf{x}')$.

Rendering IPC IPC is rendered efficiently with the PyTorch3D point cloud renderer [50, 69]. More specifically, given a target camera P^{tgt}, each point from the predicted point cloud $\mathcal{P}(\mathbf{z})$ is projected to the camera plane to form a set of 2D projections $\{\pi_{P^{\text{tgt}}}(\mathbf{x}_i) | \mathbf{x}_i \in \mathcal{P}(\mathbf{z})\}$. For each pixel coordinate $\mathbf{u} \in \{1, ..., W\} \times \{1, ..., H\}$ in the rendering lattice of the target render $\hat{I}^{\text{tgt}} \in \mathbb{R}^{3\times H\times W}$, the renderer records the ordered set

$$\Pi^{\mathbf{u}}_{\epsilon}(\mathcal{P}(\mathbf{z})) = \big(\mathbf{x}_i | \mathbf{x}_i \in \mathcal{P}(\mathbf{z}); \|\pi_{P^{\text{tgt}}}(\mathbf{x}_i) - \mathbf{u}\| \leq \epsilon \mathsf{f}_{P^{\text{tgt}}}; \\ d_{P^{\text{tgt}}}(\mathbf{x}_i) \leq d_{P^{\text{tgt}}}(\mathbf{x}_{i+1})\big),$$

of point cloud points $\mathbf{x}_i \in \mathcal{P}(\mathbf{z})$ whose 2D projections $\pi_{P^{\text{tgt}}}(\mathbf{x}_i)$ land within the $\epsilon \mathsf{f}_{P^{\text{tgt}}}$ distance from the pixel $\mathbf{u}$, and which is ordered by the depth $d_{P^{\text{tgt}}}(\mathbf{x}_i)$ of each point in the target camera P^{tgt}. $\mathsf{f}_{P^{\text{tgt}}} \in \mathbb{R}$ is the focal length of the target camera P^{tgt}.

Intuitively, $\Pi^{\mathbf{u}}_{\epsilon}(\mathcal{P}(\mathbf{z}))$ denotes the set of point cloud points whose ϵ neighborhoods are intersected by the rendering ray $\mathbf{r}_{\mathbf{u}}$ emitted from pixel $\mathbf{u}$. Note that this is an approximation: comparing the 2D camera-plane distance $\|\pi_{P^{\text{tgt}}}(\mathbf{x}_i) - \mathbf{u}\|$ to the constant $\epsilon \mathsf{f}_{P^{\text{tgt}}}$ corresponds to orthographic projections of the point neighborhoods, whereas our cameras are perspective. However, the orthographic approximation is mild in our case, since the distance of the point cloud points from the camera is relatively large compared to its focal length.

The EA raymarching then takes the set of $\mathbf{u}$'s 3D points $\Pi^{\mathbf{u}}_{\epsilon}(\mathcal{P}(\mathbf{z}))$ in order to render the color $\hat{I}^{\text{tgt}}_{\mathbf{u}} \in \mathbb{R}^3$:

$$\hat{I}^{\text{tgt}}_{\mathbf{u}}(\mathbf{r}_{\mathbf{u}}, \mathbf{z}) = \sum_{\mathbf{x}_i \in \Pi^{\mathbf{u}}_{\epsilon}(\mathcal{P}(\mathbf{z}))} w_i(\mathbf{x}_i, \mathbf{z}, \mathbf{u}) c_{\text{IPC}}(\mathbf{x}_i, \mathbf{r}_{\mathbf{u}}, \mathbf{z}).$$

For IPC, the weight $w_i(\mathbf{x}_i, \mathbf{z}, \mathbf{u}) = \Big(\prod_{j=0}^{i-1} T_j^{\text{IPC}}(\mathbf{x}_i, \mathbf{z}, \mathbf{u})\Big)\big(1 - T_i^{\text{IPC}}(\mathbf{x}_i, \mathbf{z}, \mathbf{u})\big)$ is the product of emission and absorption functions with the transmission term T_i^{IPC} defined as

$$T_i^{\text{IPC}}(\mathbf{x}_i, \mathbf{z}, \mathbf{u}) = \underbrace{f_{\mathcal{P}_\epsilon}(\mathbf{x}_i, \mathbf{z})}_{=1} \frac{\|\mathbf{u} - \pi_{P^{\text{tgt}}}(\mathbf{x}_i)\|}{\epsilon \mathsf{f}_{P^{\text{tgt}}}},$$

which approximately measures the amount of light transmitted through the spherical ϵ neigborhood of a point $\mathbf{x}_i$ which intersects the projection ray $\mathbf{r}_{\mathbf{u}}$. To demonstrate this, observe that for a pixel $\mathbf{u}^{\text{intersect}} = \pi_{P^{\text{tgt}}}(\mathbf{x}_i)$ which coincides with the projection of the 3D point $\mathbf{x}_i$, the transmission $T_i^{\text{IPC}}(\mathbf{x}_i, \mathbf{z}, \mathbf{u}^{\text{intersect}}) = 0$, *i.e.* no light is transmitted through $\mathbf{x}_i$ and the corresponding color $c_{\text{IPC}}(\mathbf{x}_i, \mathbf{r}_{\mathbf{u}^{\text{intersect}}}, \mathbf{z})$ is fully rendered. On the contrary, for a pixel $\mathbf{u}^{\text{outside}} = \pi_{P^{\text{tgt}}}(\mathbf{x}_i + \epsilon)$ outside the epsilon neighborhood, the unit transmission $T_i^{\text{IPC}}(\mathbf{x}_i, \mathbf{z}, \mathbf{u}^{\text{outside}}) = 1$ signifies that all light passes through and the point's color is ignored during rendering. Note that the above equation is very similar to the top-k point cloud rasterizer of SinSyn [69].

Point cloud predictor $\mathcal{P}(\mathbf{z})$ The point cloud predictor $\mathcal{P}(\mathbf{z})$ is the same for both IPC+AD and IPC+WCE. More specifically, $\mathcal{P}(\mathbf{z}) = \{\bar{\mathbf{x}}_i + o_{\text{MLP}}(\bar{\mathbf{x}}_i, \mathbf{z})\}_{i=1}^{N_{\text{pts}}}$ offsets a fixed set of template points $\bar{\mathcal{P}} = \{\bar{\mathbf{x}}_i\}_{i=1}^{N_{\text{pts}}}$ with an offset function $o : \mathbb{R}^3 \times \mathbb{R}^{D_{\mathbf{z}}} \mapsto \mathbb{R}^3$ implemented as an MLP with the same architecture as f_{MLP}. Therefore, o alters the template point cloud to match a specific shape given its latent shape code $\mathbf{z}$.

IPC+AD and IPC+WCE For IPC+AD, the offset function o accepts the video-specific latent code $\mathbf{z}_{\text{scene}}(\text{sequence}_{\text{ID}})$ described in sec. 5.3, while for IPC+WCE, o takes as input the aggregate warp-conditioned embedding $\mathbf{z}^{\star}_{\text{WCE}}(\bar{\mathbf{x}}_i, \{I_i^{\text{src}}\}, \{P_i^{\text{src}}\})$ evaluated at each template point $\bar{\mathbf{x}}_i \in \bar{\mathcal{P}}$. Finally, the single-scene version, abbreviated simply as IPC in tab. 2, is a special case of IPC+AD with $\mathbf{z}_{\text{scene}}(\text{sequence}_{\text{ID}}) := \mathbf{0}$ set to a constant zero vector.

Training All versions of IPC optimize the MSE between the rendered image $\hat{I}^{\text{tgt}}$ and the ground truth colors I^{tgt}. Furthermore, we make use of the ground truth segmentation masks and minimize the Chamfer distance between the set of 2D projections of the predicted point cloud points $\mathcal{P}(\mathbf{z})$, and the 2D points of the ground truth segmentation mask [36]. Note that a standard segmentation loss, such as DICE [57] or Binary Cross Entropy between the rendered alpha mask and the ground truth segmentation mask, do not apply here. This is because the gradients generated by the alpha mask renders of IPC are not well-defined and do not lead to convergence.

C. 3D annotations with Human-in-the-loop

In sec. 3, we outlined the process of annotating the AMT-collected videos with 3D ground truth. Here, we further detail the semi-automated process of labelling the quality of camera tracking and the 3D dense point cloud of the captured videos (Paragraph 4 in sec. 3).

We initialize the process by annotating an initial set of several hundreds of reconstructions with a binary label "accurate / inaccurate" by visually inspecting both the camera tracks $(P_i|P_i \in R^{4\times4})_{i=1}^{N_I}$ and the scene point cloud $\mathcal{P}(\mathcal{V})$. From each video, we then extract various metrics that are indicative of the reconstruction quality such as a per-pixel RGB and depth error of the rendered point cloud, the number of registered cameras, final bundle adjustment energy etc. The full set of metrics is outlined in tab. IV. We then train a binary Support Vector Machine (SVM [12]) with an RBF kernel that regresses the binary label given the reconstruction metrics as input.

Afterwards, the trained SVM classifies all previously unlabelled videos. In line with the uncertainty principle [59], we manually annotate a subset of previously unlabelled samples that are the closest to the SVM decision boundary. We further correct significant classification errors by inspecting the highest/lowest scoring samples. In this manner, we alternate between SVM training and manual annotation until 1.5k labels are collected (8 % of the whole dataset).

In order to validate the SVM's performance, we conduct a 5-fold cross-validation on the set of annotated videos. The cross-validation indicates that the SVM has 90% and 78% accuracy for classifying the camera tracking and point cloud quality respectively.

References

[1] Adel Ahmadyan, Liangkai Zhang, Jianing Wei, Artsiom Ablavatski, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. *arXiv preprint arXiv:2012.09988*, 2020. 2

[2] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 2, 4

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv*, 2016. 6

[4] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv*, 2017. 8

[5] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001. 2

[6] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 2

[7] Wenzheng Chen, Jun Gao, Huan Ling, Edward J. Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In *Proc. NeurIPS*, 2019. 2

[8] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems*, pages 9609–9619, 2019. 1, 2

[9] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv preprint arXiv:1602.02481*, 2016. 2

[10] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 2

[11] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994. 3

[12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3), 1995. 17

[13] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Proc. NeurIPS*, 2016. 1

[14] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 2

[15] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *2017 International Conference on 3D Vision (3DV)*, pages 402–411. IEEE, 2017. 2

[16] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020. 2

[17] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning

Metric	Domain	Description
$BA_{\text{final cost}}$	$\mathbb{R}$	The final value of the Bundle Adjustment (BA) cost function.
$BA_{\text{termination}}$	$\{0, 1\}$	The termination state of BA (converged/not converged).
$\mu_{\text{det score}}$	$[0, 1]$	An average over per-frame detection scores of the PointRend object detector.
$\mu_{\text{perc detected}}$	$[0, 100]$	Percentage of frames in which the category of interest is detected with PointRend.
N_{cameras}	$\mathbb{N}$	The number of cameras registered during BA.
$N_{\text{sparse pts}}$	$\mathbb{N}$	Number of points in the sparse point cloud.
$PCL^{\text{render}}_{\ell\text{depth}}$	$\mathbb{R}$	The average ℓ_1 depth error between the renders of the fused pointcloud $\mathcal{P}(\mathcal{V})$ into each camera P_i of a video $\mathcal{V}$ and the corresponding dense depth map P_i.
$PCL^{\text{render}}_{\ell\text{rgb}}$	$\mathbb{R}$	The average ℓ_1 RGB error between the renders of the fused pointcloud $\mathcal{P}(\mathcal{V})$ into each camera P_i of a video $\mathcal{V}$ and the corresponding frame I_i.
$PCL^{\text{render}}_{IoU}$	[0, 1]	The average Jaccard Index between the renders of the fused point cloud $\mathcal{P}(\mathcal{V})$ into each camera P_i of a video $\mathcal{V}$ and the corresponding PointRend segmentation M_i.
$PCL^{\text{direction cover}}$	$\mathbb{N}$	Measures the coverage of the views of the point cloud $\mathcal{P}(\mathcal{V})$ with the number of occupied bins in the azimuth/elevation map of projection rays corresponding to each dense point cloud point $\mathbf{x}_j$ and a camera P_{i}.

Table IV: **The list of SfM and point-cloud reconstruction metrics** that serve as a set of features for training the active-SVM that labels camera and reconstruction quality.

shape templates with structured implicit functions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7154–7164, 2019. 2

[18] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016. 2, 5

[19] Georgia Gkioxari, Justin Johnson, and Jitendra Malik. Mesh R-CNN. In *Proc. ICCV*, 2019. 2

[20] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. Shape and viewpoint without keypoints. *arXiv preprint arXiv:2007.10982*, 2020. 2

[21] GoogleResearch. Google scanned objects, September. 2

[22] David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. 2016. 4

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 14

[24] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (SIGGRAPH Asia Conference Proceedings)*, 37(6), November 2018. 2

[25] Philipp Henzler, Niloy Mitra, and Tobias Ritschel. Escaping plato's cave using adversarial training: 3d shape from unstructured 2d image collections. In *Proc. ICCV*, 2019. 1, 2, 4

[26] Philipp Henzler, Jeremy Reizenstein, Patrick Labatut, Roman Shapovalov, Tobias Ritschel, Andrea Vedaldi, and David Novotny. Unsupervised learning of 3d object categories from videos in the wild. *arXiv*, 2021. 1, 2, 4, 5, 6, 7, 8, 13

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), 1997. 15

[28] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in neural information processing systems*, pages 2802–2812, 2018. 2, 4

[29] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Godisart, Bart C. Nabbe, Iain A. Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *PAMI*, 41(1), 2019. 2

[30] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. ECCV*, 2018. 2

[31] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in neural information processing systems*, pages 365–376, 2017. 2, 16

[32] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. 2

[33] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020. 3

[34] Nilesh Kulkarni, Abhinav Gupta, David F. Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *Proc. CVPR*, pages 449–458, 2020. 2

[35] Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical surface mapping via geometric cycle consistency. In *Proc. ICCV*, 2019. 2

[36] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. *Proc. ECCV*, 2020. 2, 17

[37] Joseph J. Lim, Hamed Pirsiavash, and Antonio Torralba. Parsing IKEA Objects: Fine Pose Estimation. *Proc. ICCV*, 2013. 1

[38] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and

C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Proc. ECCV*, 2014. 3

[39] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *arXiv preprint arXiv:2007.11571*, 2020. 2

[40] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. *arXiv.cs*, abs/1904.01786, 2019. 2, 4, 5, 15

[41] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019. 2, 4, 5, 6, 7, 13, 14, 15, 16

[42] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 2

[43] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020. 2, 4, 6, 13, 14

[44] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised learning of 3D representations from natural images. *arXiv.cs*, abs/1904.01326, 2019. 2

[45] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020. 2, 4, 6, 7, 13, 15

[46] Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proc. ICCV*, 2019. 1, 4

[47] David Novotny, Diane Larlus, and Andrea Vedaldi. Learning 3d object categories by looking around them. In *Proc. ICCV*, 2017. 2

[48] David Novotný, Diane Larlus, and Andrea Vedaldi. Capturing the geometry of object categories from video supervision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. 2

[49] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 2

[50] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 4, 6, 7, 8, 13, 14, 15, 16

[51] Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Advances in neural information processing systems*, pages 4996–5004, 2016. 2

[52] Shunsuke Saito, , Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *arXiv preprint arXiv:1905.05172*, 2019. 2

[53] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3, 4

[54] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 3

[55] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *arXiv preprint arXiv:2007.02442*, 2020. 2, 4

[56] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *CoRR*, abs/1906.01618, 2019. 1, 2, 4, 5, 6, 7, 8, 13, 14, 15

[57] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*. 2017. 17

[58] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *Proc. ECCV*, 2016. 5

[59] Simon Tong and Edward Y. Chang. Support vector machine active learning for image retrieval. In *ACM Multimedia*, 2001. 17

[60] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. *arXiv preprint arXiv:2010.04595*, 2020. 2

[61] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2897–2905, 2018. 2, 5

[62] Shubham Tulsiani, Abhishek Kar, Joao Carreira, and Jitendra Malik. Learning category-specific deformable 3D models for object reconstruction. *PAMI*, 39(4):719–731, 2017. 1

[63] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017. 2

[64] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proc. CVPR*, pages 209–217, 2017. 4

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proc. NeurIPS*, 2017. 2, 6

[66] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 4

[67] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018. 2, 6, 7, 15

[68] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. *arXiv*, 2021. 2

[69] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020. 4, 7, 16

[70] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Proc. NeurIPS*, 2016. 5

[71] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2020. 2

[72] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond PASCAL: A benchmark for 3D object detection in the wild. In *Proc. WACV*, 2014. 2

[73] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4541–4550, 2019. 2

[74] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Proc. NIPS*, 2020. 2, 4, 6, 7, 13, 15

[75] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. *arXiv*, 2020. 2, 4, 5

[76] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. CVPR*, 2018. 7

[77] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. *arXiv preprint arXiv:2010.09125*, 2020. 2