Cuboids Revisited: Learning Robust 3D Shape Fitting to Single RGB Images Supplementary Material

Florian Kluger[1], Hanno Ackermann[1], Eric Brachmann[2], Michael Ying Yang[3], Bodo Rosenhahn[1]
[1]Leibniz University Hannover, [2]Niantic, [3]University of Twente

Appendix

We provide additional implementation details for our method, including a listing of hyperparameters (Tab. 1), in Sec. A of this appendix. In Sec. B, we establish the occlusion-aware distance metric for superquadrics, which we need for our quantitative evaluation. Sec. C provides an additional ablation study analysing the effectiveness of our proposed occlusion-aware inlier counting (cf. Sec. 3.2.2 in the main paper). In Sec. D, we show additional qualitative results, including failure cases of our method.

A. Implementation Details

A.1. Feature Extraction Network

We employ the *Big-to-Small* (BTS) [10] depth estimation CNN as our feature extraction network. We use a variant of their approach using a DenseNet-161 [5] as the base network. It is pre-trained on NYU Depth v2 [14] and achieves state-of-the-art results for monocular depth estimation. Please refer to [10] for details.

A.2. Sampling Weight Network

For prediction of sampling weights, we use a neural network based on the architecture for scene coordinate regression used in [2]. Refer to Fig. 1 for an overview. The input of the network is a concatenation of features $\mathcal{Y}$ and state $\mathbf{s}$ of size $H \times W \times 2$, with image width W and height H. When using ground truth depth as input, we normalise $\mathcal{Y}$ by the mean and variance of the training set. When using the features predicted by the feature extraction network, we add a batch normalisation [6] layer between the two networks in order to take care of input normalisation. We augmented the network of [2] with instance normalisation [16] layers, which proved crucial for the ability of the network to segment distinct structures in the scene. The network thus consists of 3×3 and 1×1 convolutional layers, instance normalisation layers [16], and ReLU activations [3] arranged as residual blocks [4]. While most convolutions are applied with stride one, layers two to four use a stride of two, resulting in a final output spatially subsampled by a factor of eight w.r.t. the input. We apply sigmoid activation to the last convolutional layer to predict the sampling weight sets $\mathcal{Q}(\mathcal{Y}|\mathcal{M};\mathbf{w})$ with size $\frac{H}{8} \times \frac{W}{8} \times Q$, *i.e.* Q sets of sampling weights for each input. Additionally, we apply global average pooling and a fully-connected layer with sigmoid activation to the output of the penultimate convolutional layer in order to predict the selection weights $\mathbf{q}$.

			depth input	RGB input
training	learning rate		10^{-5}	10^{-6}
	epochs		20	25
	number of instances	$\lvert\mathcal{M}\rvert$	6	4
	batch size	B	2	
	IMR weight	κ_{im}	10^{-2}	
	correlation weight	κ_{corr}	1.0	
	entropy weight	κ_{entropy}	1.0	
	single-instance samples	$\lvert\mathcal{H}\rvert$	32	
	sample count	K	2	
both	inlier threshold	τ	0.004	
	sampling weight sets	Q	4	
	f_h iterations		50	
	f_h learning rate		0.2	
test	number of instances	$\lvert\mathcal{M}\rvert$	6	
	single-instance samples	$\lvert\mathcal{H}\rvert$	4096	
	inlier cutoff (selection)	Θ	10	

Table 1: **User definable parameters** of our approach and the values we chose for our experiments using either ground truth depth or RGB images as input. We distinguish between values used during training and at test time. Mathematical symbols refer to the notation used either in the main paper or in this supplementary document.

A.3. Neural Network Training

We implement our method using PyTorch [13] version 1.7.0. We use the Adam [8] optimiser to train the neural networks. In order to avoid divergence induced by bad hypothesis samples frequently occurring at the beginning of training, we clamp losses to an absolute maximum value of

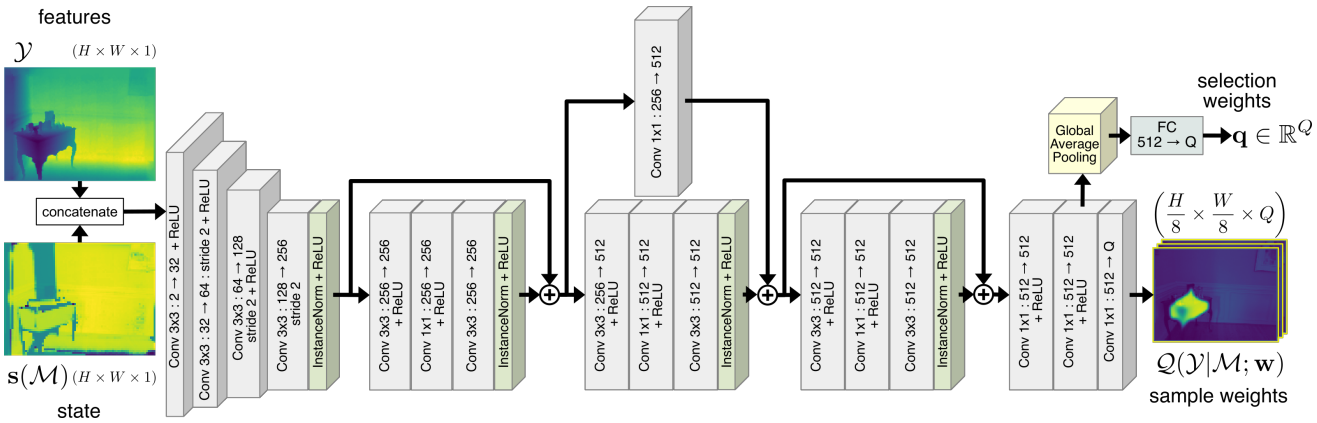


Figure 1: **Sampling Weight Network Architecture:** We stack features $\mathcal{Y}$ and state $\mathbf{s}$ into a tensor of size $H \times W \times 2$, with image width W and height H. We feed this tensor into a neural network consisting of 3×3 and 1×1 convolutional layers, instance normalisation layers [16], and ReLU activations [3] arranged as residual blocks [4]. The last convolutional layer predicts the sampling weight sets $\mathcal{Q}(\mathcal{Y}|\mathcal{M};\mathbf{w})$. We apply global average pooling and a fully-connected layer to the output of the penultimate convolutional layer in order to predict the selection weights $\mathbf{q}$. This architecture is based on [9, 2].

0.3. First, we train the sampling weight network by itself using ground truth depth for 20 epochs with a learning rate of 10^{-5}. Then we fine-tune the sampling weight network and the feature extraction network with RGB input together for 25 epochs with a learning rate of 10^{-6}. We use a batch size $B = 2$ for all experiments. Training was performed using two RTX 2080 Ti GPUs.

Regularisation. In addition to the main task loss $\ell(\mathbf{h}, \mathcal{M})$, we apply the regularisation losses ℓ_{corr} and ℓ_{entropy} in order to prevent mode collapse of the sampling weight sets $\mathcal{Q}$ (cf. Sec. 3.5 in the main paper). We furthermore inherit the inlier masking regularisation (IMR) loss ℓ_{im} from [9]. The final loss ℓ_{final} is thus a weighted sum of all these losses:

$$\ell_{\text{final}} = \ell + \kappa_{\text{corr}} \cdot \ell_{\text{corr}} + \kappa_{\text{entropy}} \cdot \ell_{\text{entropy}} + \kappa_{\text{im}} \cdot \ell_{\text{im}} \,. \quad (1)$$

A.4. Occlusion Detection

In order to detect whether a cuboid $\mathbf{h}$ occludes a point $\mathbf{y}$ from the perspective of a camera with centre $\mathbf{c}$, we must first translate $\mathbf{c}$ into the cuboid-centric coordinate frame. Without loss of generality, we assume $\mathbf{c} = (0, 0, 0)^\mathsf{T}$ and thus:

$$\hat{\mathbf{c}} = \mathbf{R}(\mathbf{c} - \mathbf{t}) = -\mathbf{R}\mathbf{t} \,. \quad (2)$$

We parametrise the line of sight for a point $\mathbf{y}$ as:

$$\mathbf{x}(\lambda) = \hat{\mathbf{y}} + \lambda \mathbf{v} \,, \text{ with } \mathbf{v} = \hat{\mathbf{c}} - \hat{\mathbf{y}} \,. \quad (3)$$

We determine its intersections with each of the six cuboid planes. For the plane orthogonal to the x-axis in its positive direction, this implies that

$$\lambda = \frac{a_x - \hat{\mathbf{y}}^\mathsf{T}\mathbf{e}_x}{\mathbf{v}^\mathsf{T}\mathbf{e}_x}, \quad (4)$$

where $\mathbf{e}_x$ denotes the x-axis unit vector. If $\lambda < 0$, $\mathbf{y}$ lies in front of the plane and is thus not occluded. Otherwise, we must check whether the intersection actually lies on the cuboid, *i.e.* $d(\mathbf{h}, \mathbf{x}(\lambda)) = 0$. If that is the case, then $\mathbf{y}$ is occluded by this part of the cuboid. We repeat this check accordingly for the other five cuboid planes. Via this procedure we define an indicator function:

$$\chi_{\text{o}}(\mathbf{y}, \mathbf{h}, i) = \begin{cases} 1 & \text{if } i\text{-th plane of } \mathbf{h} \text{ occludes } \mathbf{y}, \\ 0 & \text{else , with } i \in \{1, \ldots, 6\} \,. \end{cases} \quad (5)$$

A.5. Cuboid Fitting

We implement the minimal solver $\mathbf{h} = f_h(\mathcal{S})$, which estimates cuboid parameters $\mathbf{h}$ from a minimal set of features $\mathcal{S} = \{\mathbf{y}_1, \ldots, \mathbf{y}_C\}$, via iterative numerical optimisation using gradient descent. To this end, we apply the Adam [8] optimiser to minimise the objective $\frac{1}{|\mathcal{S}|}\|F(\mathcal{S}, \mathbf{h})\|_1$ (cf. Sec. 3.3 in the main paper). We perform 50 steps of gradient descent with a learning rate of 0.2, starting from an initial estimate $\mathbf{h}_0$.

Initialisation of the Minimal Solver. As good initialisation is crucial for fast convergence, we estimate the initial position of the cuboid via the mean of the features, *i.e.* $\mathbf{t}_0 = \frac{1}{|\mathcal{S}|}\sum_{\mathbf{y} \in \mathcal{S}} \mathbf{y}$. Secondly we estimate the rotation $\mathbf{R}$ via singular value decomposition:

$$\mathbf{R}_0 = \mathbf{V}^\mathsf{T}, \text{ with } \mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T} = \begin{bmatrix}\mathbf{y}_1 \ldots \mathbf{y}_{|\mathcal{S}|}\end{bmatrix}^\mathsf{T} . \quad (6)$$

Lastly, we initialise the cuboid size with the element-wise maximum of the absolute coordinates of the centred and rotated features, with $i \in \{1, \ldots, |\mathcal{S}|\}$:

$$\begin{pmatrix} a_{x0} \\ a_{y0} \\ a_{z0} \end{pmatrix} = \begin{pmatrix} \max_i |\hat{x}_i| \\ \max_i |\hat{y}_i| \\ \max_i |\hat{z}_i| \end{pmatrix} \text{ , with } \begin{pmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \end{pmatrix} = \mathbf{R}_0(\mathbf{y}_i - \mathbf{t}_0) \,.$$

This gives us an initial estimate $\mathbf{h}_0 = (a_{x0}, a_{y0}, a_{z0}, \mathbf{R}_0, \mathbf{t}_0)$.

A.6. Stopping Criterion

During evaluation, we always predict up to $|\mathcal{M}|$ cuboids. However, similar to [9], we select primitive instances sequentially. The set $\mathcal{M}$ contains the already selected primitives, and $\mathbf{h}$ is the next selected primitive. We only add $\mathbf{h}$ to $\mathcal{M}$ is it increases the joint inlier count by at least Θ, *i.e.*:

$$I_{\text{c}}(\mathcal{Y}, \mathcal{M} \cup \{\mathbf{h}\}) - I_{\text{c}}(\mathcal{Y}, \mathcal{M}) > \Theta\,. \tag{7}$$

Otherwise the method terminates and returns $\mathcal{M}$ as the recovered primitive configuration.

B. OA Distance for Superquadrics

The surface of a superellipsoid [1], which is the type of superquadric used in [12, 11], can be described by its inside-outside function:

$$f_{\text{sq}}(x,y,z) = \left(\left(\frac{x}{a_x}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_y}\right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_1}{\epsilon_2}} + \left(\frac{z}{a_y}\right)^{\frac{2}{\epsilon_1}} - 1\,, \tag{8}$$

with ϵ_1, ϵ_2 describing the shape of the superquadric, and a_x, a_y, a_z describing its extent along the canonical axes. If $f_{\text{sq}} = 0$, the point (x, y, z) resides on the superquadric surface. For $f_{\text{sq}} > 0$ and $f_{\text{sq}} < 0$, it is outside or inside the superquadric, respectively. Alternatively, a point on the superquadric surface can be described by:

$$\mathbf{p}(\eta, \omega) = \begin{bmatrix} a_x \cos^{\epsilon_1}(\eta) \cos^{\epsilon_2}(\omega) \\ a_y \cos^{\epsilon_1}(\eta) \sin^{\epsilon_2}(\omega) \\ a_z \sin^{\epsilon_1}(\eta) \end{bmatrix}\,, \tag{9}$$

parametrised by two angles η, ω. The surface normal at such a point is defined as:

$$\mathbf{n}(\eta, \omega) = \begin{bmatrix} a_x^{-1} \cos^{2-\epsilon_1}(\eta) \cos^{2-\epsilon_2}(\omega) \\ a_y^{-1} \cos^{2-\epsilon_1}(\eta) \sin^{2-\epsilon_2}(\omega) \\ a_z^{-1} \sin^{2-\epsilon_1}(\eta) \end{bmatrix}\,. \tag{10}$$

Unfortunately, no closed form solution exists for calculating a point-to-superquadric distance, which we would need in order to compute the occlusion-aware distance metric as described in Sec. 3.2. We therefore approximate it by sampling points and determining occlusion and self-occlusion using Eq. 9 and Eq. 10.

B.1. Occlusion

Given a 3D point $\mathbf{y} = [x,\, y,\, z]^{\mathsf{T}}$ and a camera centre $\mathbf{c} = [0,\, 0,\, 0]^{\mathsf{T}}$, we sample L points uniformly on the line of sight:

$$\mathcal{Y}_{\text{los}} = \{\frac{1}{L}\mathbf{y},\, \frac{2}{L}\mathbf{y},\, \ldots, \mathbf{y}\} = \{\mathbf{y}_1,\, \mathbf{y}_2,\, \ldots,\, \mathbf{y}_L\}\,. \tag{11}$$

For a superquadric $\mathbf{h} = (\epsilon_1, \epsilon_2, a_x, a_y, a_z, \mathbf{R}, \mathbf{t})$, we transform all points in $\mathcal{Y}_{\text{los}}$ into the superquadric-centric coordinate system:

$$\hat{\mathbf{y}} = \mathbf{R}(\mathbf{y} - \mathbf{t})\,, \tag{12}$$

and determine whether they are inside or outside of the superquadric:

$$\mathcal{F}_{\text{los}} = \{f_{\text{sq}}(\hat{\mathbf{y}}_1), \ldots, f_{\text{sq}}(\hat{\mathbf{y}}_L)\}\,. \tag{13}$$

We then count the sign changes in $\mathcal{F}_{\text{los}}$: if there are none, $\mathbf{y}$ is not occluded by the superquadric; otherwise, it is:

$$\chi_{\text{sq}}(\mathbf{y}, \mathbf{h}) = \begin{cases} 1 & \text{if } \mathbf{h} \text{ occludes } \mathbf{y}, \\ 0 & \text{else }. \end{cases} \tag{14}$$

B.2. Self-Occlusion

Using the source code provided by the authors of [12], we sample N points $\mathbf{p}_i$ uniformly on the surface of superquadric $\mathbf{h}$ and determine their corresponding surface normals $\mathbf{n}_i$. For each point, we compute the vector $\mathbf{v}_i = \mathbf{p}_i - \hat{\mathbf{c}}$, with $\hat{\mathbf{c}} = \mathbf{R}(\mathbf{c} - \mathbf{t})$ being the camera centre in the superquadric-centric coordinate system. If $\mathbf{v}_i^{\mathsf{T}}\mathbf{n}_i = 0$, *i.e.* the two vectors are orthogonal, $\mathbf{p}_i$ lies on the rim of the superquadric, which partitions it into a visible and an invisible part [7]. Assuming $\mathbf{n}_i$ points outward, it follows that $\mathbf{p}_i$ is invisible if $\mathbf{v}_i^{\mathsf{T}}\mathbf{n}_i > 0$ and $f_{\text{sq}}(\hat{\mathbf{c}}) > 0$, in which case we discard it. We denote the set of remaining visible points of $\mathbf{h}$ as $\mathcal{P}(\mathbf{h})$.

B.3. Occlusion-Aware Distance

We define the distance of a point $\mathbf{y}$ to a single superquadric $\mathbf{h}$ as the minimum distance to any of its visible points $\mathcal{P}$:

$$d_{\text{sq}}(\mathbf{h}, \mathbf{y}) = \min_{\mathbf{p} \in \mathcal{P}(\mathbf{h})} \|\mathbf{p} - \mathbf{y}\|_2\,. \tag{15}$$

Similarly to cuboids, we compute the distance of $\mathbf{y}$ to the most distant occluding superquadric, given a set of superquadrics $\mathcal{M}$:

$$d_{\text{o,sq}}(\mathcal{M}, \mathbf{y}) = \max_{\mathbf{h} \in \mathcal{M}} \left(\chi_{\text{sq}}(\mathbf{y}, \mathbf{h}) \cdot d_{\text{sq}}(\mathbf{h}, \mathbf{y}) \right)\,, \tag{16}$$

and corresponding occlusion-aware distance:

$$d_{\text{oa,sq}}(\mathcal{M}, \mathbf{y}) = \max\left(\min_{\mathbf{h} \in \mathcal{M}} d_{\text{sq}}(\mathbf{h}, \mathbf{y}),\; d_{\text{o,sq}}(\mathcal{M}, \mathbf{y}) \right)\,. \tag{17}$$

C. Ablation Study

In order to demonstrate the impact of our proposed occlusion-aware (OA) inlier counting (cf. Sec. 3.4 in the main paper), we selectively enabled or disabled the following parts of our method:

1. During training: occlusion-aware inlier counting for sampling and hypothesis selection.

2. During training: occlusion-aware inlier counting for loss computation (cf. Sec. 3.5 main paper).

3. During inference: occlusion-aware inlier counting for sampling and hypothesis selection.

When disabling the occlusion-aware inlier counting for one component, we used regular inlier counting based on the minimal L_2 distance instead. We then evaluated these variants using ground truth depth input on the NYU [14] test set, and show the results in Tab. 2.

As these results show, performance w.r.t. the occlusion-aware L_2 distance degrades severely when disabling the occlusion-aware inlier counting during inference, regardless of whether it was used during training (cf. rows 5-8 in Tab. 2). AUC percentages drop to single digits for smaller upper bounds ($20\,\mathrm{cm}$ and below), and the mean increases from around $20\,\mathrm{cm}$ to more than $1.5\,\mathrm{m}$. Conversely, the mean of the regular L_2 distance decreases by roughly a factor of four. This indicates that the recovered cuboids cover more points, but create significantly more occlusions doing so, which results in less reasonable scene abstractions. However, when we look at the impact of occlusion-aware inlier counting during training (cf. rows 1-4 in Tab. 2), the differences are not as clear-cut. Using occlusion-aware inlier counting for both sampling and loss during training still yields the highest AUC values, albeit with slimmer margins, between 0.2 and 4.8 percentage points. Disabling occlusion-aware inlier counting for the loss only (row 3) yields the second best AUC values, with the smallest margin at $\mathrm{AUC}_{@50\,\mathrm{cm}}$ and the largest margin at $\mathrm{AUC}_{@10\,\mathrm{cm}}$. Meanwhile, this configuration yields the lowest mean occlusion-aware distance – $18.9\,\mathrm{cm}$ vs. $20.8\,\mathrm{cm}$, *i.e.* 9.1% less. This indicates fewer outlier points with errors $> 50\,\mathrm{cm}$, which strongly influence the mean error, but lower representation accuracy for inlier points with errors $< 50\,\mathrm{cm}$.

D. Qualitative Results

D.1. Occlusion-Aware Inlier Counting

Complementing the ablation study in Sec. C, we compare example results with and without occlusion-aware inlier counting during inference in Fig. 3. As in Sec. C, results are based on ground truth depth input for the NYU dataset. As these examples show, the occlusion-aware inlier counting enables our method to reasonably abstract key elements of the scenes. Without occlusion-aware inlier counting, however, the method predicts cuboids which are too large and often intersecting, occluding most of the scene. The $\mathrm{AUC}_{@5\,\mathrm{cm}}$ values, which we also provide for each example in the figure, emphasise this observation.

D.2. Failure Cases

We show failure cases of our method, *i.e.* examples with below average AUC, in Fig. 2. Most common failure modes of our approach are:

- Missing scene parts, for which no cuboid was fitted (first and third column).

- Cuboids which are too large, too small, improperly oriented, or simply too coarse (second column).

- Spurious cuboids, usually very thin and barely visible form the original view (second and third column).

We conjecture that the first two failure modes can be mitigated via more effective sampling. The third failure mode may require additional consideration in the inlier counting procedure, or possibly just an increased instance cutoff threshold (cf. Sec. A.6).

D.3. Cuboid Parsing Baseline

As mentioned in the main paper, we were unable to obtain sensible results with the cuboid based approach of [15] for the NYU dataset. We trained their approach on NYU using ground truth depth input, following their instructions published together with their source code, using the same input data as we did for the superquadrics approach of [12], multiple times with different random seeds, but to no avail. Their approach mostly predicts the same or very similar cuboid configurations for different images. Often, no cuboids are recovered at all. We show a couple of examples for three different training runs in Fig. 4.

References

[1] Alan H Barr. Superquadrics and angle-preserving transformations. *CGA*, 1981. 3

[2] Eric Brachmann and Carsten Rother. Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses. In *ICCV*, 2019. 1, 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015. 1, 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2

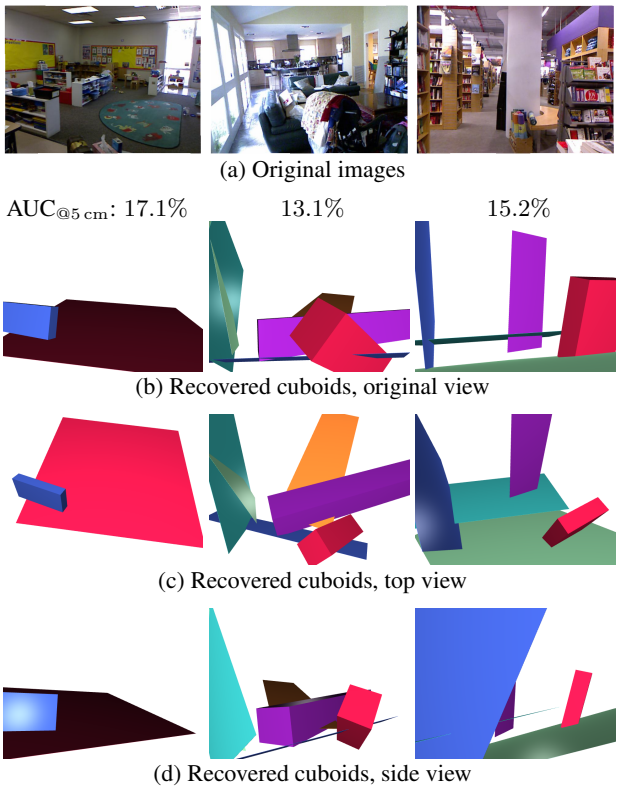
[5] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1

[7] Ales Jaklic, Ales Leonardis, Franc Solina, and F Solina. *Segmentation and recovery of superquadrics*. Springer Science & Business Media, 2000. 3

occlusion-aware inlier			occlusion-aware L_2-distance								L_2	
training sampling	training loss	inference sampling	$\text{AUC}_{@50\,\text{cm}}$		$\text{AUC}_{@20\,\text{cm}}$		$\text{AUC}_{@10\,\text{cm}}$		$\text{AUC}_{@5\,\text{cm}}$		mean (cm)		mean (cm)	
✓	✓	✓	**77.2%**	±0.08	**62.7%**	±0.07	**49.1%**	±0.12	**34.3%**	±0.14	20.8	±0.51	17.9	±0.50
✗	✓	✓	75.4%	±0.13	58.8%	±0.19	44.3%	±0.22	29.8%	±0.23	20.1	±0.43	16.6	±0.47
✓	✗	✓	77.0%	±0.11	61.8%	±0.08	47.9%	±0.08	33.1%	±0.12	**18.9**	±0.19	15.5	±0.17
✗	✗	✓	76.6%	±0.09	61.2%	±0.10	47.1%	±0.11	32.4%	±0.13	19.0	±0.37	15.5	±0.42
✓	✓	✗	11.0%	±0.29	5.3%	±0.20	3.3%	±0.11	2.0%	±0.06	151.7	±1.31	4.4	±0.08
✗	✓	✗	11.1%	±0.14	5.4%	±0.08	3.3%	±0.06	1.9%	±0.04	152.7	±0.49	4.5	±0.07
✓	✗	✗	10.3%	±0.31	5.0%	±0.16	3.1%	±0.12	1.8%	±0.08	156.4	±1.43	4.2	±0.06
✗	✗	✗	10.0%	±0.10	4.7%	±0.11	2.9%	±0.08	1.7%	±0.06	157.4	±1.25	**3.9**	±0.08

Table 2: **Ablation Study:** We analyse the influence of our occlusion-aware inlier counting. We enable or disable it for hypothesis sampling and selection during training, for loss computation during training, and for hypothesis sampling and selection during inference. We evaluate on NYU Depth v2 [14] for depth input. We present AUC values (higher is better) for various upper bounds of the occlusion-aware (OA) L_2 distance . We also report mean OA-L_2 and regular L_2 distances (lower is better). See Sec. C for details.



(a) Original images

(b) Recovered cuboids, original view

(c) Recovered cuboids, top view

(d) Recovered cuboids, side view

Figure 2: **Failure Cases.** First row: Original images from the NYU dataset. Rows (b)-(d): cuboids recovered from ground truth depth using our method, showing views from the perspective of the original image, as well as top and side views. We also report the $\text{AUC}_{@5\,\text{cm}}$ values for each example above row (b). See Sec. D.2 for details.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 1, 2

[9] Florian Kluger, Eric Brachmann, Hanno Ackermann, Carsten Rother, Michael Ying Yang, and Bodo Rosenhahn. CONSAC: Robust Multi-Model Fitting by Conditional Sample Consensus. In *CVPR*, 2020. 2, 3

[10] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation. *arXiv preprint arXiv:1907.10326*, 2019. 1

[11] Despoina Paschalidou, Luc Van Gool, and Andreas Geiger. Learning Unsupervised Hierarchical Part Decomposition of 3D Objects from a Single RGB Image. In *CVPR*, 2020. 3

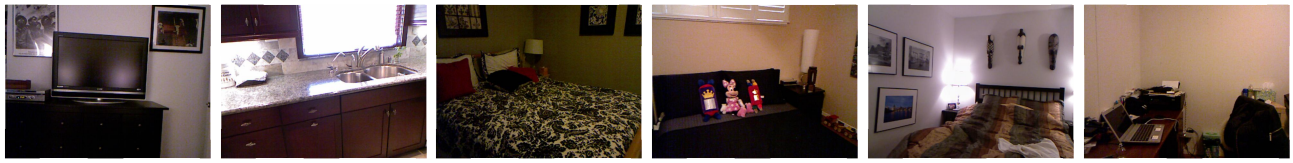
[12] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics Revisited: Learning 3D Shape Parsing Beyond Cuboids. In *CVPR*, 2019. 3, 4

[13] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017. 1

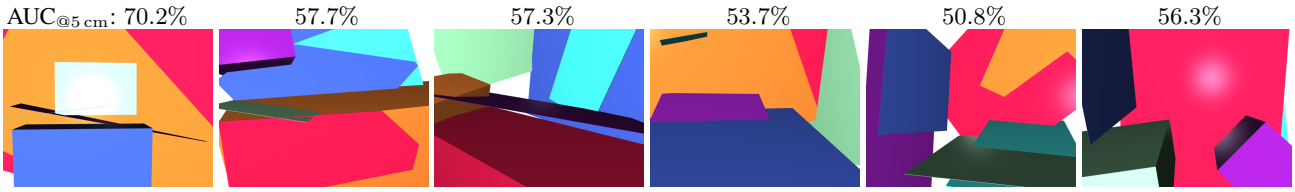
[14] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012. 1, 4, 5

[15] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning Shape Abstractions by Assembling Volumetric Primitives. In *CVPR*, 2017. 4, 7

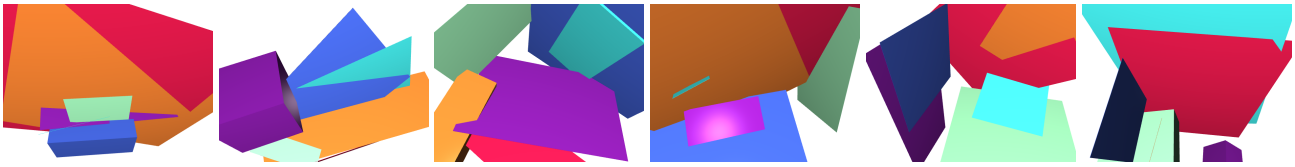
[16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. In *CoRR*, 2016. 1, 2



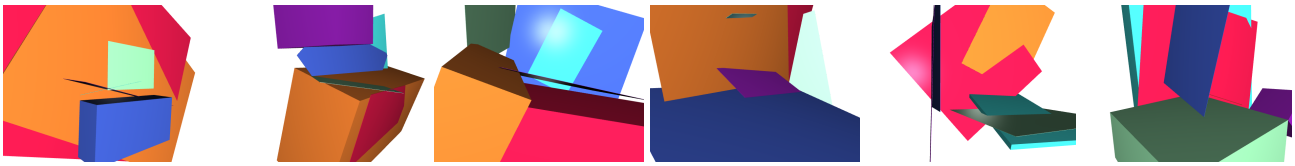
(a) Original images



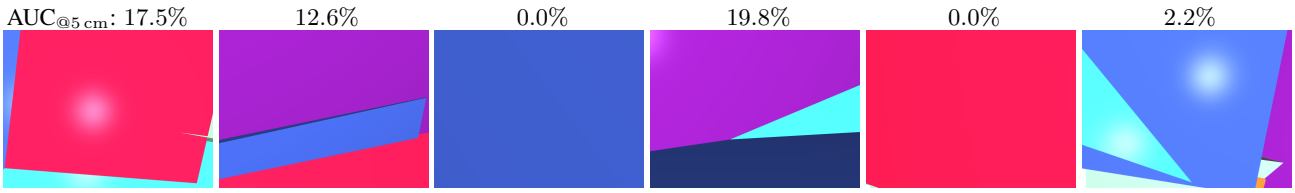
(b) Recovered cuboids using occlusion-aware inlier counting, original view



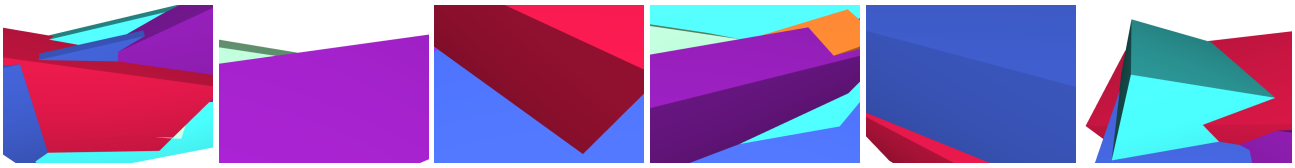
(c) Recovered cuboids using occlusion-aware inlier counting, top view



(d) Recovered cuboids using occlusion-aware inlier counting, side view



(e) Recovered cuboids without occlusion-aware inlier counting, original view



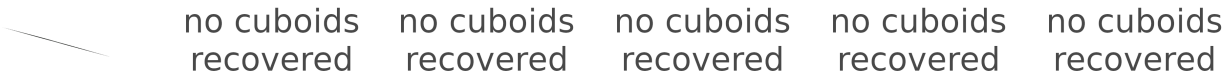
(f) Recovered cuboids without occlusion-aware inlier counting, top view

(g) Recovered cuboids without occlusion-aware inlier counting, side view

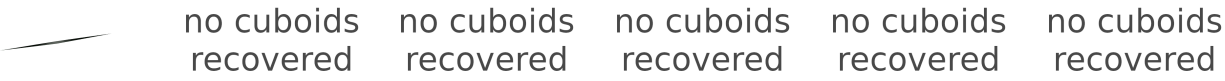
Figure 3: **Occlusion-Aware Inlier Counting.** First row: Original images from the NYU dataset. Rows (b)-(d): cuboids recovered from ground truth depth using our method with occlusion-aware inlier counting during inference. Rows (e)-(g): results *without* occlusion-aware inlier counting. We show views from the perspective of the original image, as well as top and side views. We also report the $\text{AUC}_{@5\,\text{cm}}$ values for each example above rows (b) and (e). See Sec. D.1 for details.

(a) Original images

(b) Recovered cuboids using [15], first training run



(c) Recovered cuboids using [15], second training run



(d) Recovered cuboids using [15], third training run

Figure 4: **Cuboid Parsing Baseline:** We show qualitative results for the cuboid based approach of [15]. Row (a) shows the original images from the NYU dataset. Rows (b)-(d) show corresponding cuboid predictions by [15] for three different training runs using ground truth depth as input. As these examples show, the method is unable to recover sensible cuboid configurations for these real-world indoor scenes. See Sec. D.3 for details.