FvOR: Robust Joint Shape and Pose Optimization for Few-view Object Reconstruction Supplementary Materials

Zhenpei Yang[1] Zhile Ren[2] Miguel Angel Bautista[2]
Zaiwei Zhang[1] Qi Shan[2] Qixing Huang[1]
[1]The University of Texas at Austin [2]Apple

1. Technical Details

1.1. Pose Initialization Approaches

FvOR (Ours). We first use a modified ResNet18 to extract the per-view feature, then those coarse feature maps (at $\frac{1}{8}$ input resolution) are fed into the cross-frame attention module to perform both self-attention and cross-frame attention. The resulting feature maps are reshaped and further decoded into per-pixel scene coordinate prediction. For each image, we randomly sample 1000 scene coordinates and employ OpenCV's implementation of PnP+RANSAC [2] with 5000 iterations to get our pose estimation. A diagram of our pose initialization module is in Fig. 1.

Extreme-Rot. The coarse feature map is passed to Extreme-Rot's prediction module [3] to predict a distribution of the underlying quantity. We use 6 separate prediction modules for 3 Euler angles and 3 translations along each axis respectively.

1.2. Shape Optimization Module

Implementation Details. We use a modified version of ResNet-18(with up-convolution layers and skip connections) to serve as our feature backbone. It takes input of an image $I_i \in \mathcal{R}^{3\times h\times w}$ and produce a feature map $F_i \in \mathcal{R}^{c\times h\times w}$. The 3D Convolutional U-Net consists of 6 down-sampling layers that gradually reduce the spatial resolution and then 6 up-sampling layers plus skip connections to recover the input resolution. The network illustration can be found in Fig. 2.

1.3. Pose Optimization Module

A diagram of the pose optimization network is in Fig. 3. Since the core of our pose optimization module is aligning current geometry with input images, it needs a reasonably good initial geometry to start with. Hence, we pre-train the shape module with GT Pose. Then we simulate noisy input geometry (as we will encounter during the test time) by adding Noise@L1 to the input pose of the shape module. We generate this type of geometry on the fly, and we train the pose refinement module to recover the GT pose from Noise@L3 perturbation. Note that the shape module is also fine-tuned (with noisy pose) at this stage. However, our training of the shape module and pose module is not end-to-end (as there is no gradient back-propagation from the pose module to the shape module). We use a smaller noise level for the shape module as we need it to generate a highly accurate shape, and we found increasing the training noise level will hurt its accuracy.

1.4. Alternating Shape Update and Pose Update

As shown in Algorithm 1 of the main paper, our algorithm consists of an outer loop and an inner loop. We alternate between shape and pose update in the outer loop, and update the pose with fixed shapes in the inner loop. We use 3 outer loop iterations, and 5 inner loop iterations during the test time. In addition to $\mathcal{L}_p$ (defined in Eq 4 of the main paper), we also added a regularization term $\mathcal{L}_r$:

$$\mathcal{L}_{\text{r}} = \lambda_{\text{reg}} \sum_i \left\| \begin{pmatrix} \hat{R}_i & \hat{t}_i \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} R_i^0 & t_i^0 \\ 0 & 1 \end{pmatrix} \right\|_{\mathcal{F}}^2,$$

where $\hat{R}_i$,$\hat{t}_i$ and R_i,t_i are the current pose estimate and GT pose for the i^{th} image respectively. This term prevents the pose drift from the initial pose in future iterations.

1.5. Evaluation Details

As discussed in the main paper, the commonly used evaluation in previous works [1,6] does not factor out the alignment factor, which will not reflect shape quality faithfully. Thus, we propose to factor out the alignment before calculating *IoU*, *Chamfer-L1* and *Normal-Consistency* scores. To factor out the alignment factor, we propose to locally align the prediction and GT to solve for a scale factor s, and rigid transformation $T = (R|t)$. Taking advantage of the implicit representation of the prediction(all evaluated methods in the table use implicit representation), we minimize the following loss using PyTorch's Adam [4] optimizer:

$$\min_{s,R,t} \sum_{x_i} |g(s \cdot (Rx_i + t))|, \tag{1}$$

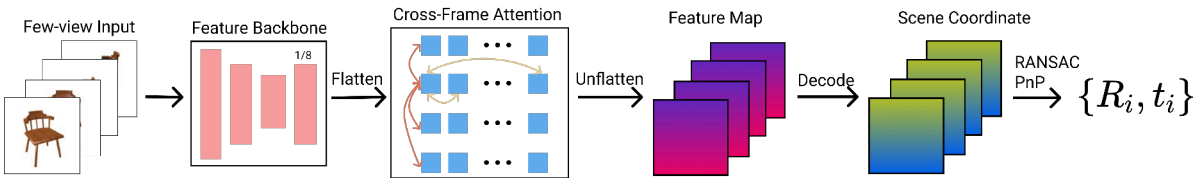

Figure 1. A diagram of the pose initialization network architecture of FvOR. We use the predicted scene coordinate to estimate the pose.

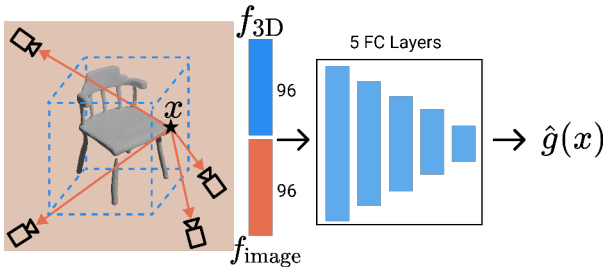

Figure 2. Shape optimization module network architecture. For each query point x, a pixel-aligned image feature f_{image} is obtained by interpolating each view's feature map. 3D convolutional feature f_{3D} is obtained by interpolating 3D feature volume. The resulting feature vectors are concatenated and pass through 5 fully connected layers to generate the final signed-distance prediction.

where x_i is the surface point sampled from ground truth mesh. In essence, Eq 1 seeks to find a similarity transformation that converts the ground truth mesh onto the zero level set of the prediction.

2. Additional Studies

2.1. Pose Initialization

Per-category Quantitative Results on ShapeNet. We show the per-category pose initialization on ShapeNet in Table 1. Our scene coordinate representation is better than the other representations for most cases.

Different Number of Views. We test how the performance will change when given different number of views during test time. The results can be found in Tab. 3. More views leads to more accurate pose prediction.

2.2. Shape Module

Quantitative. We show the per-category 3D reconstruction results on ShapeNet in Table **??**. Our shape module achieve the best-in-class performance across most categories.

More Analysis on Feature Representation. We show in Table 4 more analysis on feature representation. We tested different kinds of grid resolution used in building 3D convolutional feature. Lower grid resolution significantly decreases the training memory requirement, as well as increase the inference speed. We also found that when the grid resolution is small, then image feature f^{image} have more salient improvement. This is reasonable since in this case the grid are too coarse to capture fine-grained details. In contrast, image feature is not restricted to grid resolution, thus provide complementary benefits with relatively small cost.

Generalization. As in 3D43D [1], we conduct a generalization test on our shape module shown in Table 2. We followed their practice and train our shape module on 3 categories, and test on the rest 10 categories. Our approach achieves significant improvement on the generalization over 3D43D [1]. This further demonstrates our design choice.

Different Number of Views. We test how the performance change when given different number of views during test time. The results can be found in Fig. 5. More input views leads to better results.

2.3. Pose Update Module

Effectiveness of learned feature representation. The pose update module learns a feature representation that is suitable for alignment. We compare using the initial feature representation(before traninig) with using trained feature representation for alignment in Table 4. In this experiment, we use a fixed shape generated with a pre-trained shape module and gt camera pose. And we then perturb the camera pose of each image and run multiple pose update to correct the camera pose. The learned feature representation clearly helps the alignment.

References

[1] Miguel Ángel Bautista, Walter Talbott, Shuangfei Zhai, Nitish Srivastava, and Joshua M. Susskind. On the generalization of learning-based 3D reconstruction. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021. 1, 2, 3, 5

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 1

[3] Ruojin Cai, Bharath Hariharan, Noah Snavely, and Hadar Averbuch-Elor. Extreme rotation estimation using dense correlation volumes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14566–14575, 2021. 1, 3

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[5] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks:

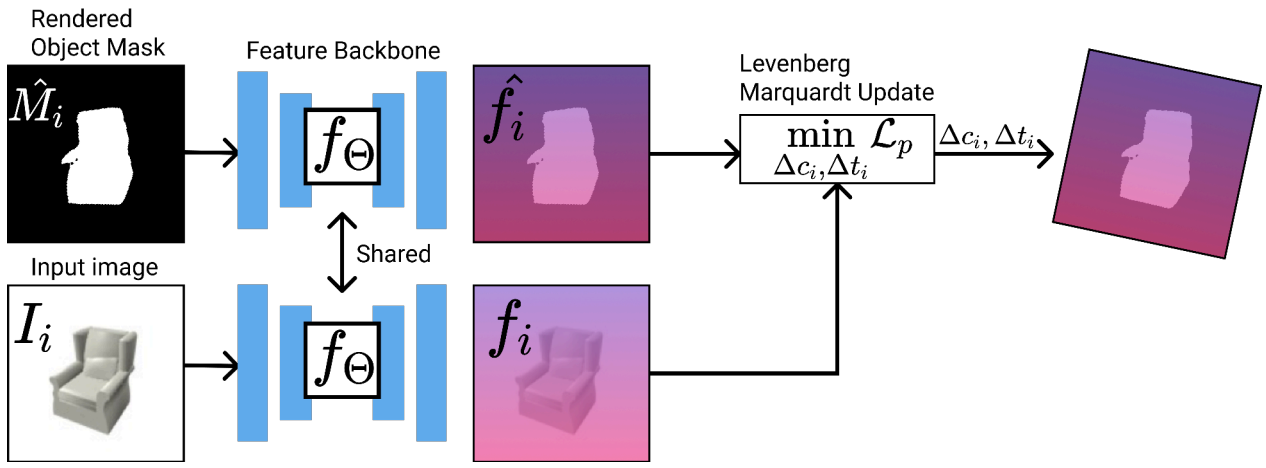

Figure 3. A diagram of the pose optimization module. We use a feature backbone (similar to the one used in our pose initialization network) to extract per-pixel feature map for both rendered object mask and input view. Then we compute the pose update $\{\Delta c_i, \Delta t_i\}$ by aligning these two feature maps.

		plane	bench	cabinet	car	chair	display	lamp	speaker	rifle	sofa	table	phone	boat	mean
pixel error	DISN [7]	2.004/0.992	1.390/0.948	1.461/0.740	0.984/0.828	1.267/0.984	4.560/1.122	3.581/2.517	5.477/1.510	2.457/1.088	1.734/0.893	2.273/0.988	2.927/0.864	1.847/1.251	2.459/1.133
	Cai *et al.* [3]	2.065/1.134	1.359/1.223	2.397/1.270	1.207/1.130	1.453/1.168	2.171/1.220	5.099/1.910	4.403/1.296	1.416/1.110	1.536/1.226	2.301/1.265	1.962/1.227	1.911/1.161	2.252/1.257
	FvOR-Quat	1.903/0.961	2.032/1.231	2.372/1.186	1.143/0.984	1.985/1.404	7.010/1.638	3.692/2.495	7.013/2.491	1.897/0.979	2.331/1.354	3.285/1.434	3.177/1.491	1.950/1.350	3.061/1.461
	FvOR (Ours)	0.748/0.488	0.724/0.406	0.708/0.301	0.530/0.446	0.728/0.533	3.289/0.632	2.464/1.528	3.323/0.820	1.531/0.548	0.831/0.463	1.169/0.350	1.229/0.436	0.969/0.770	**1.403/0.594**
Rot error	DISN [7]	4.682/2.304	2.803/1.552	3.199/1.443	1.806/1.547	2.438/1.796	11.506/2.060	11.456/6.976	11.899/2.749	6.401/2.355	2.793/1.466	4.195/1.596	10.338/2.037	4.654/2.438	6.013/2.332
	Cai *et al.* [3]	3.693/1.004	1.170/0.969	3.011/0.997	1.099/0.934	1.613/0.958	3.311/0.910	8.908/2.434	7.662/1.003	2.653/0.959	1.671/0.914	2.878/0.978	2.649/0.950	3.199/0.958	3.348/**1.075**
	FvOR-Quat	4.192/2.213	3.539/2.125	5.520/2.105	1.985/1.763	3.605/2.440	15.827/2.997	11.560/7.346	14.656/5.340	4.572/2.130	3.699/1.999	5.936/2.117	10.436/3.172	4.695/2.612	6.940/2.951
	FvOR (Ours)	1.667/1.135	1.368/0.699	1.485/0.424	0.829/0.698	1.217/0.831	8.452/0.933	7.661/4.016	6.574/1.412	4.454/1.617	1.277/0.655	2.056/0.450	3.703/0.890	2.289/1.662	**3.310**/1.186
Trans error	DISN [7]	0.024/0.019	0.014/0.012	0.007/0.005	0.013/0.011	0.015/0.011	0.017/0.011	0.022/0.014	0.015/0.011	0.025/0.021	0.012/0.010	0.011/0.007	0.012/0.008	0.024/0.020	0.016/0.012
	Cai *et al.* [3]	0.015/0.009	0.009/0.009	0.010/0.009	0.009/0.009	0.010/0.009	0.011/0.009	0.020/0.012	0.015/0.009	0.012/0.009	0.009/0.009	0.010/0.009	0.011/0.009	0.011/0.009	**0.012/0.009**
	FvOR-Quat	0.028/0.022	0.017/0.015	0.010/0.009	0.016/0.014	0.021/0.016	0.022/0.014	0.025/0.018	0.021/0.015	0.026/0.022	0.017/0.015	0.016/0.011	0.016/0.009	0.030/0.024	0.020/0.016
	FvOR (Ours)	0.017/0.012	0.010/0.008	0.006/0.004	0.010/0.009	0.012/0.009	0.012/0.009	0.024/0.015	0.019/0.011	0.023/0.018	0.011/0.008	0.009/0.005	0.010/0.008	0.020/0.017	0.014/0.010

Table 1. Quantitative results on pose estimation on ShapeNet dataset. We show the mean and median pixel error for each cateogry. The image resolution used here is 224×224

Category	OccNet [5]	OccNet† [1, 5]	3D43D [1]	FvOR w/ GT Pose
bench	0.288/0.508/0.729	0.147/1.960/0.625	0.463/0.113/0.617	0.581/0.00728/0.8783
cabinet	0.295/0.917/0.674	0.312/1.273/0.655	0.629/0.250/0.844	0.619/0.0142/0.871
display	0.120/2.868/0.560	0.127/3.179/0.534	0.409/0.428/0.770	0.597/0.01156/0.910
lamp	0.100/3.365/0.586	0.138/2.653/0.623	0.369/2.057/0.738	0.486/0.0124/0.8332
speaker	0.315/1.460/0.660	0.333/1.344/0.677	0.627/0.392/0.829	0.675/0.0150/0.875
rifle	0.180/1.866/0.567	0.095/2.610/0.444	0.498/0.115/0.760	0.725/0.0044/0.916
sofa	0.525/0.732/0.776	0.356/1.445/0.663	0.679/0.147/0.858	0.779/0.00862/0.920
table	0.186/1.122/0.694	0.177/1.771/0.700	0.455/0.255/0.827	0.555/0.00941/0.893
phone	0.036/1.588/0.689	0.131/1.457/0.592	0.549/0.184/0.861	0.753/0.00859/0.954
boat	0.347/0.683/0.661	0.256/1.524/0.603	0.521/0.145/0.776	0.642/0.0102/0.858
Mean	0.239/1.511/0.660	0.207/1.922/0.612	0.520/0.409/0.806	**0.641/0.010166/0.891**

Table 2. Novel category generalization experiments. Each method is trained on car/chair/plane categories and tested on the rest 10 categories. The numbers in each cell are IoU/Chamfer-L1/F-score. OccNet [5] use single view. The rest methods use 5-views. The last two columns use GT camera poses. The results for OccNet, OccNet† and 3D43D are obtained from [1] thus they do not factor out global similarity metrics. *3D43D* and *FvOR w/ GT Pose* use ground truth camera pose.

Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019. 3, 5

[6] Haozhe Xie, Hongxun Yao, Shengping Zhang, Shangchen Zhou, and Wenxiu Sun. Pix2vox++: multi-scale context-aware 3d object reconstruction from single and multiple images. *International Journal of Computer Vision (IJCV)*, 128(12):2919–2935, 2020. 1

[7] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. DISN: Deep implicit surface network

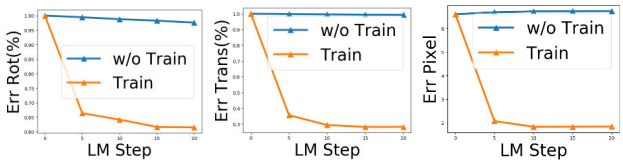

Figure 4. Testing the pose update module in isolation by aligning a fixed shape to images with perturbed camera poses. We show the percentage of rotation and translation error, as well as the pixel error after along the LM steps. After end-to-end training, our pose update module can recover from considerably amount of noise.

	3 Views	5 Views*	7 Views	9 Views
Pixel-Error	1.85/0.63	1.40/0.59	1.29/0.59	**1.25/0.58**
Rotation-Error	4.00/1.21	3.31/1.19	2.87/**1.14**	**2.77/1.14**
Translation-Error	0.017/0.011	**0.014/0.010**	**0.014/0.010**	**0.014/0.010**

Table 3. Analysis of the pose initialization module's performance under different number of testing views on ShapeNet. The model is trained using 5 views. More views gives better pose prediction.

Grid	Metrics	All	w/o f_{image}	w/o f_{3D}
64^3	IoU↑	**0.783**	0.782	0.718
	Chamfer-L1↓	**0.058**	0.060	0.082
	Inference FPS↑	9.7	11.0	**13.8**
	Training Mem(GB)↓	32.1	28.6	**22.1**
32^3	IoU↑	**0.771**	0.765	0.718
	Chamfer-L1↓	**0.063**	0.065	0.082
	Inference FPS↑	11.0	12.6	**13.8**
	Training Mem(GB)↓	24.7	**19.1**	22.0
16^3	IoU↑	**0.754**	0.730	0.720
	Chamfer-L1↓	**0.068**	0.076	0.082
	Inference FPS↑	11.2	12.7	**13.9**
	Training Mem(GB)↓	22.9	**21.0**	22.0

Table 4. More analysis on the effect of 3D convolutional representation and image representation on decoder. We use ShapeNet and ground truth poses in this experiment. The results are averaged across all 13 categories. The training memory are calculated with batch size 12 and image size 224×224.

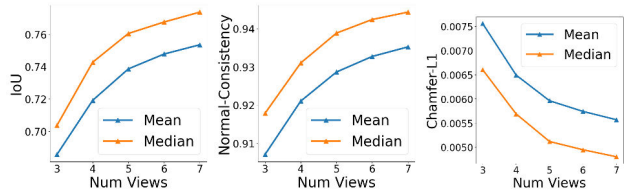

Figure 5. This figure shows how the reconstruction results change w.r.t different number of input views. For each metric we show mean and median result on ShapeNet chair category. We can see our 3D module can be adapted to different number of (few-)views during inference.

for high-quality single-view 3d reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 492–502, 2019. 3

[8] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020. 5

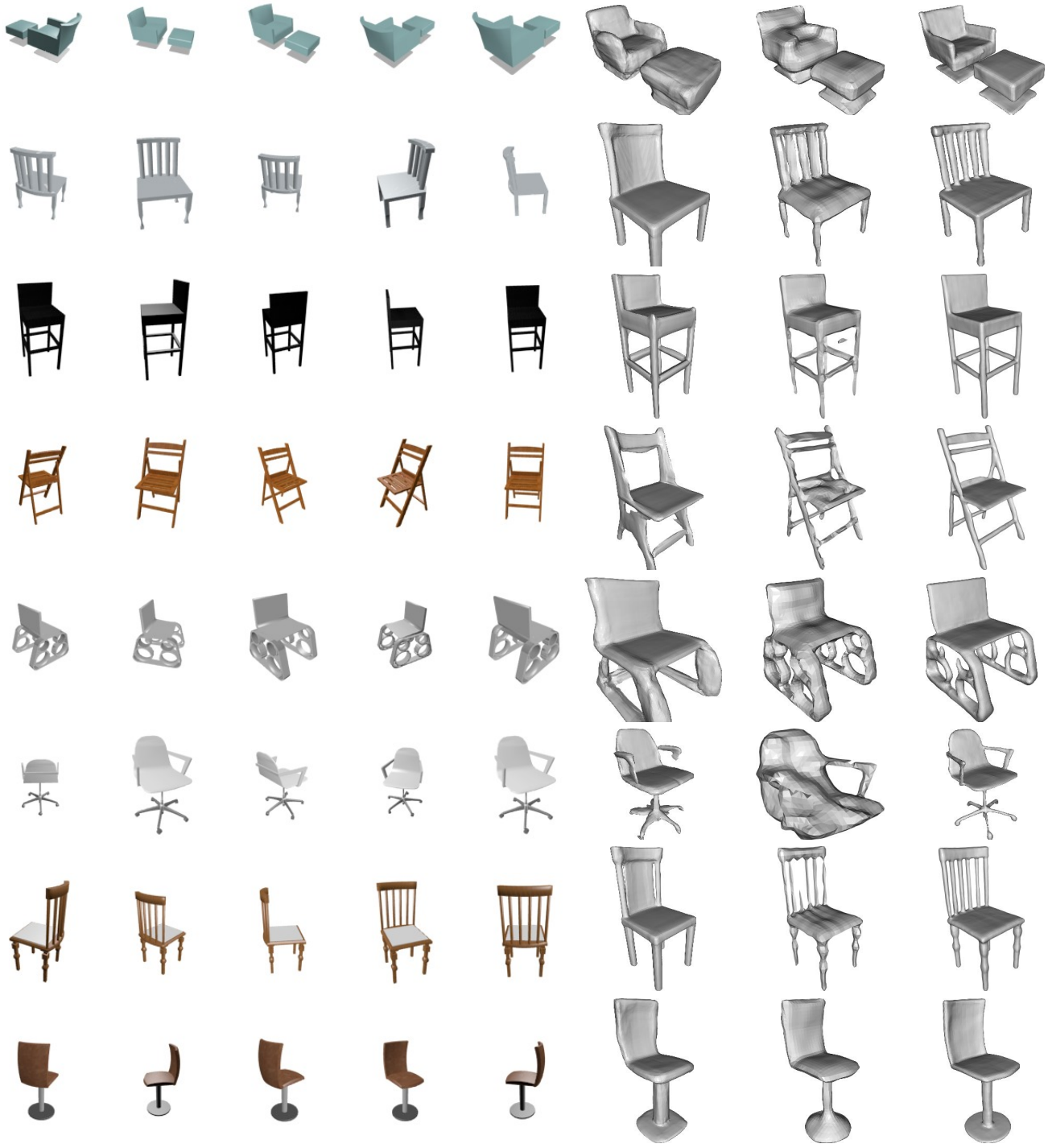

Figure 6. Qualitative comparison on ShapeNet. On the left we show the five input images. On the right we show the prediction of OccNet† [1, 5], IDR [8] and FvOR(ours). IDR and FvOR use our predicted camera poses.

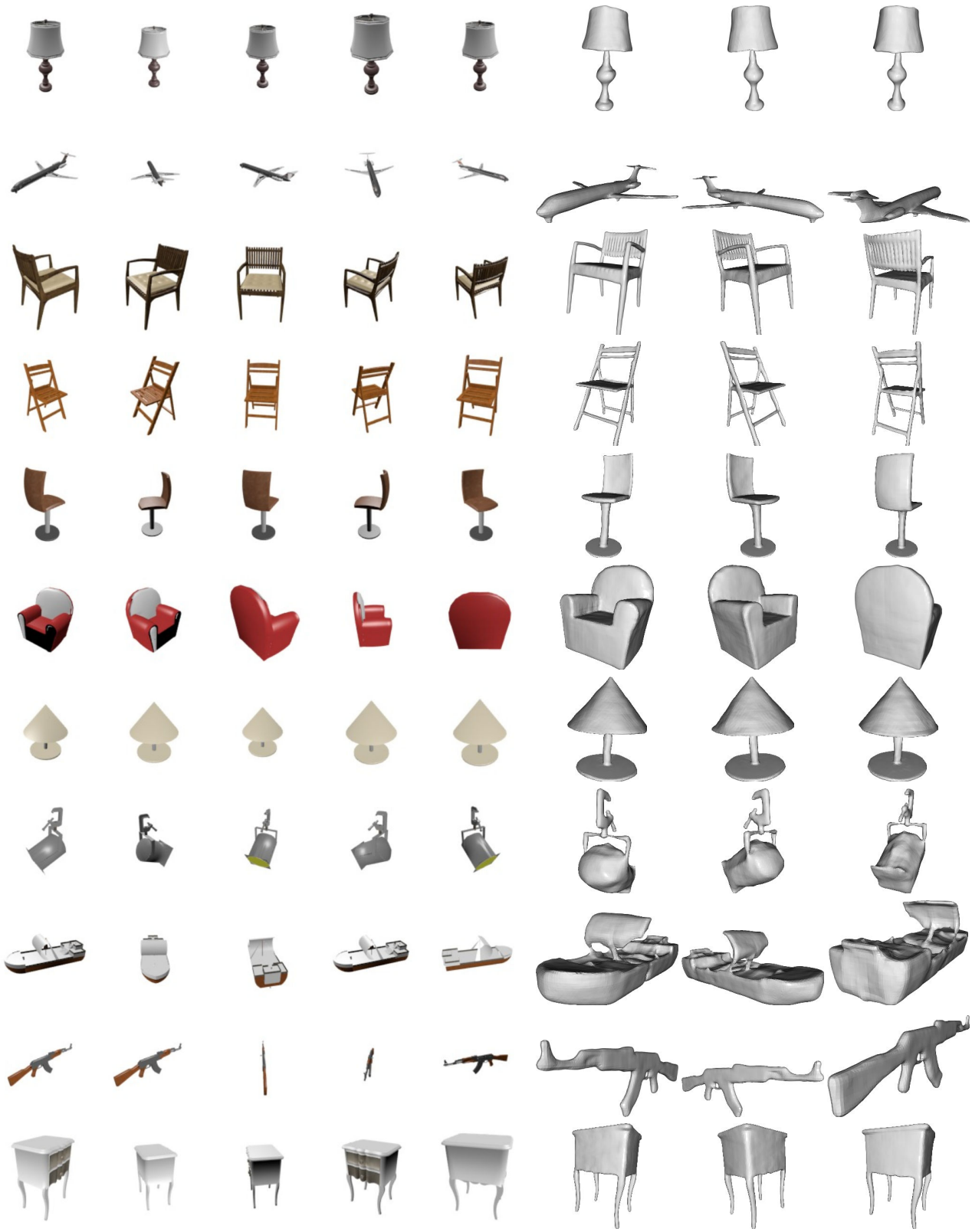

Figure 7. We show qualitative results on more categories of FvOR. On the left we show the 5 view inputs, on the right we show 3 perspective of final reconstructed mesh. FvOR can produce fine-grained reconstruction across all categories.