A. Supplementary material – Overview

The supplementary material is organized as follows. In Sec. A.1 we provide the implementation details for our approach. In Sec. A.2 we give the details of the evaluation metrics used in the main paper and here. In Sec. A.3 we provide the per class results for the Pix3D dataset. In Sec. A.4 we provide an ablation of the ratio between real data and sythetic datasets. In Sec. A.5 we show detailed results on benefits of multiple refiner iterations. Sec. A.6 provides additional quantitative evaluation of the performance of our model. Finally, in Sec. A.7 we provide more qualitative results on Pix3D, Stanford Cars, and CompCars datasets.

A.1. Implementation details and training data

We base our implementation on the render and compare approach of CosyPose [22] for 6D object pose estimation. We recall the main implementation details, explain the main differences with [22], and give the details of our training data.

Network architecture. The architecture of the network F (equation (1) in the main paper) relies on a ResNet-50 [16] backbone, followed by average pooling and a linear layer for predicting the update $\Delta\theta$. The first input block in the backbone is inflated from 3 to 6 channels, to allow for the input of the merged RGB input image and the RGB rendered view. The (cropped) input image and rendering are resized to the input resolution: 640×640 for Pix3D dataset and 300×200 for StanfordCars and CompCars datasets.

Initialization. In all experiments, we set the initial focal length $f^0 = 600$ pixels, which we found experimentally to be a good initial value for all datasets. This focal length could also be initializatied using an EXIF file, or using a coarse estimate directly predicted by a different method. The initialization of the 6D object pose T^0 follows [22] but relies on the initial focal length f^0 instead of using the ground truth focal length for computing an approximation of the object 3D translation. The initial depth of the object is set to $z = 1$ m, and the $x - y$ components of the 3D translation are derived analytically by computing the 3D position of the object center that reprojects to the center of the 2D detection, assuming the camera projection model defined by f^0. The initial object rotation is set to the identity: $R^0 = I_3$.

Coarse estimate and refinement. We follow CosyPose [22] and use two separate networks for coarse initialization and iterative refinement. The coarse network corrects the largest errors (between the observed state and the fixed initialization θ^0) during the first iteration $k = 1$. A separate refinement network iteratively refines the estimates by correcting smaller errors. The refinement network runs for multiple iterations, we run K iterations of the refinement network at test time in our experiments, with $K = 15$ on Pix3D and $K = 55$ on the Stanford cars/CompCars datasets in our experiments.

Training input error distribution. We use the same network architecture defined above for the coarse and refinement networks, but both are trained with different error distributions to simulate what each network is going to see at test time. During training, the initialization of the coarse network is the same as the one used at test time and described in the previous paragraph. Simulating the error distribution of the refinement network is more complicated as its input is not fixed and depends on the coarse estimate. To simulate the errors in focal length which the refinement network will see, we sample the focal length f^k from a gaussian distribution centered on the ground truth f^{gt}, with variance $0.15 f^{gt}$. The error of the input pose given to the refiner is sampled from a Gaussian with standard deviation of $1cm$ around the $x - y$ components of translation, 5 cm for the depth, and noise is added to the ground truth rotation matrix using three Euler angles sampled from Gaussian distributions with variance of $15°$.

Training data. For training our coarse and refinement networks, we use the same training images. They consist of both real training images (of the Pix3D, CompCars/Stanford Cars datasets) and one million synthetic images that are generated for each dataset using the following procedure. For each image, we sample a random object instance, sample its rotation uniformly in the quaternion space, and sample its 3D position within a box of 15 cm size. We add random textures to the object and to the background. The camera-to-object distance is sampled within the interval (0.8, 3.0) meters for the Stanford/Comp cars datasets, and (0.8, 2.4) meters for Pix3D. The focal length is sampled within (200, 1000) pixels, which covers the range of focal lengths from all datasets. While sampling each minibatch during training, one of the real images is sampled with probability 0.5% while the synthetic images are sampled with probability 99.5% and account for most images in each minibatch. Following [22], we also use data augmentation to increase the number of training images. Data augmentation includes adding blur, contrast, brightness, color, and sharpness image filters to the image, and replacing the background with an image from the Pascal VOC dataset with probability 0.3.

Training procedure. The coarse and refinement networks are initialized using classification network pretrained on ImageNet, and are trained using the same procedure as in [22]. Training is performed on 40 NVIDIA A100 GPUs using a global batch size of 1280. The average training time for one coarse/refiner model is around 5 hours. Each network is trained for 10M iterations using the Adam optimizer [21] with a learning rate of 3×10^{-4}. We use a linear warmup of the learning rate during the first 700K iterations and decrease it to 3×10^{-5} after 7M iterations. During infer-

ence, the network can process 32 640×640 pixel resolution images in approximately 10 seconds. This time includes coarse estimation and 15 refiner iterations.

2D detection and instance-recognition. We use Mask R-CNN [15] for predicting a 2D bounding box of the object of interest and identifying the object instance that is rendered during the alignment. The Mask R-CNN is based on a ResNet-50 [16] feature pyramid (FPN) backbone [25]. The network is initialized from a network trained on MS COCO, and the first ten convolutional layers remain fixed during training. This detector is trained using only the data provided by the Pix3D and Stanford/Comp cars datasets.

Cropping strategy. The images from the datasets are center cropped to 640×640px for Pix3D and 300×200px for Stanford cars and CompCars. The input image is padded to conserve the input aspect ratio. The second cropping happens before the input to the network itself. Let us call (x_c, y_c) the 2D coordinates resulting from the projection of the 3D object center by the camera with intrinsic parameter matrix K and $[x_1, y_1, x_2, y_2]$ the coordinates of the bounding box provided by external means (for example, the Mask R-CNN detector), where x_1 is the lower-left coordinate, x_2 is the lower-right coordinate, y_1 is the upper-left coordinate and y_2 is the upper-right coordinate of the provided bounding box. Then we define

$$x_{dist} = \max(|x_1 - x_c|, |x_2 - x_c|), \tag{16}$$

$$y_{dist} = \max(|y_1 - y_c|, |y_2 - y_c|). \tag{17}$$

Then, the cropped image width and height are given by

$$w = \max(x_{dist}, y_{dist}/r) \cdot 2\lambda, \tag{18}$$

$$h = \max(x_{dist}/r, y_{dist}) \cdot 2\lambda, \tag{19}$$

where r is the aspect ratio of the input image and $\lambda = 1.4$ is a parameter controlling the enlargement of the input image to capture the whole object. This value was chosen following [24].

Loss weights. We utilize $\alpha = 10^{-2}$ and $\beta = 1$ as weights for the losses given by equations (8) and (9) in the main paper.

A.2. Evaluation criteria

We now recall the metrics presented in [45], commonly [12, 13, 45] used on these datasets and also used in this work.

Detection metric. We report the detection accuracy $Acc_{D_{0.5}}$ which corresponds to the percentage of images for which the intersection over union between the ground truth and predicted 2D bounding box is larger than 0.5. Note that an incorrect object prediction is not penalized by this metric as our method can predict the focal length and object 6D pose even if the model is only approximate as long as it belongs to the correct category for which the 3D models are approximately aligned, similar to [12, 13, 45].

6D pose metrics. We report the point matching error $e_{R,t}$ that measures the error between the 3D points of the object model transformed with the ground truth and with the estimated 6D pose with respect to the camera:

$$e_{R,t} = \frac{d_{bbox}}{d_{img}} \underset{p \in \mathcal{M}^\star}{\text{avg}} \frac{||(Rp + t) - (\hat{R}p + \hat{t})||_2}{||\hat{t}||_2}, \tag{20}$$

where d_{bbox} is the diagonal of the ground truth 2D bounding box, d_{img} is the diagonal of the image, $\mathcal{M}^\star$ is the 3D model of the ground truth object instance, (R, t) is the predicted 6D pose and $(\hat{R}, \hat{t})$ is the ground truth 6D pose. Note that the point error in 3D (the numerator of (20)) is normalized by the ground truth object-to-camera distance $||\hat{t}||_2$ and multiplied by the relative size of the object in the image $\frac{d_{bbox}}{d_{img}}$ [12].

Following [12], we also use metrics that evaluate separately the quality of the estimated 3D translation and rotation. We use the rotation error computed using the geometric distance between the predicted rotation R and the ground truth rotation $\hat{R}$ $e_R = \frac{||\log(\hat{R}^T R)||_F}{\sqrt{2}}$, and the normalized translation error $e_t = \frac{||t - \hat{t}||_2}{||\hat{t}||_2}$, where t is the predicted translation and $\hat{t}$ is the ground truth translation. For all the errors, we report the median value (denoted as $MedErr_{Rt}$, $MedErr_R$, $MedErr_t$, respectively). Following [12], for the rotation error we also report the percentage of images with $e_R \leq \frac{\pi}{6}$ denoted as $Acc_{R\frac{\pi}{6}}$.

Focal length and reprojection metrics. Following [12], we report the relative focal length error $e_f = \frac{|\hat{f} - f|}{\hat{f}}$ between the estimated focal length f and the ground truth focal length $\hat{f}$. We also report the reprojection error e_P which is similar to the error of 6D pose (eq. (20) but reprojects the 3D points into the image, also taking into account the estimated focal length f:

$$e_P = \underset{p \in \mathcal{M}^\star}{\text{avg}} \frac{||\pi(R, t, f, p) - \pi(\hat{R}, \hat{t}, \hat{f}, p)||_2}{d_{bbox}}, \tag{21}$$

where p are the 3D points of the object model $\mathcal{M}^\star$ of the ground truth object instance, $\pi(K(f), R, t, p)$ is the reprojection of a 3D point p using the estimated parameters, and $\pi(K(\hat{f}), \hat{R}, \hat{t}, p)$ is the reprojection of the same 3D point p using ground truth parameters, and d_{bbox} is the diagonal of the ground truth 2D bounding box. We report the median value of the reprojection error $MedErr_P$ and the percentage of images where the reprojection error is below 0.1 of the image, $Acc_{P_{0.1}}$

A.3. Per class results on the Pix3D dataset

In Tab. 4 we show the performance of our FocalPose approach on individual Pix3D classes. For **bed**, **chair** and **sofa** our algorithm clearly outperforms the prior methods on the five out of eight reported metrics. In particular, we see a clear improvement in the estimated focal length and 3D translation, which validates the contribution of our work. For tables, our approach improves only two out of the eight metrics. We believe this could be attributed to the fact that tables are often symmetric, which makes the 6D object pose estimation approach hard and often ambiguous, as discussed in the main paper. Object symmetries are one of the main failure models of our approach. The overall difficulty of the **table** class is clearly visible from the significantly worse results for all the tested methods on this class.

A.4. Training data ablation

Manually annotating real in-the-wild images [44, 45] with the focal length and 6D pose is difficult because it requires significant effort and the ambiguities can be hard to resolve. This setting results in relatively few available training images. Moreover, the annotations are often of poor quality as has been also discussed in Sec. 4.2 and illustrated in Fig. 4 (row 8) in the main paper. Using synthetic data allows generating many images with accurate annotations. In Tab. 3, we report the results of our coarse model trained with only real data, only synthetic data, or a mix of synthetic and real data in each mini-batch (the fraction of real data in the mixed-data mini-batch is indicated in the table row). Using (exact) synthetic data in addition to a small number of (human-labeled) real images in each mini-batch yields the lowest median error.

A.5. Multiple refiner iterations

Finally, in Figure 7, we show how the model performance evolves with an increasing number of refiner iterations at inference time. Two effects can be observed. First, the translation and focal length errors tend to go down with the number of iterations and they empirically reach a fixed error value. On the other hand, we observe that the rotation errors can increase with the number of iterations, which can be seen for the Pix3D table class. We believe this finding could be attributed to the fact that our refiner model is trained only for one iteration. These results can be potentially improved by increasing the number of refiner iterations during training at the cost of additional compute.

A.6. Detailed results

To show fine-grained information about the errors of our model, we provide a set of histograms and plots that are complementary to the results in Table 2 in the main paper and Table 4 in this supplementary material.

Dataset	$MedErr_R$	$MedErr_t \cdot 10$	$MedErr_f \cdot 10$
Synth only	5.44	2.18	2.04
Synth + Real 0.5%	**2.98**	**1.29**	**1.36**
Synth + Real 5%	3.08	1.33	1.40
Real only	4.13	1.92	1.91

Table 3. **Ablation for combining real and synthetic training data on Pix3D sofa dataset.** Mix of mostly synthetic data with a small number of real images in each mini-batch performs best.

Figure 6. **Inaccuracies in ground truth annotations in the Pix3D dataset.** Example of an alignment with an incorrect 3D model predicted by our approach (right) that results in a lower 3D translation and focal length errors compared to the aligned ground truth 3D model (middle). This is caused by a mismatch between the bed depicted in the input image (with no mattress) and the ground truth 3D model.

In Figure 8 we show the distributions of rotation and reprojection errors for the Pix3D dataset and in Figure 9 for the CompCars and Stanford Cars datasets. For the Pix3D chair and table classes we observe peaks at $\sim$90° intervals, which suggests that many errors in those classes come from symmetrical objects that cause problems for our approach. For the car datasets we observe a large peak at $\sim$ 180°, which also shows that some of the car models are fitted to incorrect orientations due to (almost) symmetrical models.

Figure 10 shows rotation and projection accuracies at different projection and rotation error thresholds. The standard thresholds used in previous work are quite loose and correspond to the right-most endpoints of the reported graphs, *i.e.*, reprojection error of 0.1 (10% of the object bounding box size) and rotation error of 30 degrees. We observe that the accuracy of our approach drops only slightly over a range of tighter thresholds, up to 0.05 relative reprojection error and up to about 15° rotation error. For stricter thresholds (below around 0.05 and 15°), the accuracy of our model starts dropping significantly, which shows that there is still space for improvement in future work.

A.7. Additional qualitative results

In this section, we provide more qualitative results of our approach. Figures 11–18 show additional results for the chair, bed, sofa, and table classes in the Pix3D dataset. Figures 19 and 20 show additional results for the Stanford cars and CompCars datasets, respectively. The qualitative results demonstrate the high accuracy of the alignments obtained by our approach despite variation in focal length, variability of the 3D models that have often very little tex-

Method	Dataset	Class	Detection	Rotation		Translation	Pose	Focal	Projection	
			$Acc_{D_{0.5}}$	$MedErr_R \cdot 1$	$Acc_{R\frac{\pi}{6}}$	$MedErr_t \cdot 10^1$	$MedErr_{R,t} \cdot 10^1$	$MedErr_f \cdot 10^1$	$MedErr_P \cdot 10^2$	$Acc_{P_{0.1}}$
[45]	Pix3D	bed	98.4%	5.82	95.3%	1.95	1.56	2.22	6.05	74.9%
[12] LF	Pix3D	bed	99.0%	5.13	96.3%	1.41	1.04	1.43	3.52	90.6%
[12] BB	Pix3D	bed	**99.5%**	5.40	**97.9%**	1.66	1.17	1.59	3.55	**93.2%**
Ours	Pix3D	bed	98.4%	**3.16**	91.6%	**1.28**	**0.93**	**1.28**	**1.91**	88.9%
[45]	Pix3D	chair	94.9%	7.52	88.0%	2.69	1.58	1.98	6.04	75.3%
[12]-LF	Pix3D	chair	95.2%	7.52	88.8%	1.92	1.21	1.62	3.41	88.2%
[12]-BB	Pix3D	chair	**97.3%**	6.95	**91.0%**	1.68	1.08	1.58	3.24	**90.9%**
Ours	Pix3D	chair	91.8%	**3.56**	85.4%	**1.49**	**0.94**	**1.36**	**1.73**	79.3%
[45]	Pix3D	sofa	96.5%	4.73	94.8%	2.28	1.62	2.42	4.33	82.2%
[12] LF	Pix3D	sofa	96.5%	4.49	95.0%	1.92	1.33	1.79	2.56	93.7%
[12] BB	Pix3D	sofa	**98.3%**	4.40	97.0%	1.63	1.16	1.73	2.13	**95.6%**
Ours	Pix3D	sofa	96.9%	**2.98**	97.6%	**1.29**	**0.83**	**1.36**	**1.52**	93.9%
[45]	Pix3D	table	94.0%	10.94	72.9%	3.16	2.28	3.03	8.90	53.6%
[12] LF	Pix3D	table	94.0%	10.53	73.5%	2.16	**1.62**	**2.05**	5.92	69.5%
[12] BB	Pix3D	table	**95.7%**	10.80	**77.2%**	2.81	1.78	2.10	**5.74**	**72.4%**
Ours	Pix3D	table	94.9%	**9.98**	61.8%	**1.90**	1.68	2.13	6.72	54.7%

Table 4. **Comparison with the state of the art for 6D pose and focal length prediction** on the Pix3D dataset split by class. **Bold** denotes the best result among directly comparable methods. See section A.3 in this supplementary for a more detailed analysis of the results.

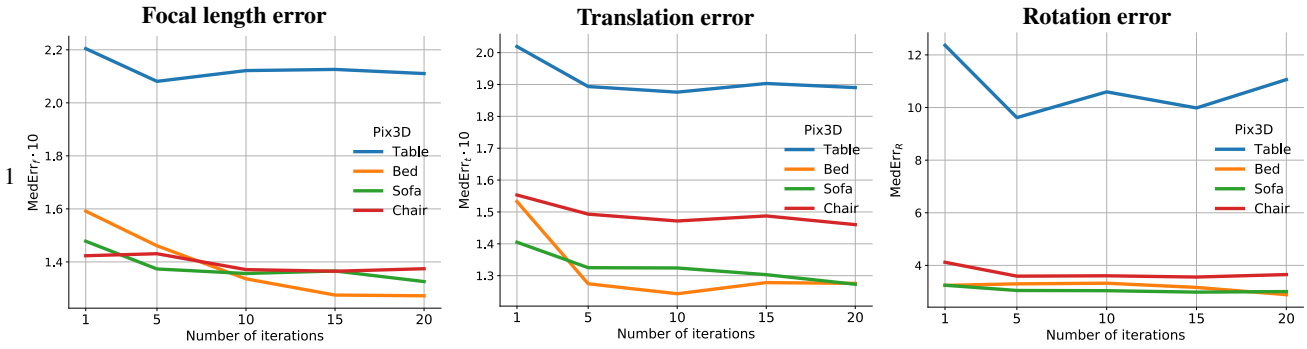

Figure 7. Evolution of errors with an increasing number of refiner iterations at inference time for different object classes on the Pix3D dataset.

ture, occlusions, and cluttered backgrounds. Finally, Figure 21 shows additional examples of failure modes on the Pix3D dataset.

For Pix3D, we provide good results for the chair class in Fig. 11 and Fig. 12, for the bed class in Fig. 13 and Fig. 14, for the sofa class in Fig. 15 and Fig. 16 and for the table class in Fig. 17 and Fig. 18. We also provide qualitative results for Stanford cars in Fig. 19 and for CompCars in Fig. 20. Please notice the quality of alignment that our approach can achieve. We provide the failure cases for the Pix3D dataset in Fig. 21.

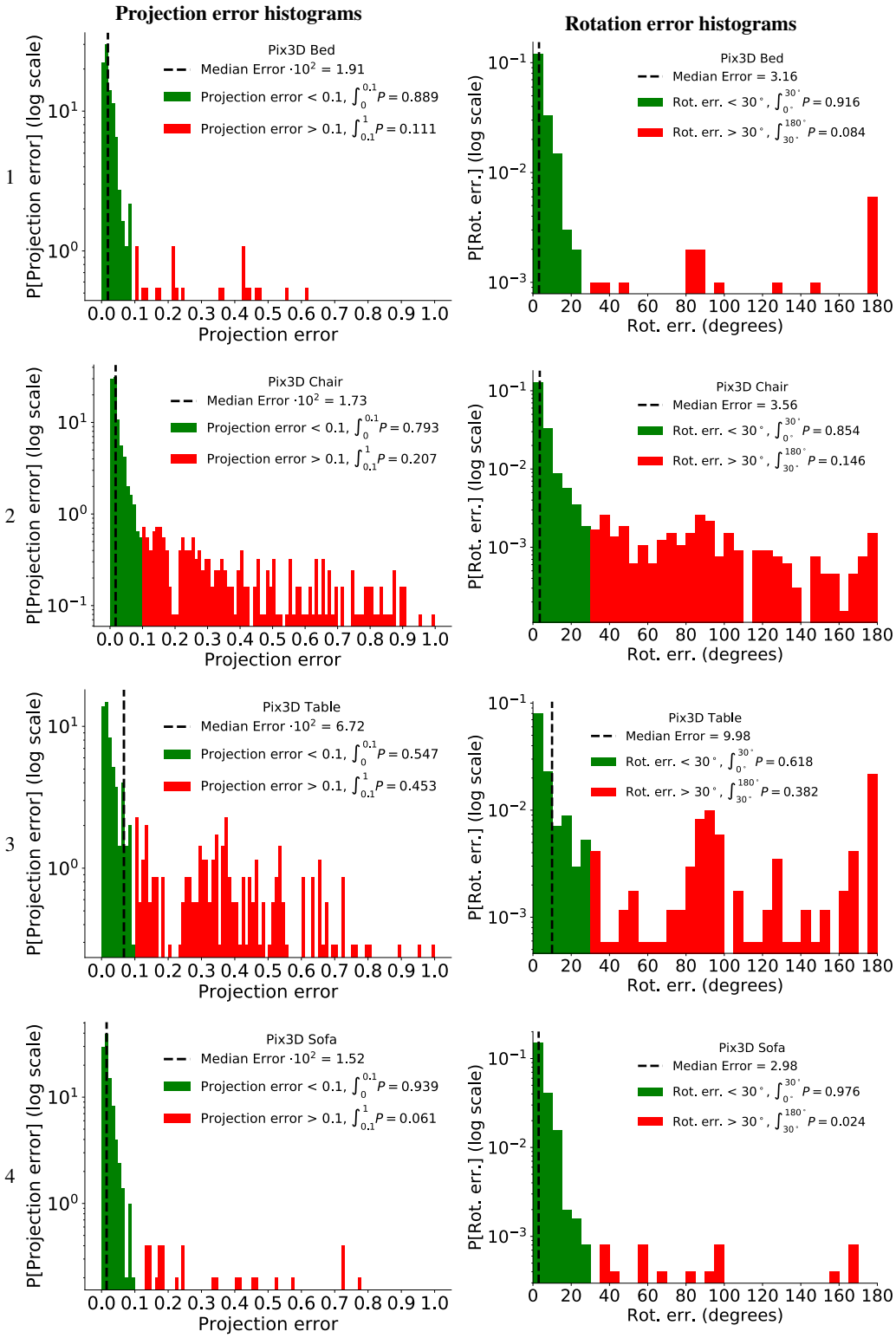

Figure 8. **Projection error histograms (left) and rotation error histograms (right) for the Pix3D object classes.** Please note the logarithmic scale of the y-axis.

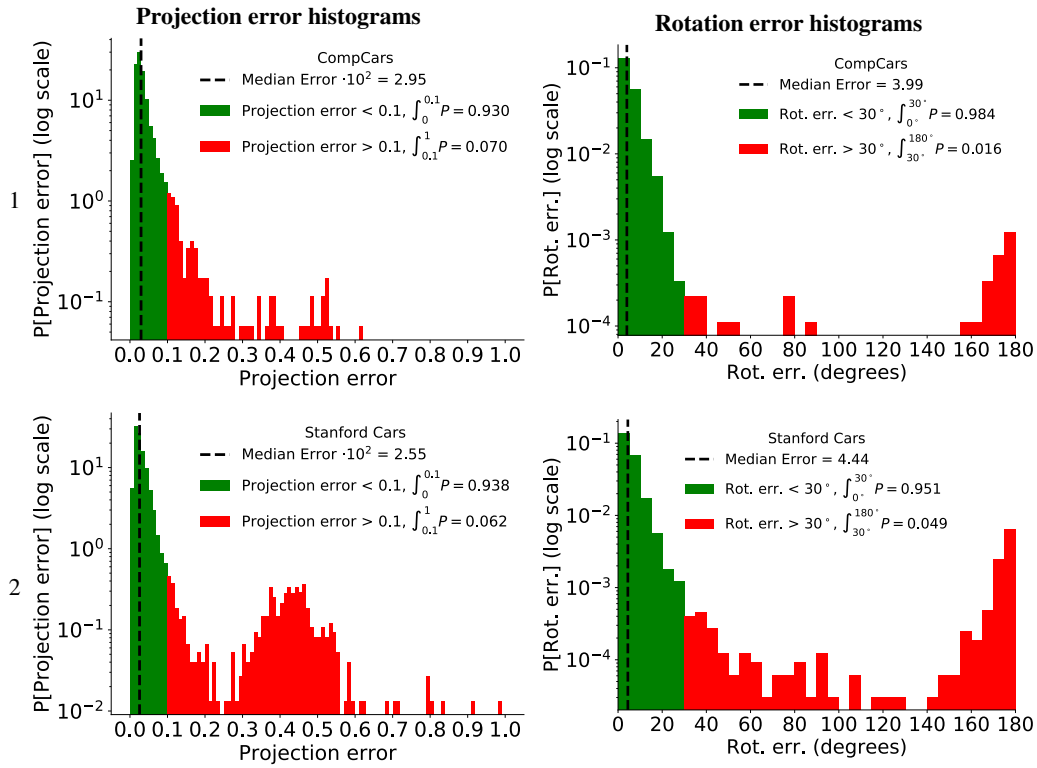

Figure 9. **Projection error histograms (left) and rotation error histograms (right) for the CompCars (first row) and Stanford Cars (second row) datasets.** Please note the logarithmic scale of the y-axis.

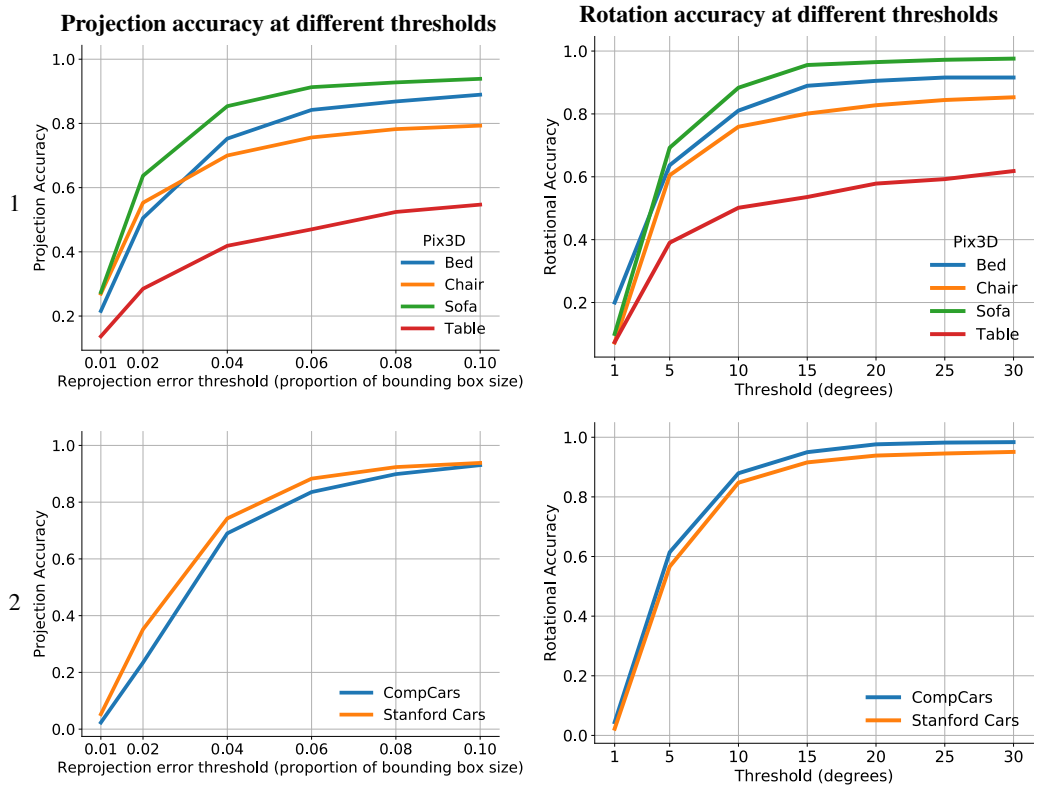

Figure 10. Projection and rotation accuracies at different error thresholds.

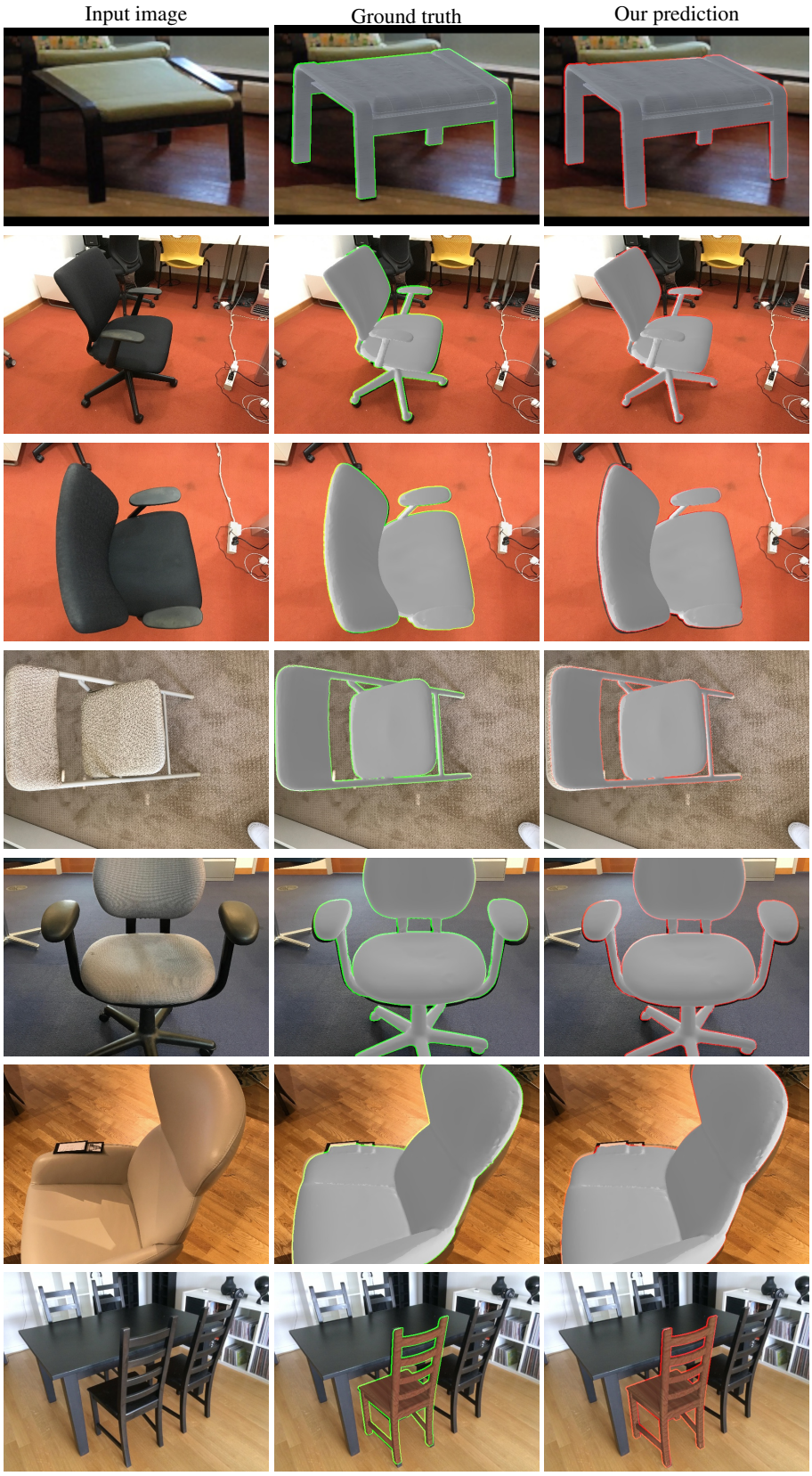

Figure 11. Qualitative results for Pix3D chairs - part 1.

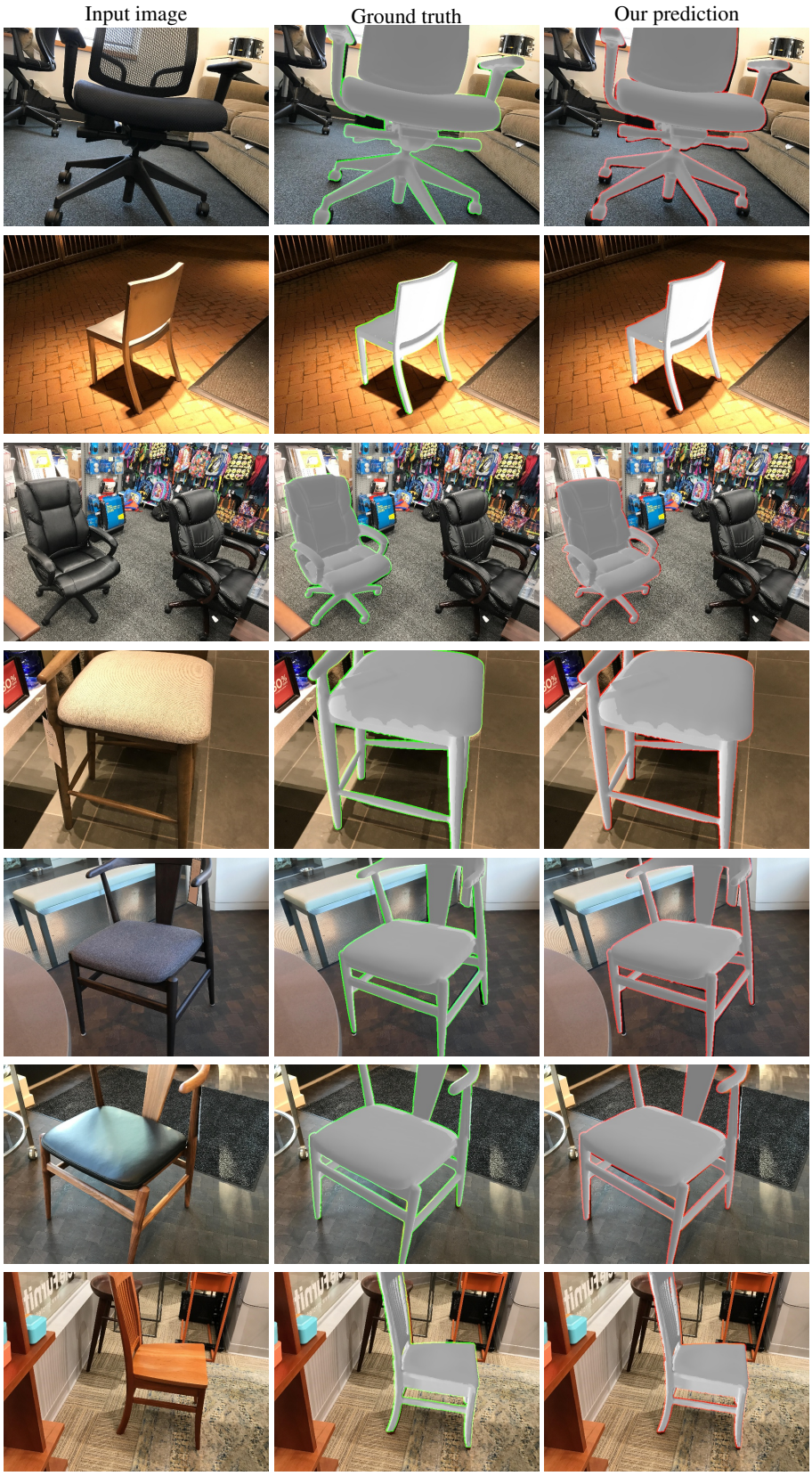

Figure 12. Qualitative results for Pix3D chairs - part 2.

Figure 13. Qualitative results for Pix3D beds - part 1.

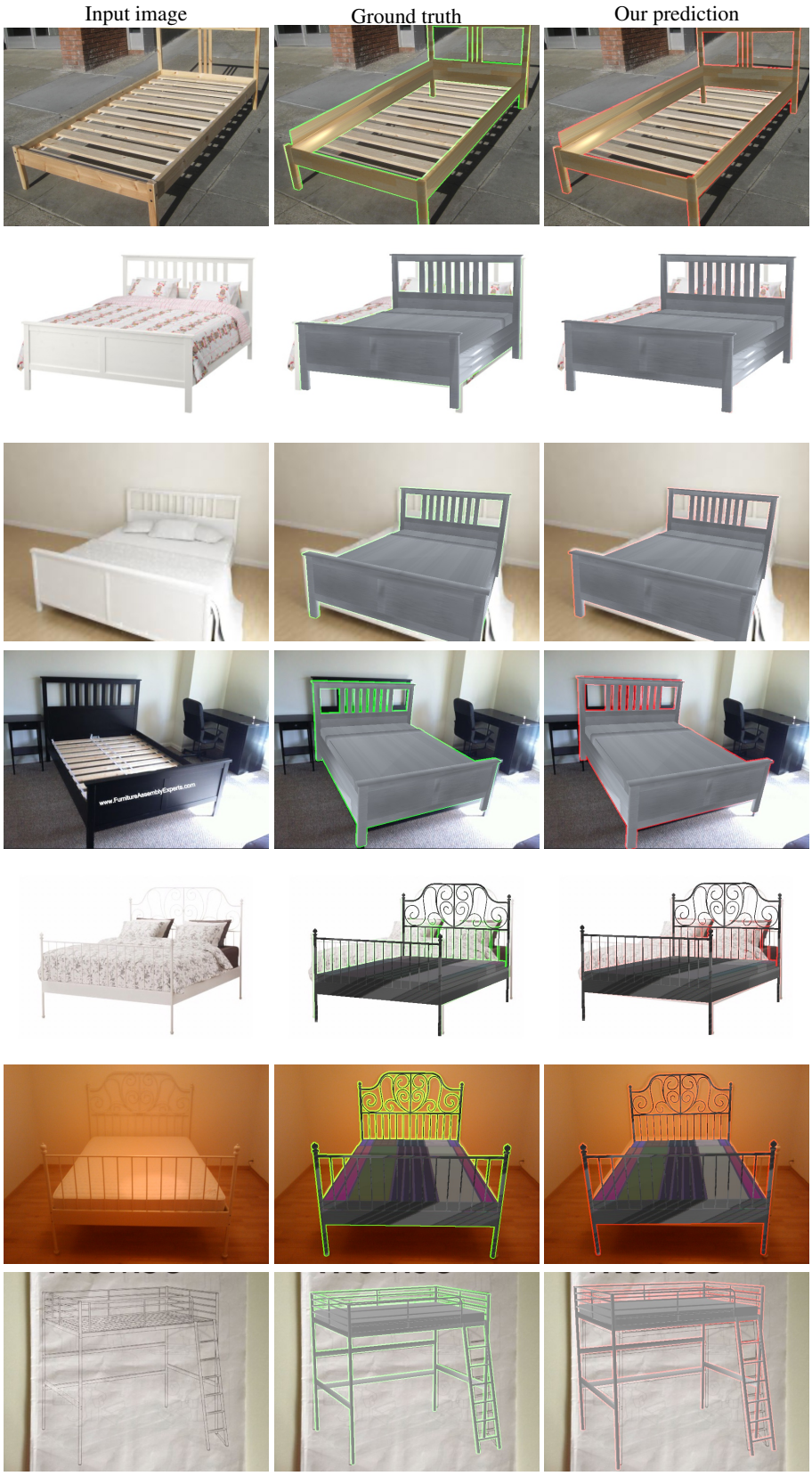

Figure 14. Qualitative results for Pix3D beds - part 2.

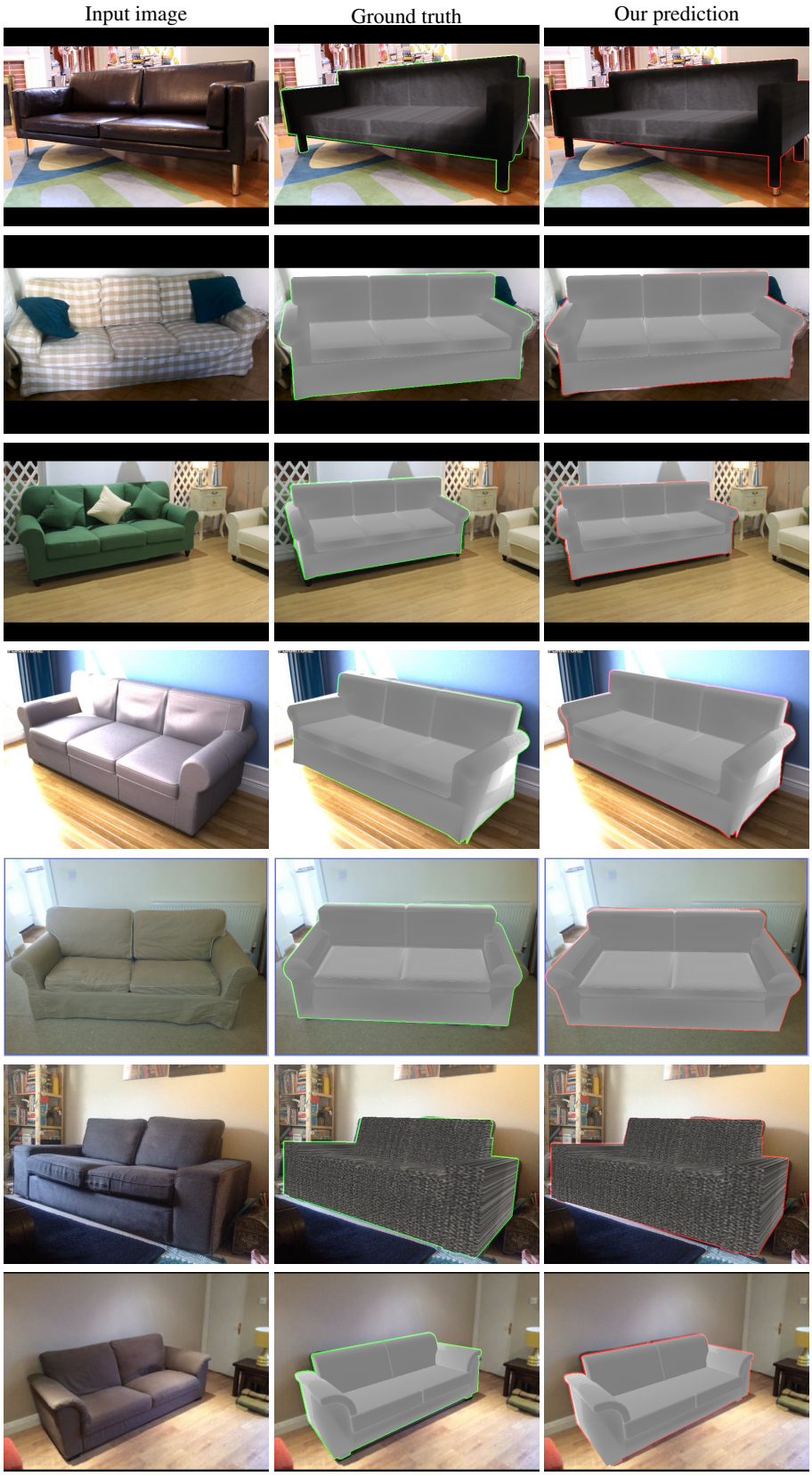

Figure 15. Qualitative results for Pix3D sofas - part 1.

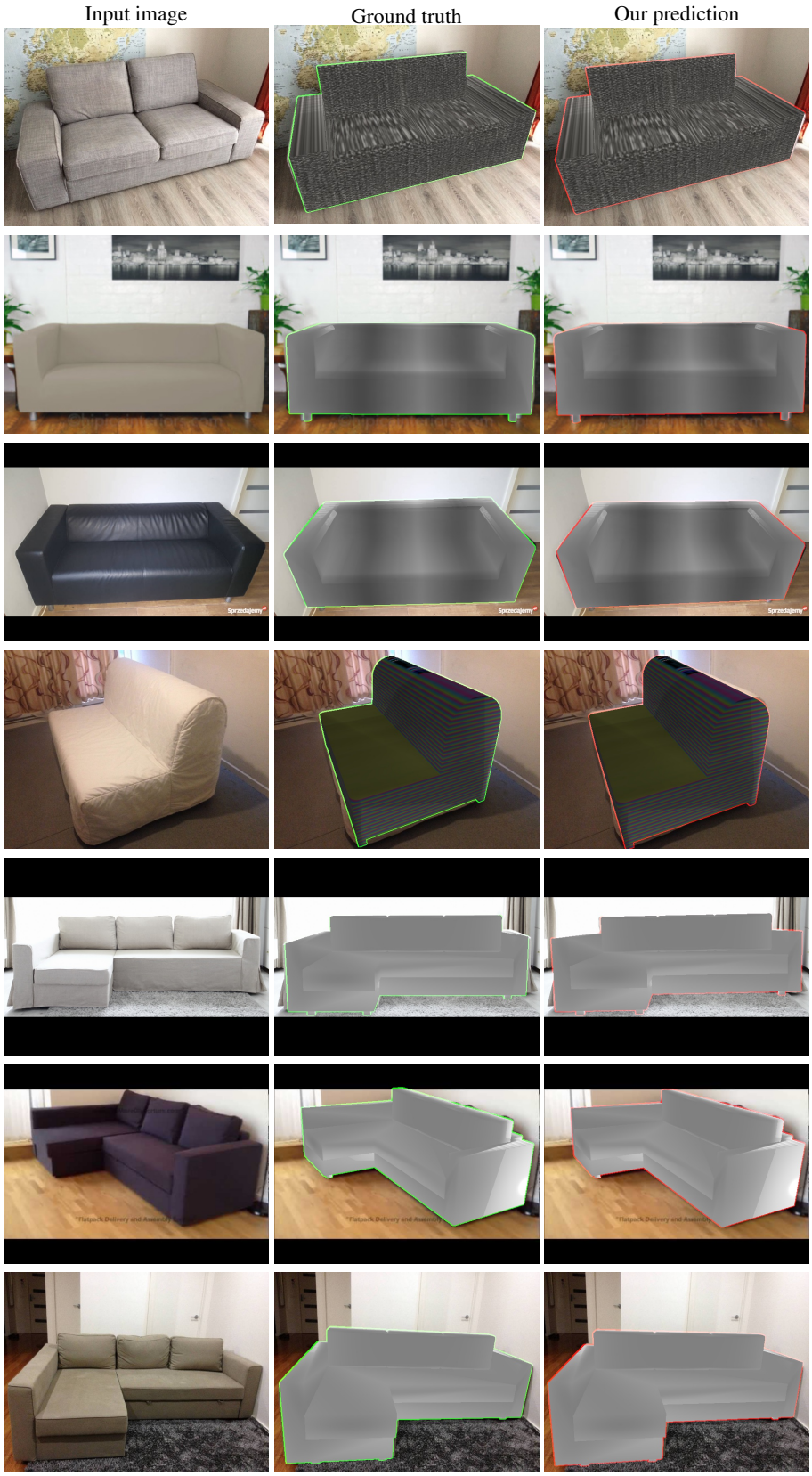

Figure 16. Qualitative results for Pix3D sofas - part 2.

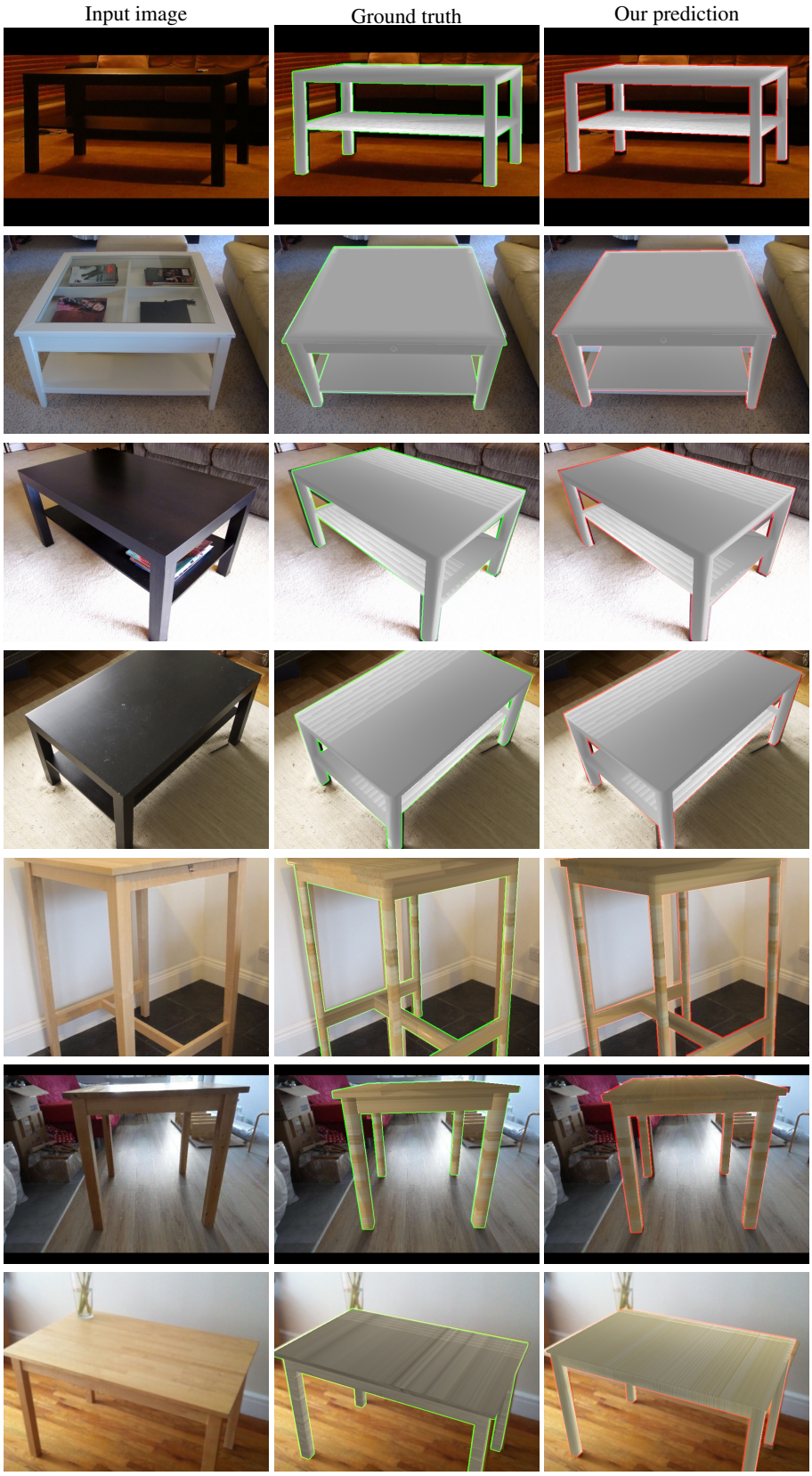

Figure 17. Qualitative results for Pix3D tables - part 1.

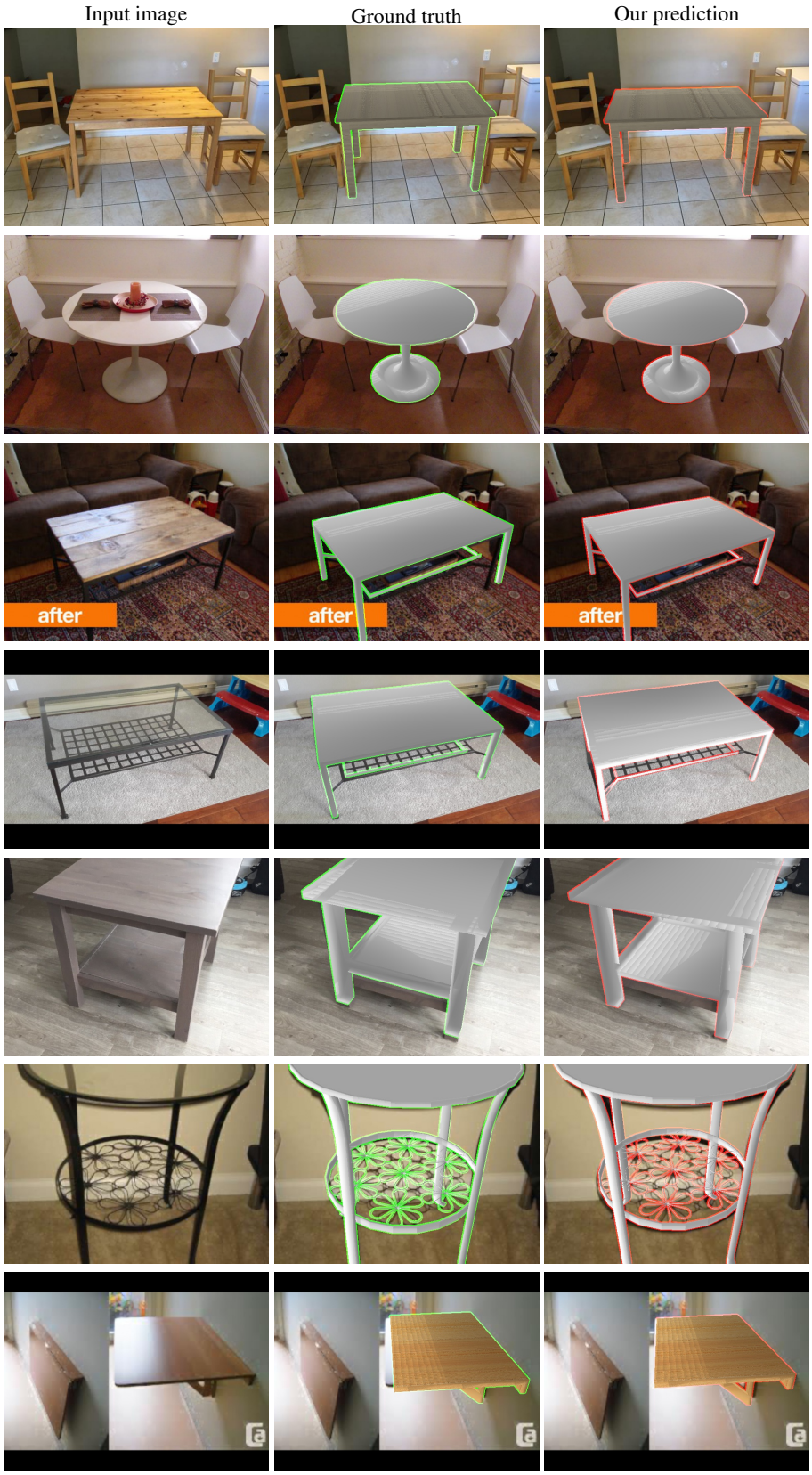

Figure 18. Qualitative results for Pix3D tables - part 2.

Figure 19. Qualitative results for the CompCars dataset.

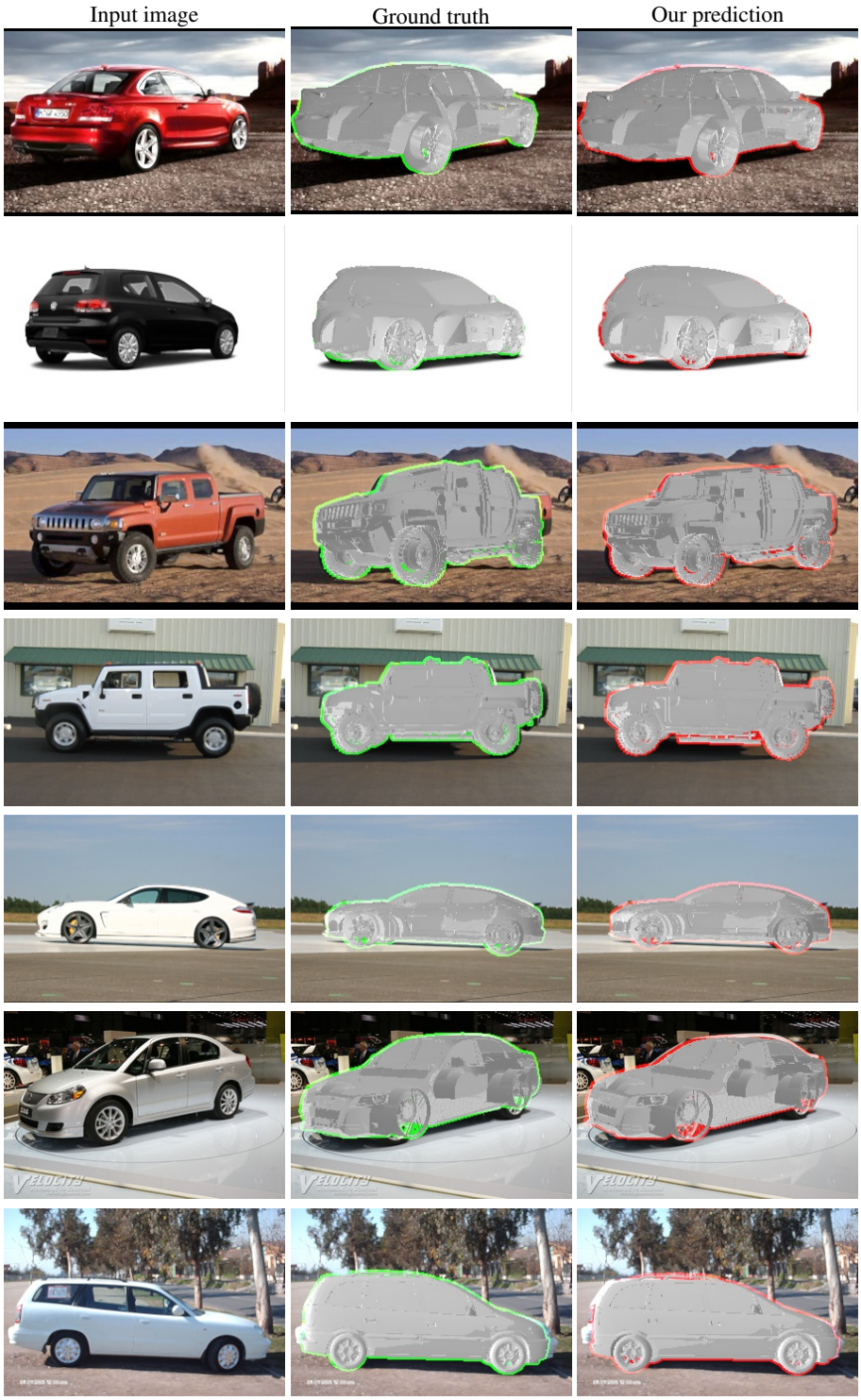

Figure 20. Qualitative results for the Stanford car dataset.

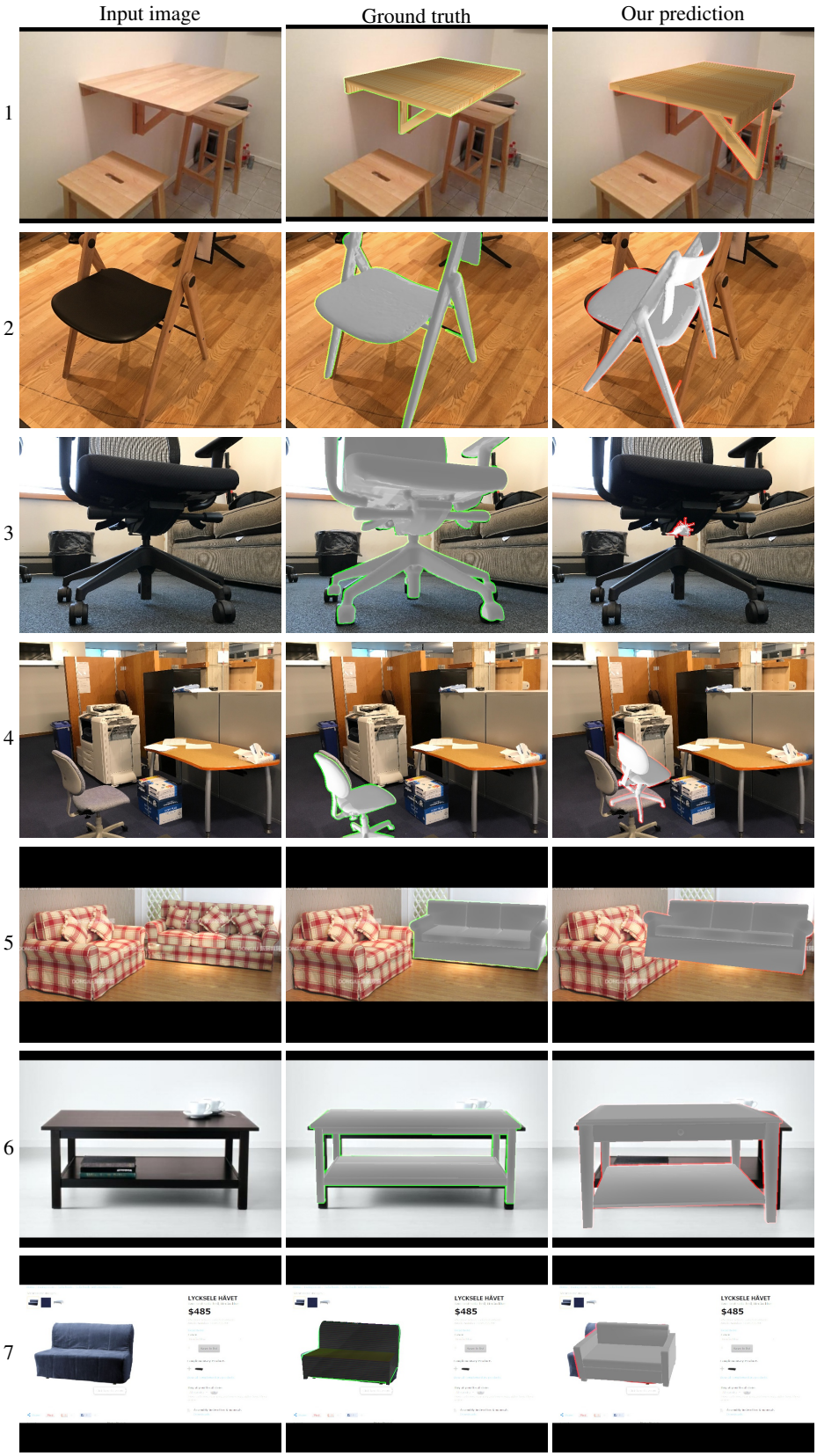

Figure 21. Examples of failures in the Pix3D dataset. Typical failures include symmetric objects (rows 1-2), local minima (rows 3-5) and misalignment due to the incorrect model (row 6-7). For more details please see Sec. 5 and Fig. 4 in the main paper.