Supplementary Materials for "Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation"

Chao Wen[1*] Yinda Zhang[2*] Zhuwen Li[3*] Yanwei Fu[1†]
[1]Fudan University [2]Google LLC [3]Nuro, Inc.

This supplementary material includes the implementation details, baseline algorithm designs, and more experiment results.

1. Network Architecture

1.1. Deformation Sampling

In particular, a level-K icosahedron is obtained by sampling the middle points of all the edges from a level-(K-1) icosahedron, and a level-0 icosahedron vertex coordinates can be calculated as

$$C_0 = \begin{vmatrix} \phi & 1 & 0 \\ -\phi & 1 & 0 \\ \phi & -1 & 0 \\ -\phi & -1 & 0 \\ 1 & 0 & \phi \\ 1 & 0 & -\phi \\ -1 & 0 & \phi \\ -1 & 0 & -\phi \\ 0 & \phi & 1 \\ 0 & -\phi & 1 \\ 0 & \phi & -1 \\ 0 & -\phi & -1 \end{vmatrix} / \sqrt{1+\phi^2}, \tag{1}$$

where

$$\phi = \frac{1+\sqrt{5}}{2}. \tag{2}$$

Here, each row represents a vertex coordinate on the level-0 icosahedron.

In our experiment, we use Meshlab [3] to generate level-1 icosahedron [10] vertices coordinates and corresponding edges for deformation hypotheses and local GCN graph topology. The vertex coordinates are scaled along the radius to a pre-defined value. More details about connection and implementation can be found in [1, 11].

1.2. Deformation Reasoning

The network architecture for deformation reasoning is shown in Tab. 1. The Deformation Reasoning component takes current vertices coordinates, hypothesis feature and hypothesis coordinates as input. It consists of 6 graph convolution layers with residual connections. The last graph convolution layer is followed by a softmax layer and the final output is normalized to weights, which are used to obtain new coordinates for the vertices through weighted sums.

1.3. Perceptual Feature Pooling

The perceptual feature pooling layer projects all 3D vertices onto feature maps and obtain the vertices features from the corresponding 2D coordinates. Suppose a 3D vertex with coordinate (X, Y, Z) in the camera view; its 2D projection in image is:

$$\begin{aligned} x &= \frac{X}{Z} * f_x + c_x, \\ y &= \frac{Y}{Z} * f_y + c_y, \end{aligned} \tag{3}$$

where f_x and f_y denote the focal lengths along horizontal and vertical image axis, and (c_x, c_y) is the projection of the camera center.

To pool feature from multiple images for each vertex, we transform the vertex into the camera coordinate of each input views using the camera extrinsic matrix. Suppose the $\{R, T\}$ is the transformation from the world coordinate to a camera coordinate, and V is the coordinate of a vertex in the world coordinate, its location in the camera coordinate can be obtained by $V_c = R \cdot V + T$.

2. Baselines Methods

In Sec. 4.2 of the main submission, we propose two baselines extending the Pixel2Mesh architecture for multi-view shape generation. Here we show more details about them.

2.1. P2M-M

For the P2M-M baseline, we first run the single-view Pixel2Mesh on each of the input views to generate a shape respectively. These shapes are then transformed into the

[*]indicates equal contributions.
[†]indicates corresponding author.

Tensor Name	Layer	Parameters	Input Tenstor	Activation
Vertices Coordinate	Input	-	-	-
Hypothesis Coordinate	Input	-	-	-
Hypothesis Feature	Input	-	-	-
GraphConv1	GraphConv	339×192	Hypothesis Feature	ReLU
GraphConv2	GraphConv	192×192	GraphConv1	ReLU
GraphConv3	GraphConv	192×192	GraphConv2	ReLU
Add1	Add	-	GraphConv2, GraphConv3	-
GraphConv4	GraphConv	192×192	Add1	ReLU
GraphConv5	GraphConv	192×192	GraphConv4	ReLU
Add2	Add	-	GraphConv4, GraphConv5	-
GraphConv6	GraphConv	192×1	Add2	ReLU
Hypothesis Score	Softmax	-	GraphConv6	-
New Vertices Coordinate	Weighted Sum	-	Hypothesis Score, Hypothesis Coordinate, Vertices Coordinate	-

Table 1. **Network Architecture for Deformation Reasoning.**

world coordinate and converted into signed distance function (SDF) [4, 8, 7]. We directly average these SDFs and run Lorensen *et al.* [6] to obtain the triangular meshes.

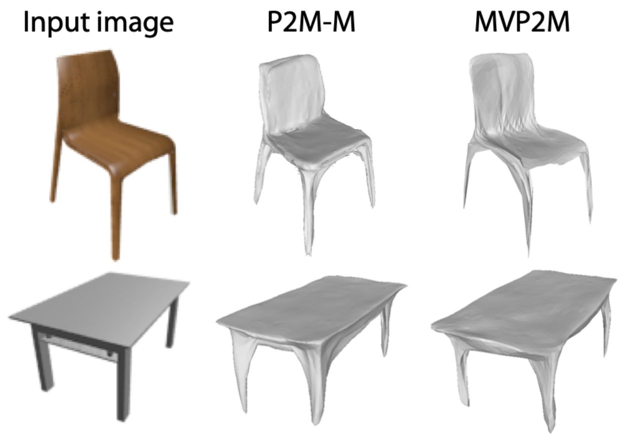

Figure 1. **Results of baselines.**

2.2. MVP2M

For the P2M-M baseline, Pixel2Mesh sees only one image at once. Here, we extend Pixel2Mesh to access multiple images in a single network forward pass by having it pools multi-view features from all the inputs. This can be achieved by replacing the perceptual feature pooling layers with our multi-view version as introduced in Sec. 1.3. In particular, perceptual features are pooled from layer 'conv3_3', 'conv4_3', and 'conv5_3' from the VGG-16 network, and feature statistics (Sec. 3.1.2 in the main submission) are calculated and concatenated, which ends up with a 1280 dimension feature vector. In practice, we also tried to pool geometry related features from early convolution layers (i.e. 'conv1_2', 'conv2_2', and 'conv3_3'), but found it doesn't work as well as the case with semantic feature pooled from later layers.

Fig. 1 shows some examples of results from both baselines.

3. More Experiment Results

In this section, we provide more results for quantitative and qualitative evaluations and ablation study.

3.1. Comparison to State-of-the-art

In the main submission, we compare to the state-of-the-art methods in F-score. Here we show the comparison in Chamfer distance in Tab. 2. Again, we achieve overall the best performance (i.e. the lowest Chamfer distance) comparing to all the previous methods and baselines. We also achieve the best performance for most of the categories, except for very few categories in which geometry and texture are usually too simple to learn cross-view information.

3.2. Ablation Study

3.2.1 Effect of Re-sample Loss

More comparison between the model trained with the traditional and our re-sampled Chamfer loss is shown in Fig. 2. As can be seen in the zoom-in areas, our re-sampled Chamfer loss can effectively penalize large flying triangles caused by a few flying vertices , and thus the results of our full model are free from such artifacts.

3.2.2 Effect of More Iterations

In our main submission, we show the numerical improvements with more iterations. Here we show some qualitative

Category	Chamfer Distance(CD) ↓				
	3DR2N2†	LSM	MVP2M	P2M-M	Ours
Couch	0.806	0.730	0.534	0.496	**0.439**
Cabinet	0.613	0.634	0.488	0.359	**0.337**
Bench	1.362	0.572	0.591	0.594	**0.549**
Chair	1.534	0.495	0.583	0.561	**0.461**
Monitor	1.465	0.592	0.658	0.654	**0.566**
Firearm	0.432	0.385	0.305	0.428	**0.305**
Speaker	1.443	0.767	0.745	0.697	**0.635**
Lamp	6.780	1.768	**0.980**	1.184	1.135
Cellphone	1.161	0.362	0.445	0.360	**0.325**
Plane	0.854	0.496	**0.403**	0.457	0.422
Table	1.243	0.994	0.511	0.441	**0.388**
Car	0.358	0.326	0.321	0.264	**0.249**
Watercraft	0.869	0.509	**0.463**	0.627	0.508
Mean	1.455	0.664	0.541	0.548	**0.486**

Table 2. **Comparison to Multi-view Shape Generation Methods.** We show the Chamfer Distance on each semantic category. Our method achieves the best performance overall. The notation † indicates the methods which does not require camera extrinsics.

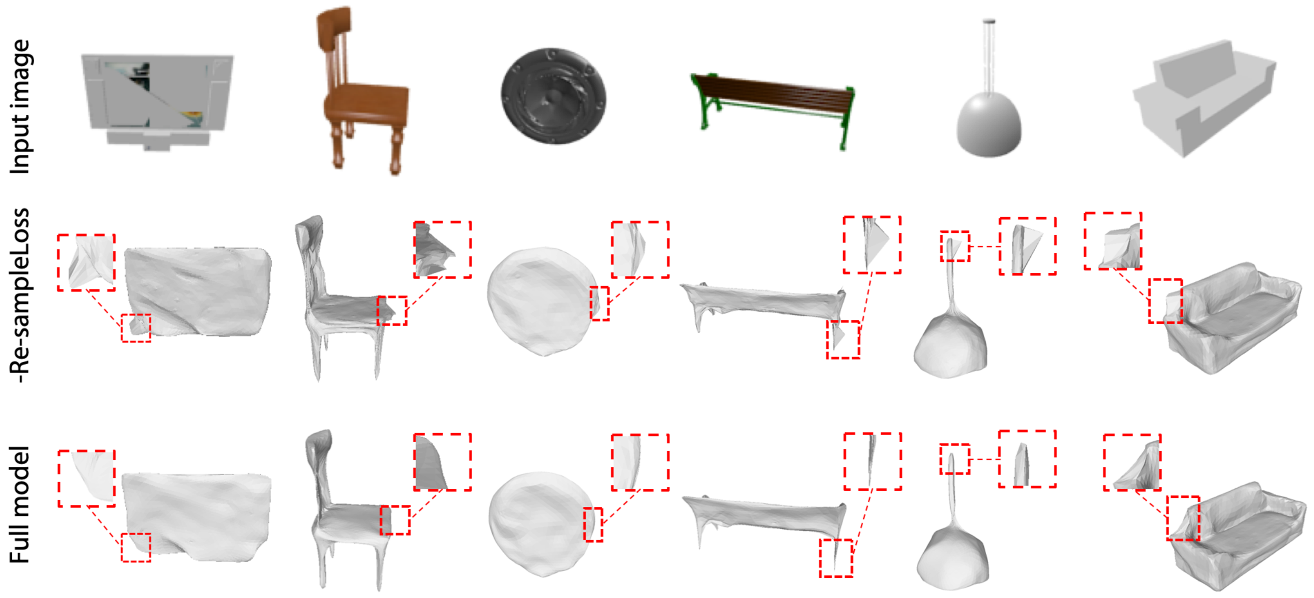

Figure 2. **Effect of Re-sampled Loss.** We show more qualitative comparison between model trained with original Chamfer loss and our re-sampled version. The re-sampled loss (Full model) helps to prevent the flying pixel and spike artifacts.

results in Fig. 3. As can be seen, thin structure and surface details are recovered throughout iterations as reflected in the zoom-in regions.

3.2.3 Effect of Different Coarse Shape Generation

We add another experiment using MVP2M and P2M-M as coarse shape initialization methods respectively. Here we show the comparison results in Tab. 3. MDN consistently improves upon both P2M-M and MVP2M. Ours is also slightly better than P2M-M+MDN as the initialization is better. In order to emphasize the generalization ability of using non-ellipsoid initial, we also add experimental results of training on the chair class using other voxel-based methods e.g. 3DR2N2 as a rough shape initialization methods. The qualitative and quantitative result are shown in Fig. 4. As can be seen, MDN generalizes to meshes obtained from 3DR2N2 directly without finetuning.

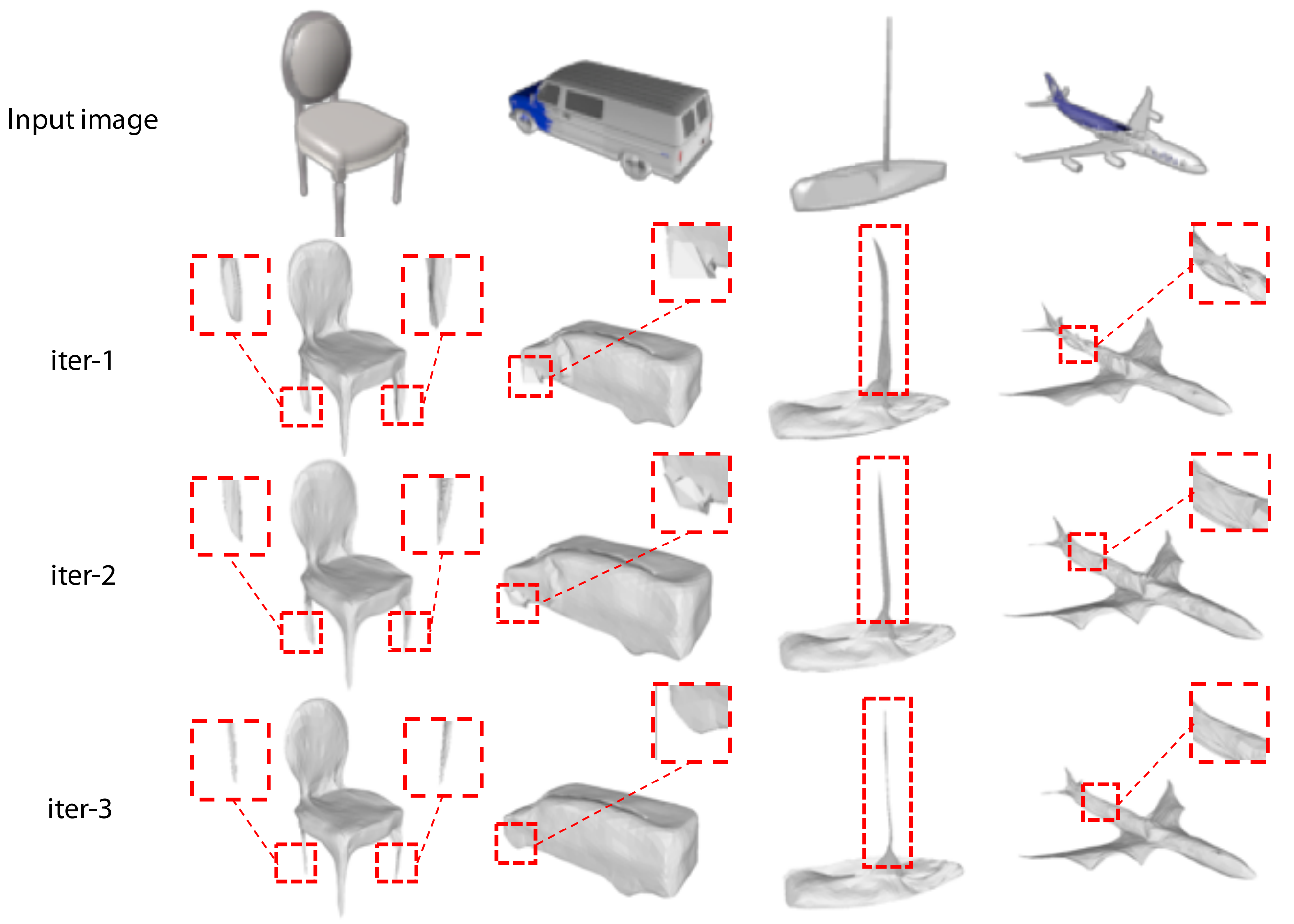

Figure 3. **Effect of Iterations.** We show the output of our system after each iterations. Thin structures and geometry details are recovered in the later iterations.

Methods	CD↓	F-score(τ)↑	F-score(2τ)↑
P2M-M	0.548	61.72	76.45
P2M-M+MDN	0.493	64.47	79.31
MVP2M	0.541	61.05	77.10
Ours	0.486	66.48	80.30

Table 3. **Effect of Different Coarse Shape Generation.**

(a) Example mesh result.

Metrics	w/o MDN	w/ MDN
CD	2.438	**1.418**
F-score(τ)	20.24	**36.81**
F-score(2τ)	31.62	**52.45**

(b) 3DR2N2 scheme with/wo MDN on chair

Figure 4. Experiments results using non-ellipsoid initial.

3.3. More Qualitative Results

In the end, we show more qualitative results in Fig. 5, Fig. 6, and Fig. 7. For each example, we show the input image and the results from 3D-R2N2 [2], LSM [5], P2M [9], our model, and the ground truth. In overall, our model produces accurate shapes that align well with input views and maintain good surface details.

4. Discussion about Self-intersection

Some experiments results indicate Pixel2Mesh suffers from self-intersection since it was not explicitly handled. In contrast, we observed that Pixel2Mesh++ produces results with less intersection even though we did not particularly handle it either. We conjecture that this is because geometric reasoning cross checks information from multi-view, and thus the shape generation is more stable and robust. Using more stable features and larger Laplacian regular terms in training may alleviate this problem as well.

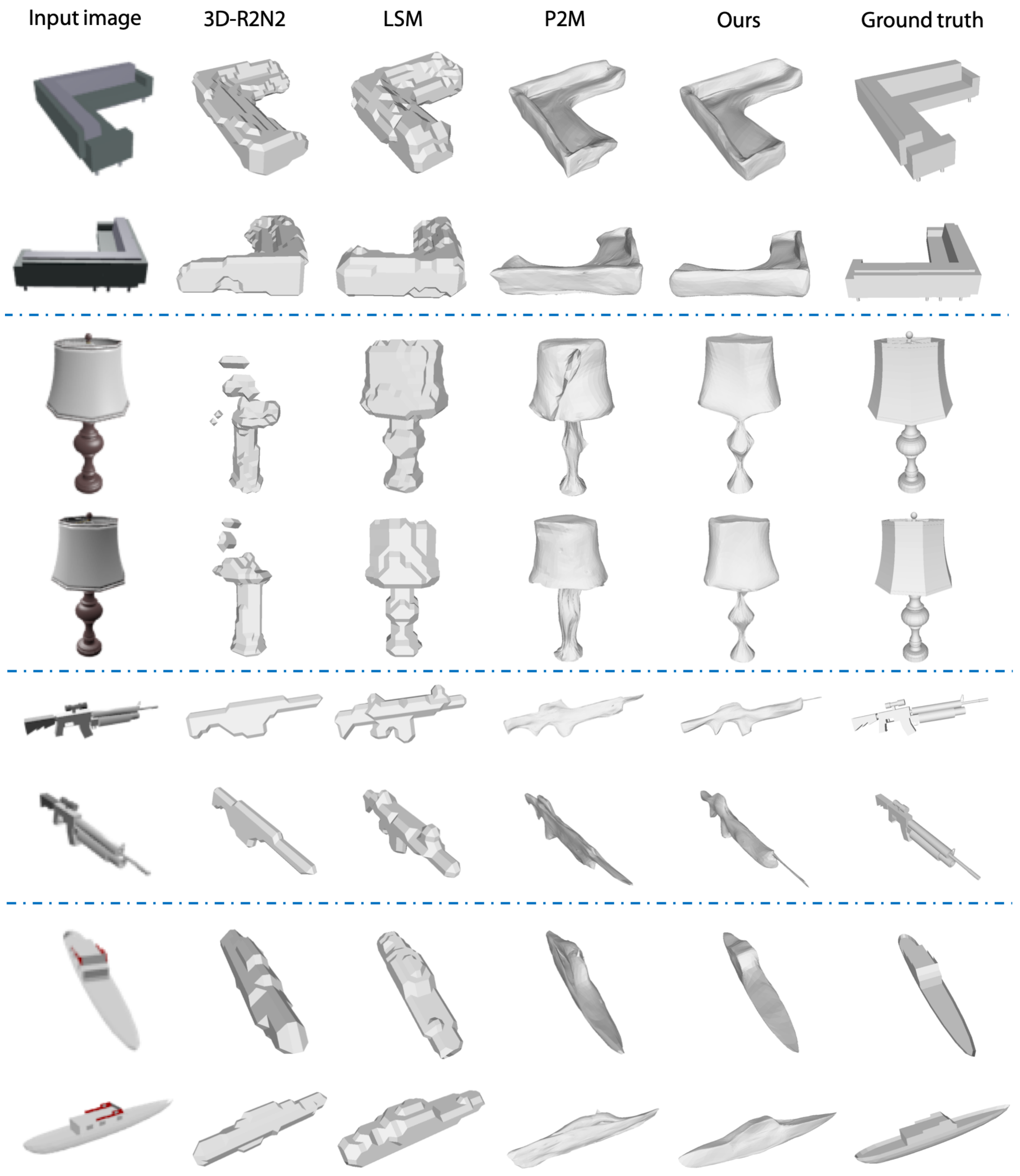

Figure 5. **More Qualitative Results.** From top to bottom, we show for each example: two camera views, results of 3DR2N2, LSM, Pixel2Mesh, ours, and the ground truth.

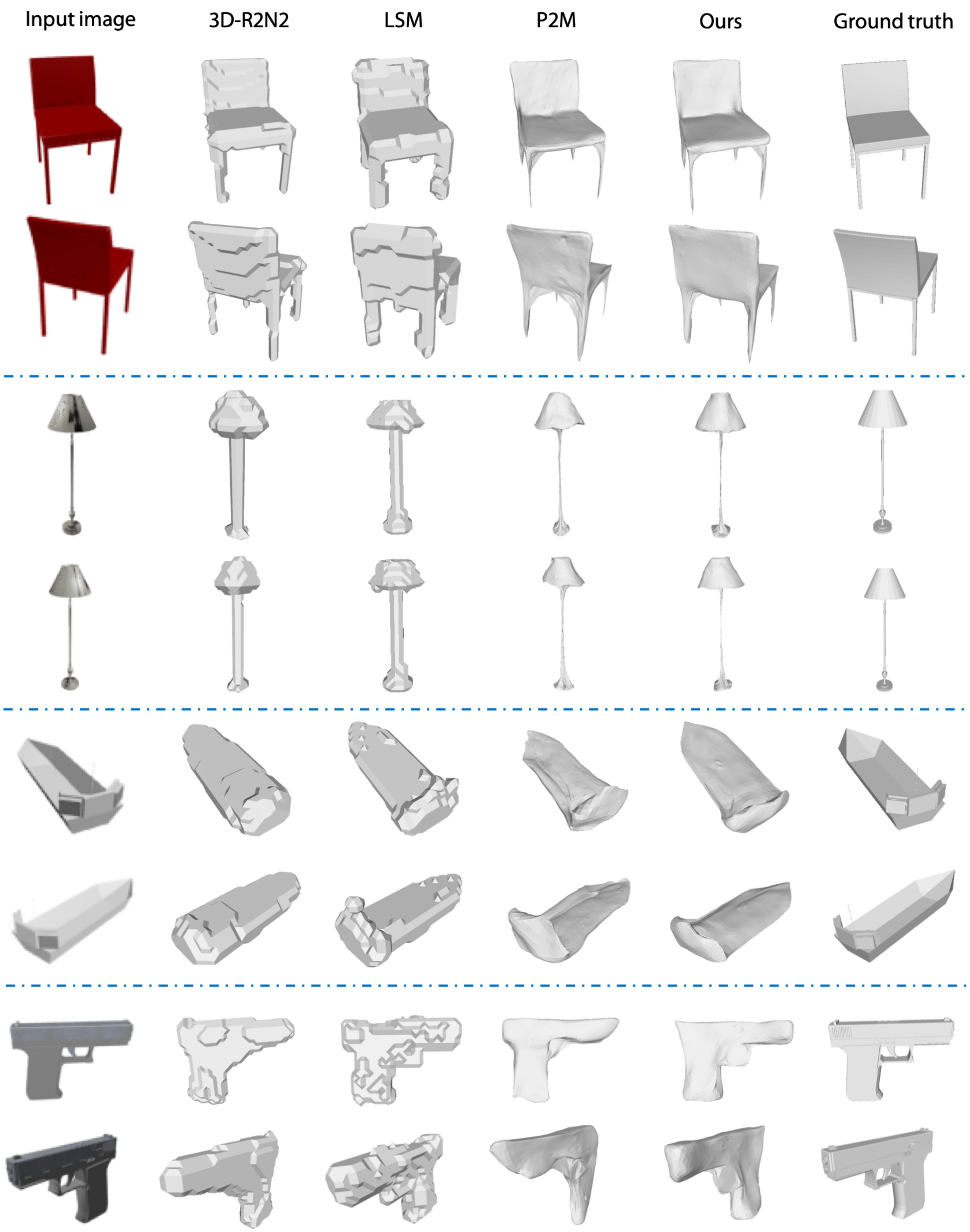

Figure 6. **More Qualitative Results.** From top to bottom, we show for each example: two camera views, results of 3DR2N2, LSM, Pixel2Mesh, ours, and the ground truth.

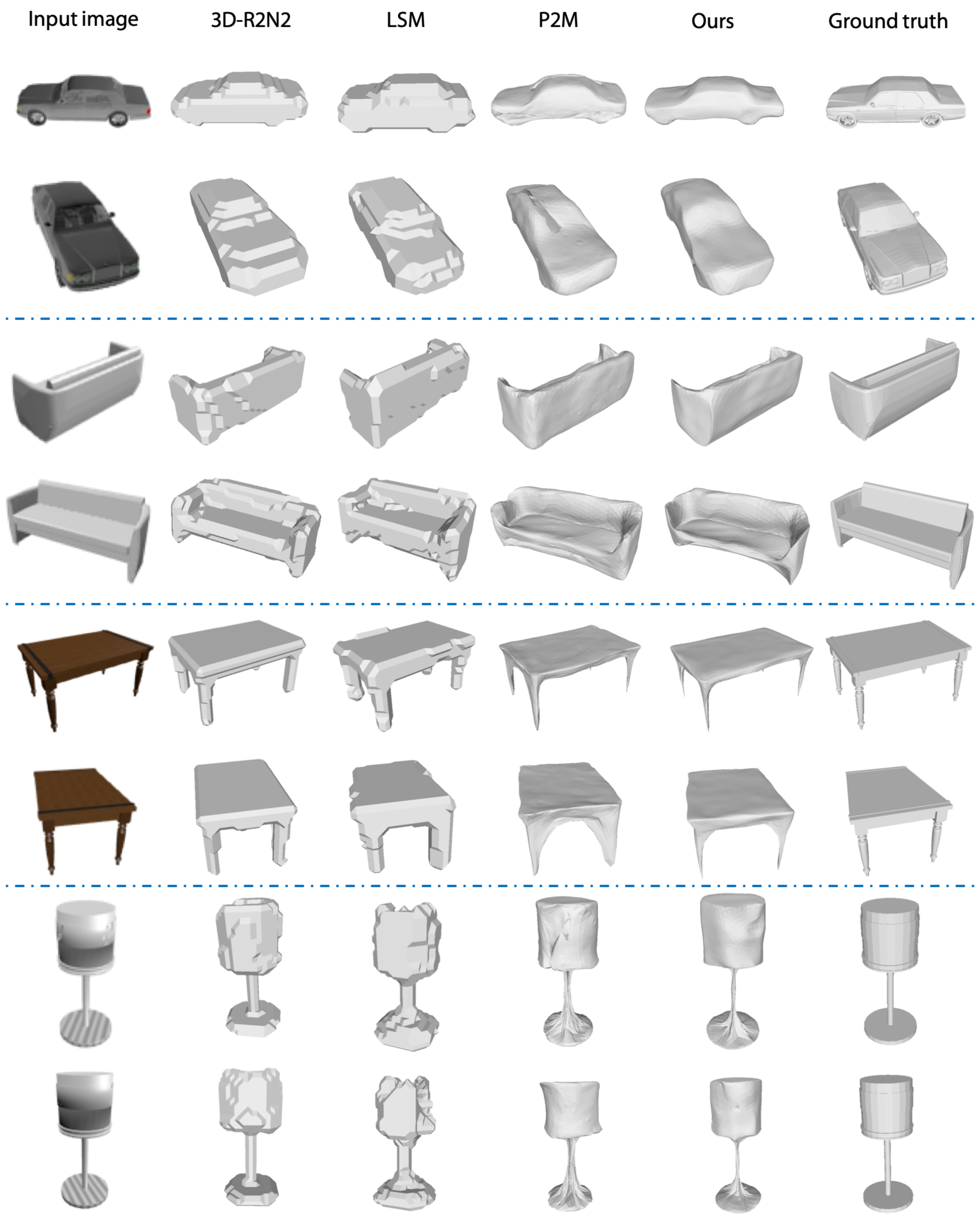

Figure 7. **More Qualitative Results.** From top to bottom, we show for each example: two camera views, results of 3DR2N2, LSM, Pixel2Mesh, ours, and the ground truth.

References

[1] icosahedron2sphere. http://3dvision.princeton.edu/pvt/icosahedron2sphere/icosahedron2sphere.m. Accessed: 2009. 1

[2] 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 *ECCV*, 2016. 4

[3] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136, 2008. 1

[4] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. 1996. 2

[5] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NeurIPS*, pages 365–376, 2017. 4

[6] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, 1987. 2

[7] David Stutz. Learning shape completion from bounding boxes with cad shape priors. http://davidstutz.de/, September 2017. 2

[8] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *CVPR*. IEEE Computer Society, 2018. 2

[9] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018. 4

[10] Wikipedia. Pentakis icosidodecahedron — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Pentakis%20icosidodecahedron&oldid=874013415, 2019. [Online; accessed 29-March-2019]. 1

[11] Jianxiong Xiao, Tian Fang, Peng Zhao, Maxime Lhuillier, and Long Quan. Image-based street-side city modeling. In *ACM transactions on Graphics (TOG)*, volume 28, page 114. ACM, 2009. 1