Supplementary Materials of NeuralIndicator: Implicit Surface Optimization with Neural Indicator Priors

Shi-Sheng Huang [1] **Guo Chen** [1] **Li Heng Chen** [1] **Hua Huang** [1]

Abstract

This supplementary materials document contains six parts: (1) we give the detail about the joint learning in Section 1, the experiments in Section 2, time analysis in Section 2.1 and discussion in Section 3, (2) Fig. 1 to 3 give more visual surface reconstruction results from DPoint dataset using different surface reconstruction approaches including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours, (3) Fig. 4 to 7 give more visual surface reconstruction results from FAMOUS dataset using different surface reconstruction approaches including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours, (4) Fig. 8 to 11 give more visual surface reconstruction results from Thingi10k dataset using different surface reconstruction approaches including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours, and (5) Fig. 12 give more visual surface reconstruction results from DPoint dataset using different surface reconstruction approaches including P2S, LIG, NDC, NKSR and Ours. (6) Fig. 13 and 14 show more visual results of 3D reconstructed geometry surface from 3D Scene dataset using PSR, PCP and Ours.

1. Joint Learning Details

We formulate a unified framework to learn the neural implicit function $f(x, \theta)$ together with the smooth indicator function χ_P in a unsupervised fashion. Our approach contain two sub-networks, including a neural implicit function $f_M(x, \theta)$ sub-branch and a smooth indicator function χ_P sub-branch, with θ and oriented point $\mathbf{P}$ are network parameters to be learnt. For χ_P sub-branch SIFG module from the oriented points $\mathbf{P}$, we extract the on-surface points from the indicator function χ_P using Flying Edges method, and project them onto the on-surface of the $f(x, \theta)$ via *differential projection*, which are further used as explicit on-surface priors. For $f(x, \theta)$ sub-branch, we adopt the encoder-decoder network from DeepSDF (Park et al., 2019) to represent the SDF. With the regularization loss from the neural indicator priors, we jointly train the χ_P and the neural implicit function $f(x, \theta)$ together in an end-to-end training manner. The final surface is extract using Marching Cubes from the zero level set of the neural implicit function $f(x, \theta)$ once convergence.

Losses. We use regularization losses mainly from our NeuralIndicator during the optimization. Specifically, we formulate the loss used in the optimization as :

$$L = L_{ind} + \lambda_1 L_{udf} + \lambda_2 L_{sdf} + \lambda_3 L_{CD} + \lambda_4 L_{ek} \quad (1)$$

including the Indicator loss L_{ind}, absolute distance loss L_{udf}, SDF loss L_{sdf}, Eikonal loss L_{ek} and CD loss L_{CD} defined above. λ_i $(i = 1, .., 4)$ are weight parameters to balance different loss items.

Oriented Point Resampling. Like Iso-Points (Wang et al., 2021), we adopt a warm-up mechanism to initialize the oriented points. Specifically, we first train the neural implicit function only within 500 epochs, and then uniformly sample 5000 on-surface points (with normal estimated) as the initial oriented point set $\mathbf{P}$. During the network learning, we also uniformly resample more on-surface points from the zero level set of indicator function after every 1000 epochs, with 5000 point number increase for each resampling.

2. More Details about the Experiments

Implementaion Details. We implemented our system using the PyTorch framework. Specifically, for the smooth indicator function generation (SIFG) in our *NeuralIndicator* component, we use the pytorch.fft package in the Pytorch framework for the Fast Fourier Transform (FFT) to the PDE solver, and generate indicator filed with $128 \times 128 \times 128$ voxel resolution in a unit voxel grid. We adopt the encoder-decoder MLP as the neural implicit function learning as used in DeepSDF (Park et al., 2019). For the Flying Edges method, we use the implementation in PyVista package (Sul-

*Equal contribution [1]School of Artificial Intelligence, Beijing Normal University, Beijing, China. Correspondence to: Hua Huang <huahuang@bnu.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

livan & Kaszynski, 2019) to extract on-surface points. During the network training, we choose Adam optimizer with an adaptive learning rate, and set the maximum epoch number as 10,000. The balancing weight parameter in our SIFG is set $\lambda = 0.01$. The parameters in the neural indicator priors are set $\lambda_1 = 0.1$, $\lambda_2 = 0.1$, $\lambda_3 = 0.05$ and $\lambda_4 = 0.01$ in all of our experiments respectively.

About How to Tune the Four Hyperparameters. Basically, we set $\lambda_1 = 0.1$, $\lambda_2 = 0.1$, $\lambda_3 = 0.05$ and $\lambda_4 = 0.01$ in all of our experiments respectively. From the above time and accuracy analysis study, we suggest keep these settings within the start warm-up stage, then we can increase λ_1, λ_2 and λ_3 for better reconstruction accuracy. If for a smooth shape, we can even set $\lambda_4 = 0$ for a better time efficiency.

Dataset. We use five public point cloud datasets, including both synthetic datasets (ABC (Koch et al., 2019), FAMOUS (Erler et al., 2020), Reconbench (Berger et al., 2013), Thingi10K (Zhou & Jacobson, 2016)) and real-scan dataset (DPoint (Wu et al., 2015) dataset), to evaluate our approach. Wherein, ABC contains variety of high-quality CAD models with mechanical parts. As like Point2Surf (Erler et al., 2020)) and SAP (Peng et al., 2021), we use the test subset (with 100 meshes) in ABC, 22 diverse meshes in FAMOUS and test subset (with 100 meshes) in Thingi10K to perform the evaluation. Since ABD, FAMOUS, Reconbench and Thingi10K have the ground truth data, we mainly use these four datasets to perform the quantitatively evaluation. For the DPoint dataset (Wu et al., 2015) consisting 20 real-scan challenging incomplete and noisy point cloud data, we choose to perform the qualitative evaluation.

Comparing Approaches. We choose five representative approaches, including PSR[1] (version 13.8), iPSR[2], Point2Mesh[3], Iso-Points[4], SAP[5], PCP[6], NeuralTPS[7] and Neural-IMLS[8]. PSR is a popular point cloud surface reconstruction approach, which can be viewed as representative traditional surface reconstruction approach without using deep learning. Iso-Points, SAP and PCP are three recent unsupervised neural implicit function learning approaches as like ours. Although Point2Mesh (Hanocka et al., 2020) didn't perform neural implicit function learning, we also choose Point2Mesh for comparison since it's a state-of-the-art unsupervised learning surface reconstruction approach.

Fair Comparison Considering Point Normal. Since the four synthetic datasets (ABC (Koch et al., 2019), FAMOUS (Erler et al., 2020), Reconbench (Berger et al., 2013), Thingi10K (Zhou & Jacobson, 2016)) providing ground-truth point normals, while the real-scan dataset (DPoint (Wu et al., 2015) dataset) didn't, for fair comparison during the subsequent quantitative and qualitative comparison, we give the same point cloud input for all of the comparing approaches. Specifically, during the comparison (both quantitative and qualitative) on the four synthetic datasets, we feed the point cloud input with ground truth point normals. For the scenarios on the DPoint dataset, we first compute their normals using principal component analysis (PCA) by ensuring the sign consistency (Tombari et al., 2010) as implemented in Pytorch3D[9], and then perform the comparison using the estimated point normals for all of the comparing approaches.

[1]https://github.com/mkazhdan/PoissonRecon
[2]https://github.com/houfei0801/ipsr
[3]https://github.com/ranahanocka/point2mesh
[4]https://github.com/yifita/iso-points
[5]https://github.com/autonomousvision/shape_as_points
[6]https://github.com/mabaorui/PredictableContextPrior
[7]https://github.com/chenchao15/NeuralTPS
[8]https://github.com/bearprin/Neural-IMLS

Table 1. Accuracy (CD1 and CD2 ($\times 10^4$)), time and memory comparison between different system variants.

Method	CD1	CD2	Time(s)	Memory(G)
w/o L_{ind}	1.39	0.70	3367	5.8
w/o L_{udf}	1.71	2.41	3095	5.4
w/o L_{sdf}	1.45	0.91	3054	5.3
w/o L_{cd}	1.55	1.26	2974	5.2
w/o L_{ek}	1.34	0.69	2147	3.1
FULL	1.33	0.67	3578	6.1

2.1. System Study

We have conducted an accuracy and computational cost study for the five losses, including the L_{ind}, L_{udf}, L_{sdf}, L_{cd} and L_{ek}. Specifically, we conducted experiments using system variants without L_{ind}, L_{udf}, L_{sdf}, L_{cd} and L_{ek} respectively. Table 1 shows the average CD1, CD2 accuracy along with the average running time and memory storage consumption. As we can see, from the accuracy perspective,L_{ind} (CD1=1.39, CD2=0.70) and L_{ek} (CD1=1.34, CD2=0.69) don't take too much influence to the final accuracy (CD1=1.33, CD2=0.67), while L_{udf} (CD1=1.71, CD2=2.41),L_{sdf} (CD1=1.45, CD2=0.91),L_{cd} (CD1=1.55, CD2=1.26) make major influence to the final accuracy. From the running time and memory storage consumption perspective, L_{ek} costs the major time (increasing from 2147s to 3578s) and memory storage consumption (increasing from 3.2G to 6.1G), while the other four losses don't take much time and memory storage compared with the full system. So L_{ek} costs the most but doesn't influence the final accuracy too much. However, according to previous approaches like Iso-Points and SAP, L_{ek} benefits better geometry details reconstruction. So for a smooth shape, we suggest remove L_{ek} for better time and memory efficiency, but for a highly-detailed shape, L_{ek} is better to be included.

[9]https://github.com/facebookresearch/pytorch3d

3. Discussion

Our current solution couldn't achieve fantastic surface reconstruction results for extremely sparse points, which would be mainly due to the insufficient constraints of the sparse points to the signed distance function learning as Iso-Points and ours used. In the further, we would like to explore more effective global shape priors to achieve much better surface reconstruction specially for sparse point cloud like wire-framed points (Huang et al., 2019). Our current framework is also not suitable to directly reconstruction from large scale scenes. Although adopting the grid-based learning strategy like LIG (Jiang et al., 2020) could successfully perform this task, extending our approach to large scale scene reconstruction is also an interesting future work.

Finally, it is also a valuable direction by introducing differential render techniques (Liu et al., 2019) into the neural implicit function optimization, which could leverage cross domain data (such as 2D images) to achieve a more realistic surface reconstruction with textures.

References

Berger, M., Levine, J. A., Nonato, L. G., Taubin, G., and Silva, C. T. A benchmark for surface reconstruction. *ACM TOG*, 32(2):20:1–20:17, 2013.

Erler, P., Guerrero, P., Ohrhallinger, S., Mitra, N. J., and Wimmer, M. Points2surf learning implicit surfaces from point clouds. In *ECCV*, volume 12350, pp. 108–124, 2020.

Hanocka, R., Metzer, G., Giryes, R., and Cohen-Or, D. Point2mesh: a self-prior for deformable meshes. *ACM TOG*, 39(4):126:1–126:12, 2020.

Huang, Z., Carr, N., and Ju, T. Variational implicit point set surfaces. *ACM TOG*, 38(4):124:1–124:13, 2019.

Jiang, C. M., Sud, A., Makadia, A., Huang, J., Nießner, M., and Funkhouser, T. A. Local implicit grid representations for 3d scenes. In *IEEE CVPR*, pp. 6000–6009, 2020.

Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., and Panozzo, D. ABC: A big CAD model dataset for geometric deep learning. In *IEEE CVPR*, pp. 9601–9611, 2019.

Liu, S., Chen, W., Li, T., and Li, H. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *IEEE ICCV*, pp. 7707–7716, 2019.

Park, J. J., Florence, P., Straub, J., Newcombe, R. A., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE CVPR*, pp. 165–174, 2019.

Peng, S., Jiang, C., Liao, Y., Niemeyer, M., Pollefeys, M., and Geiger, A. Shape as points: A differentiable poisson solver. In *NeurIPS*, pp. 13032–13044, 2021.

Sullivan, B. and Kaszynski, A. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, 4(37):1450, May 2019.

Tombari, F., Salti, S., and Di Stefano, L. Unique signatures of histograms for local surface description. In *ECCV*, pp. 356–369. Springer, 2010.

Wang, Y., Wu, S., Öztireli, A. C., and Sorkine-Hornung, O. Iso-points: Optimizing neural implicit surfaces with hybrid representations. In *IEEE CVPR*, 2021.

Wu, S., Huang, H., Gong, M., Zwicker, M., and Cohen-Or, D. Deep points consolidation. *ACM TOG*, 34(6): 176:1–176:13, 2015.

Zhou, Q. and Jacobson, A. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.

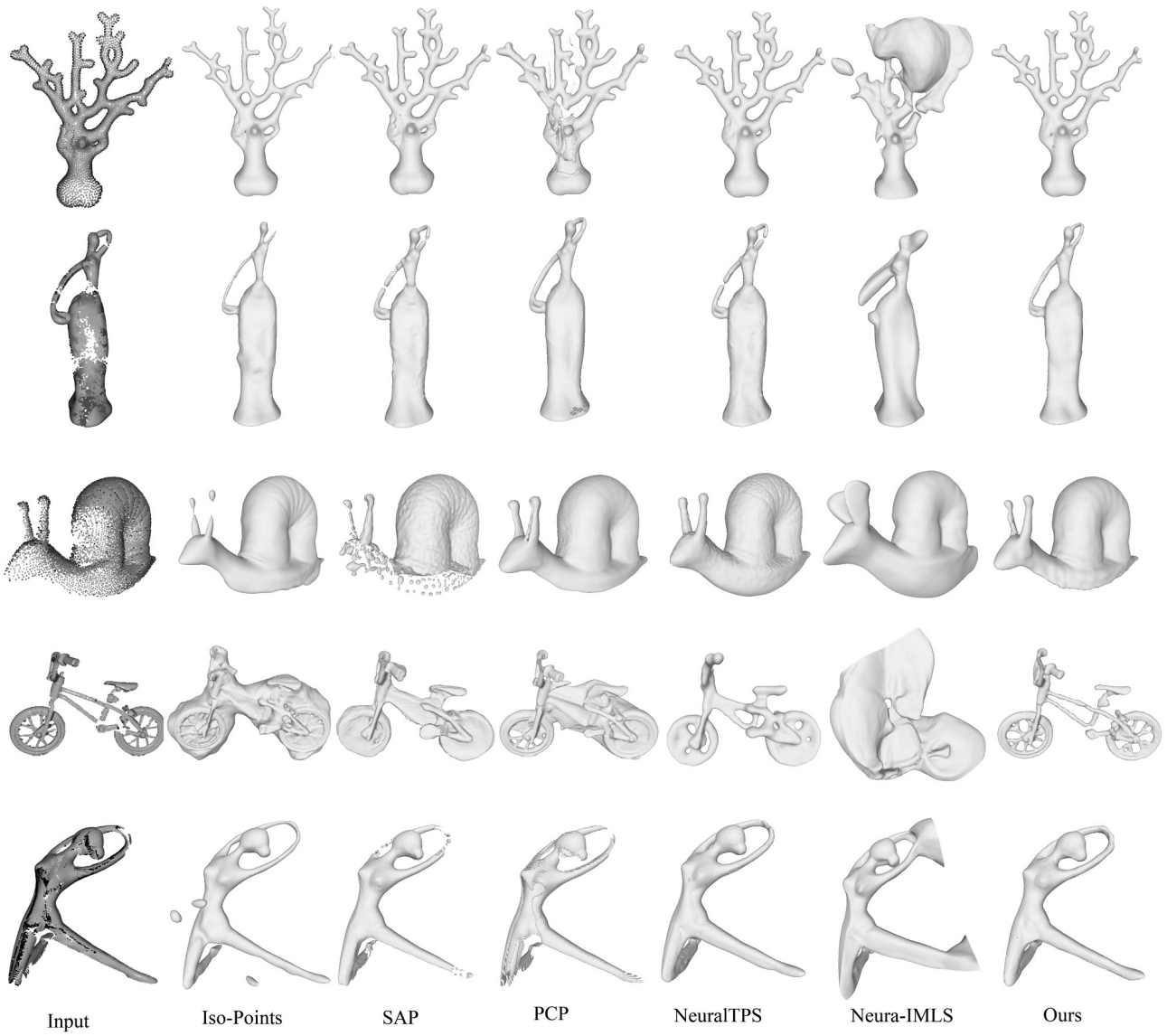

Figure 1. More visual results of 3D reconstructed geometry surface from DPoint dataset using five different surface reconstruction approaches, including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

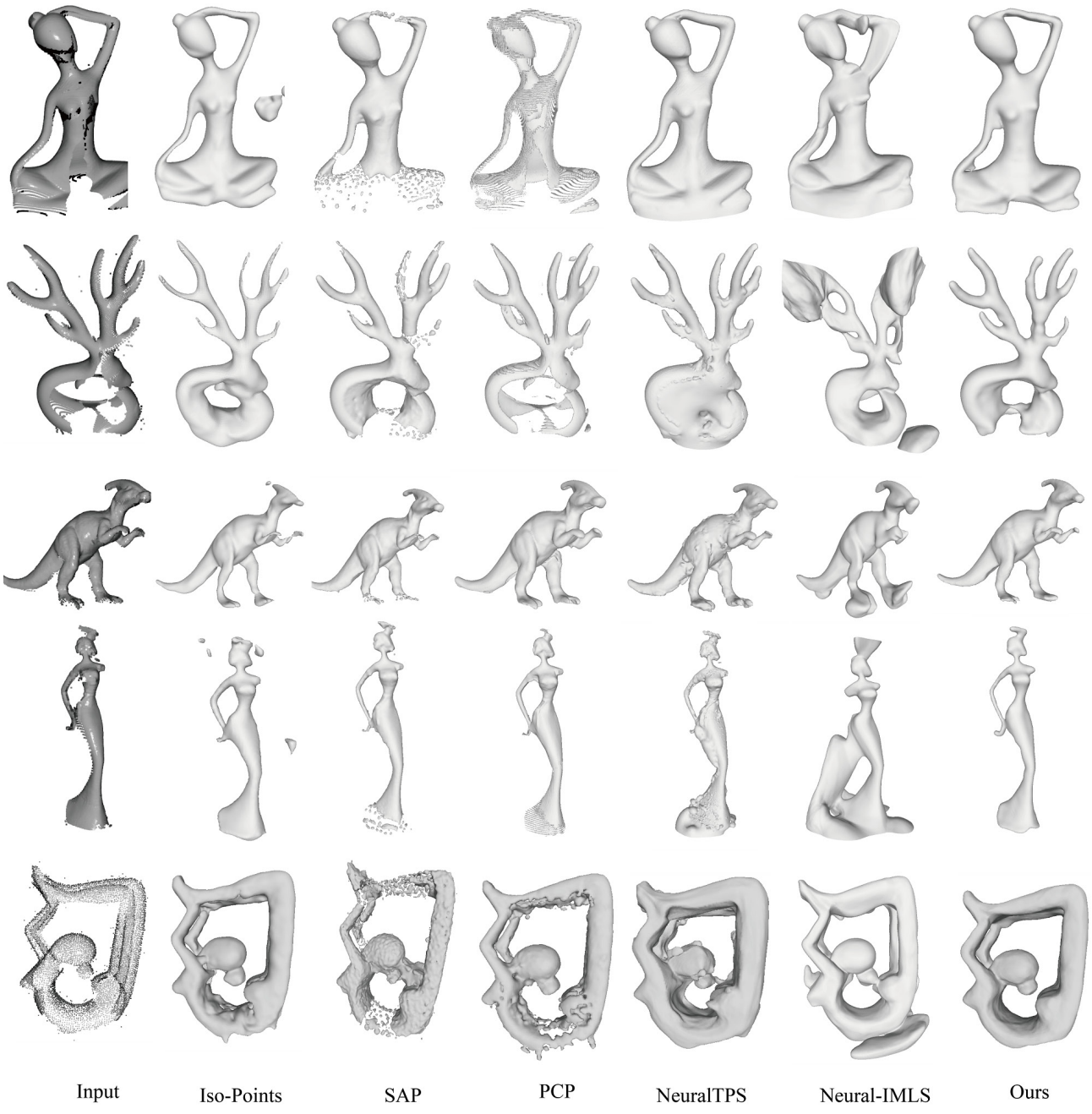

Figure 2. More visual results of 3D reconstructed geometry surface from DPoint dataset using five different surface reconstruction approaches, including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

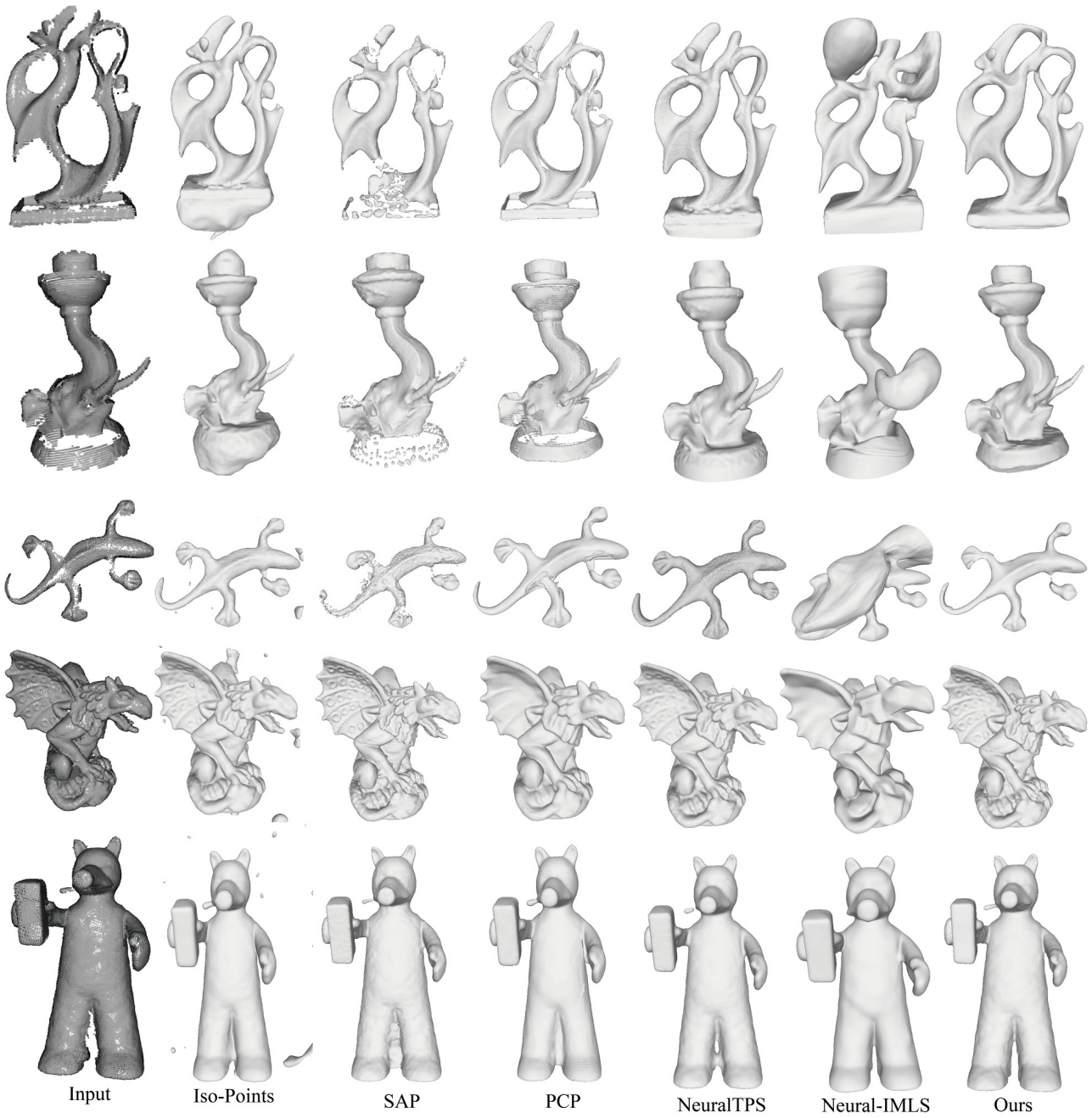

Figure 3. More visual results of 3D reconstructed geometry surface from DPoint dataset (top 3 rows) and Reconbench dataset (bottom 2 rows) using five different surface reconstruction approaches, including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

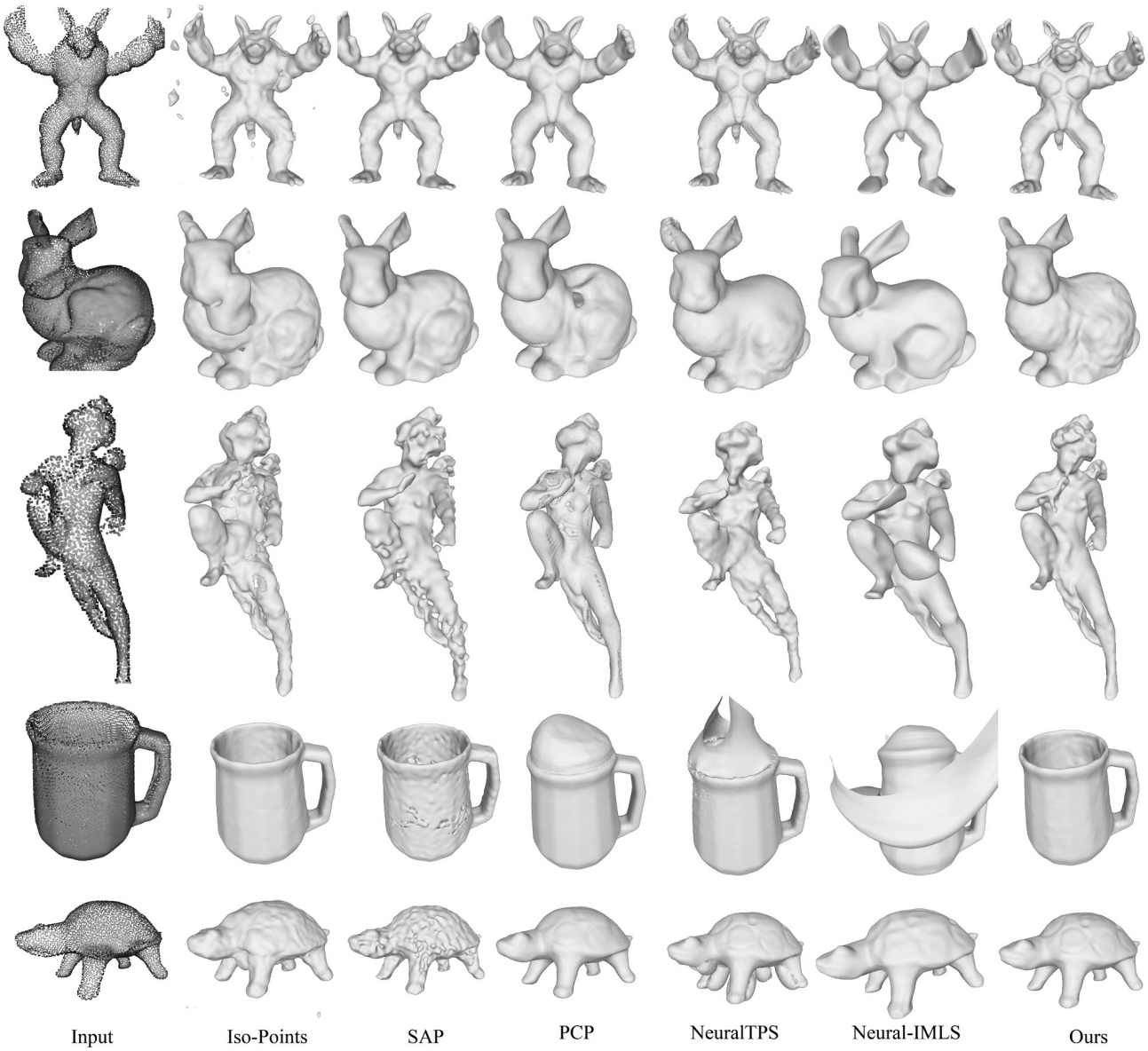

Figure 4. More visual results of 3D reconstructed geometry surface from FAMOUS dataset using five different surface reconstruction approaches, including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

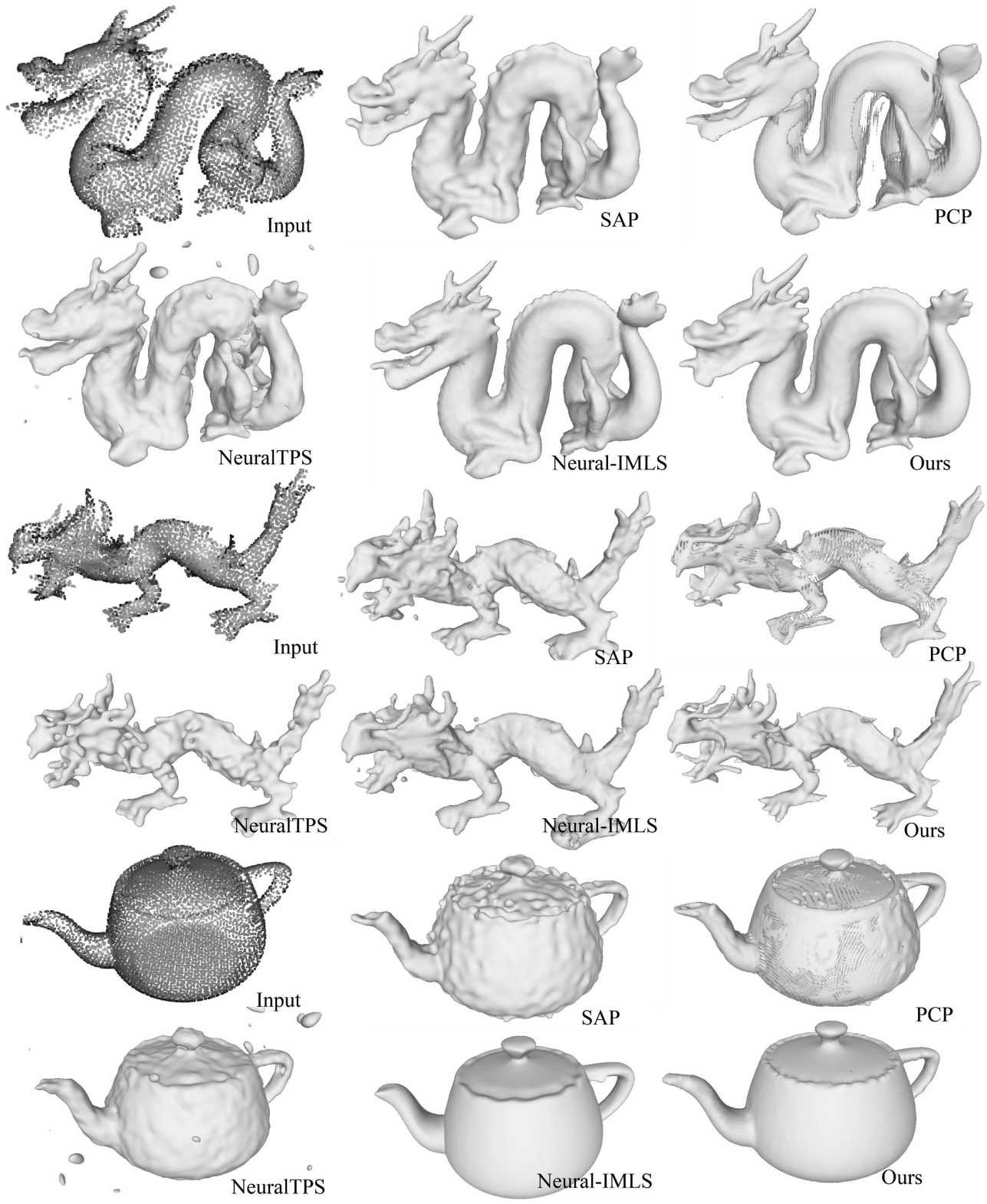

Figure 5. More visual results of 3D reconstructed geometry surface from FAMOUS dataset using five different surface reconstruction approaches, including SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

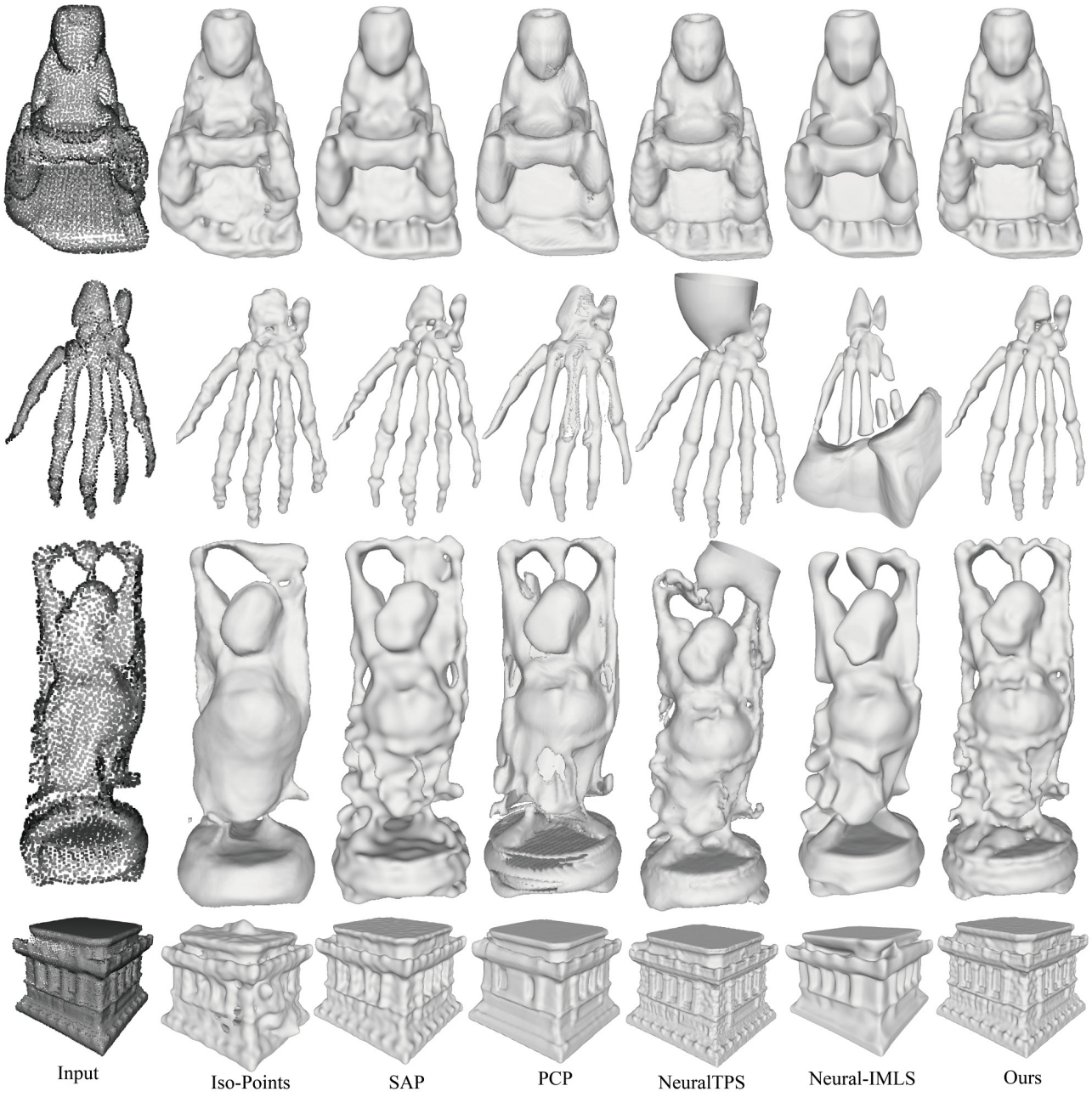

Figure 6. More visual results of 3D reconstructed geometry surface from FAMOUS dataset using five different surface reconstruction approaches, including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

Figure 7. More visual results of 3D reconstructed geometry surface from FAMOUS dataset using five different surface reconstruction approaches, including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

Figure 8. More visual results of 3D reconstructed geometry surface from Thingi10k dataset using five different surface reconstruction approaches, including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

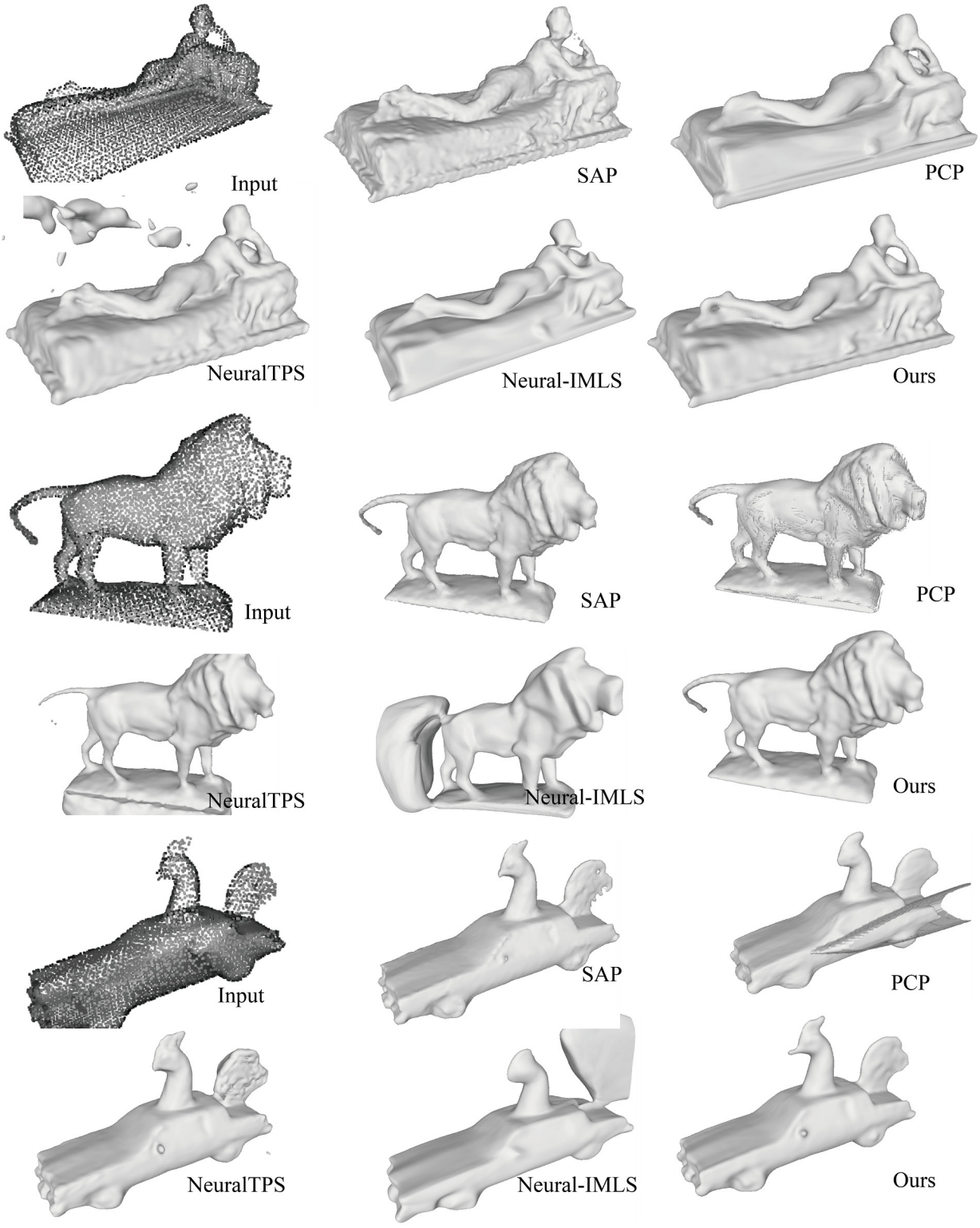

Figure 9. More visual results of 3D reconstructed geometry surface from Thingi10k dataset using five different surface reconstruction approaches, including SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

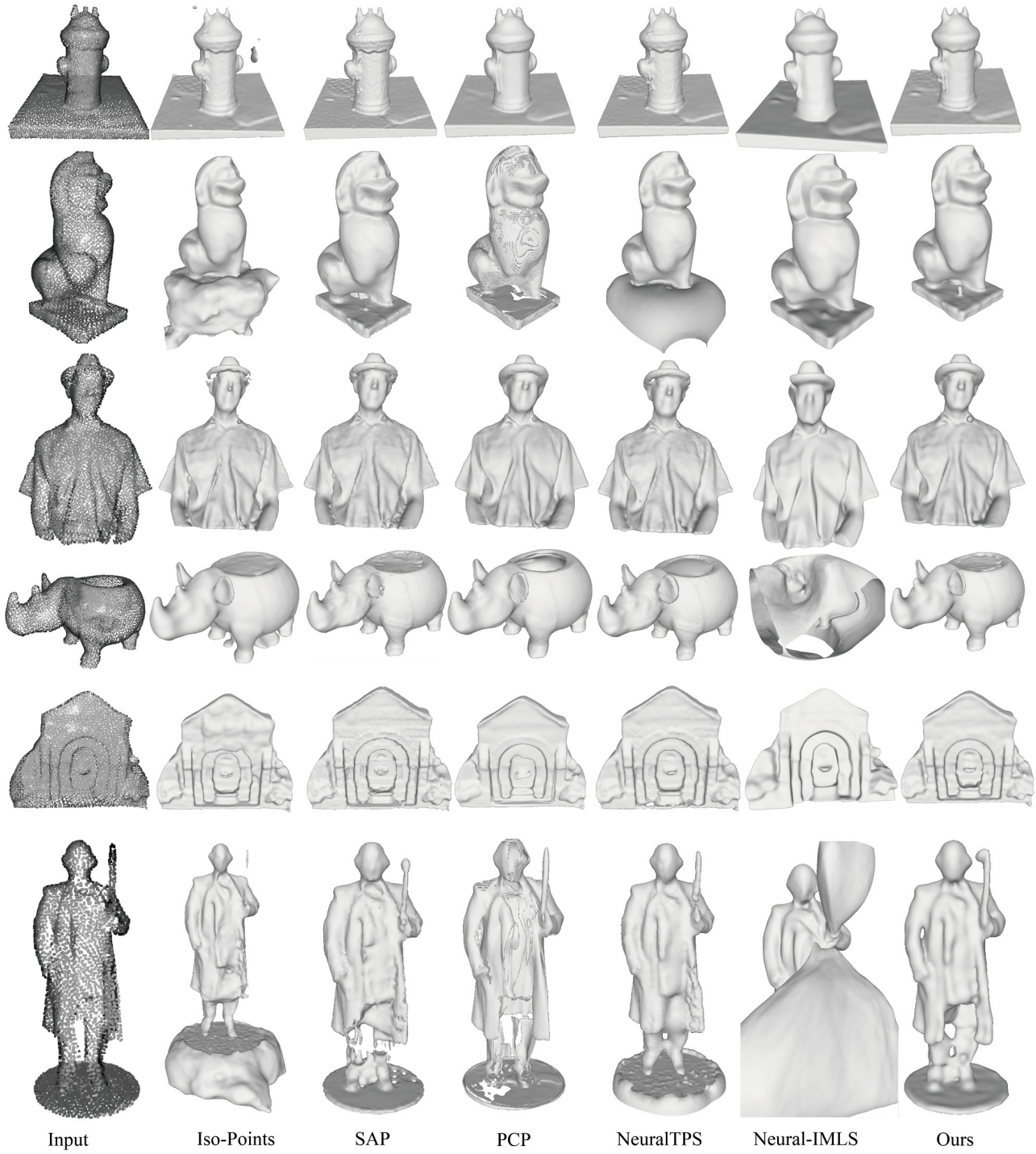

Figure 10. More visual results of 3D reconstructed geometry surface from Thingi10k dataset using five different surface reconstruction approaches, including Iso-Points, SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

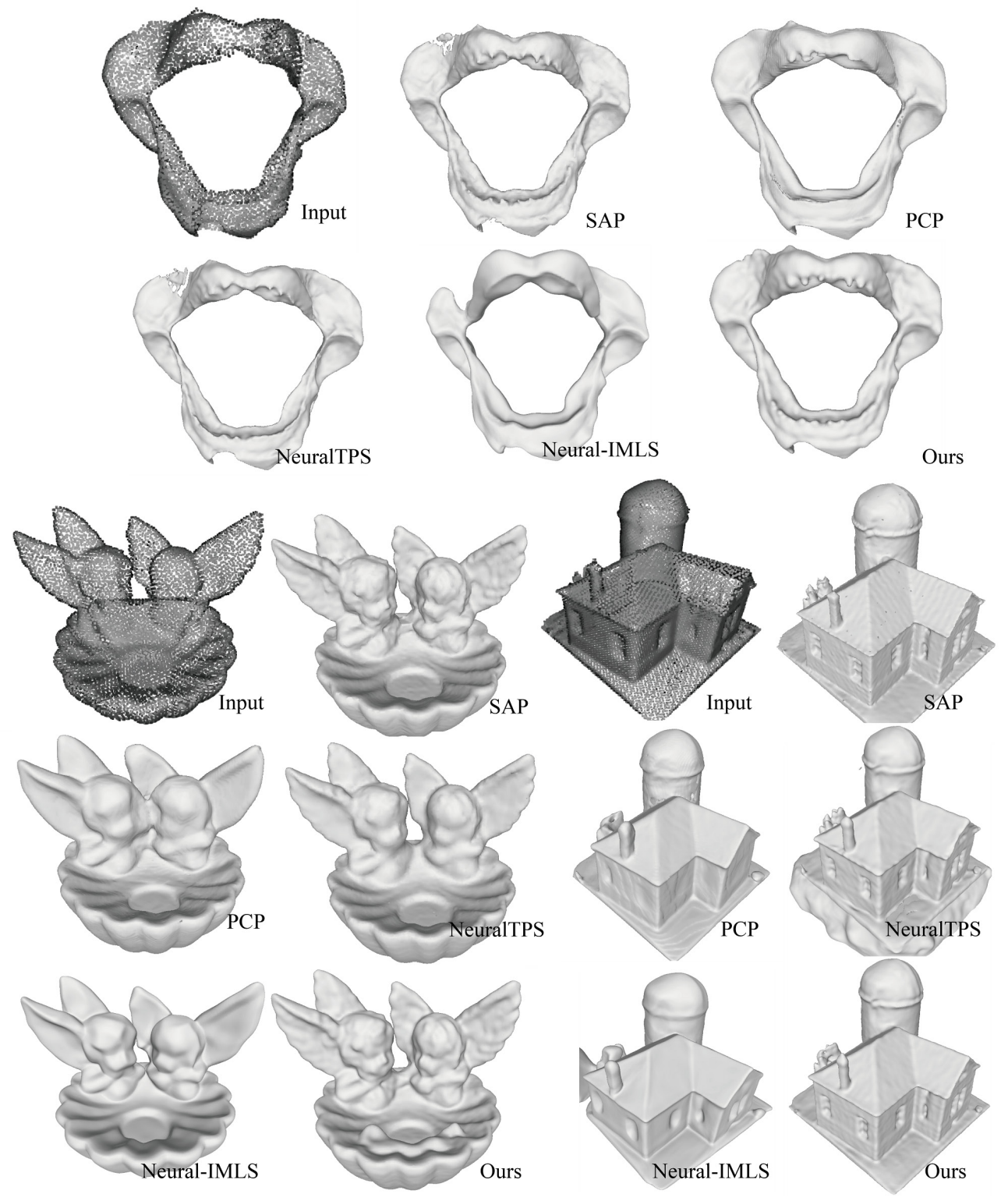

Figure 11. More visual results of 3D reconstructed geometry surface from Thingi10k dataset using five different surface reconstruction approaches, including SAP, PCP, NeuralTPS, Neural-IMLS and Ours.

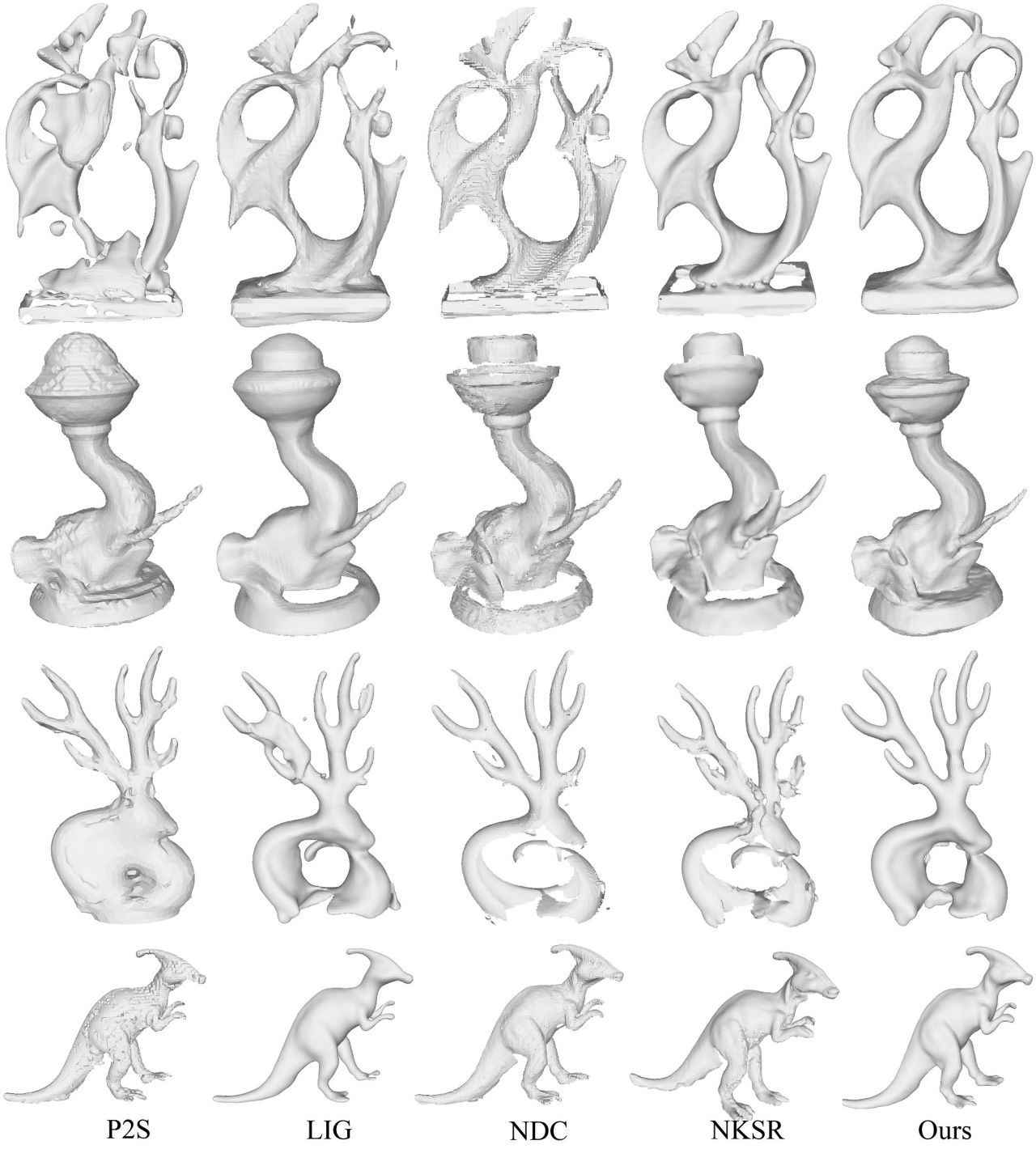

Figure 12. More visual results of 3D reconstructed geometry surface from DPoint dataset using four different surface reconstruction approaches, including P2S, LIG, NDC, NKSR and Ours.

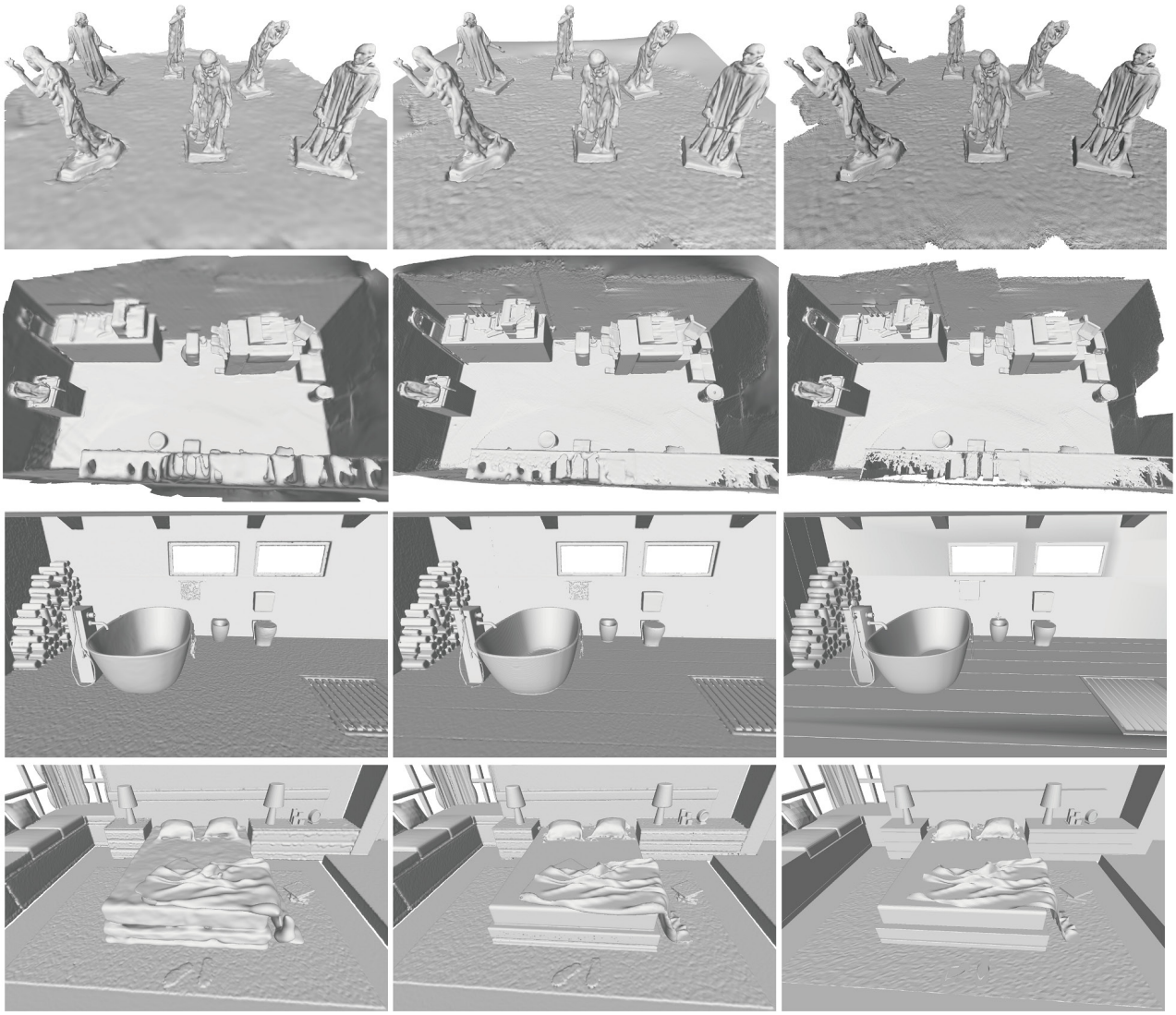

Figure 13. More visual results of 3D reconstructed geometry surface from 3D Scene dataset using PSR, PCP and Ours.

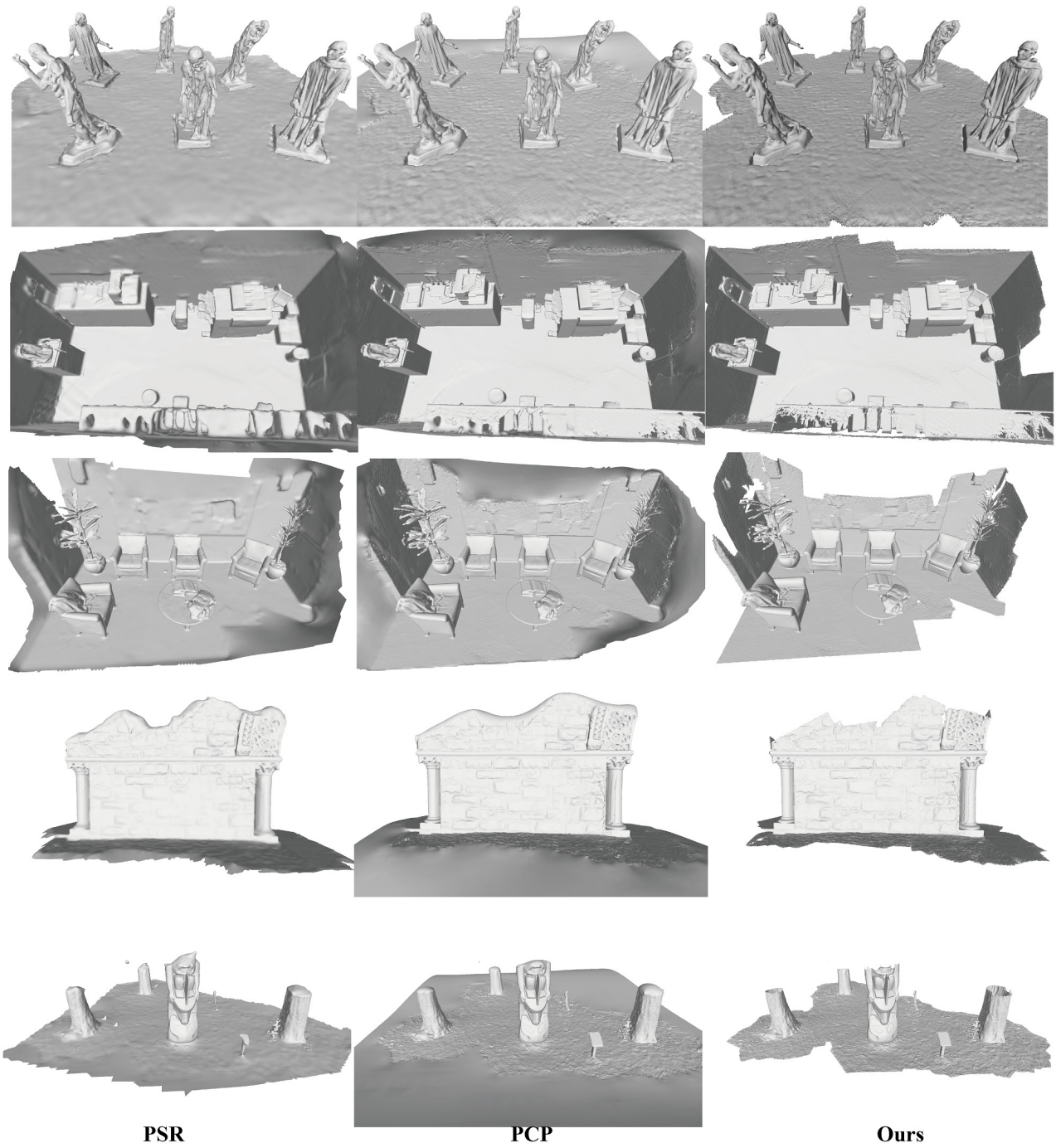

Figure 14. More visual results of 3D reconstructed geometry surface from 3D Scene dataset using PSR, PCP and Ours.