Supplementary Material for Semi-signed prioritized neural fitting for surface reconstruction from unoriented point clouds

Runsong Zhu[1*] Di Kang[2] Ka-Hei Hui[1] Yue Qian[2] Shi Qiu[1]
Zhen Dong[3†] Linchao Bao[2] Pheng-Ann Heng[1] Chi-Wing Fu[1]
[1]The Chinese University of Hong Kong [2]Tencent AI Lab [3]Wuhan University

In this supplementary material, we provide more results (Sec. 1), additional method details (Sec. 2) and experimental details (Sec. 3), and that could not be fitted into the main paper due of lack of space.

1. More results

Main examples. As shown in Supp. Fig. 2, we provide more visual comparisons (Sec 3.2 and Sec 3.3 in the main paper), including IGR [7], SAL [1], SALD [2], DiGS [4], and ours. The results demonstrate that our method is able to reconstruct more accurate surfaces while producing less artifacts. Yet, existing neural methods [1,2,4,7] occasionally generate ghost surfaces. In addition, we provide more comparison examples on Thingi10K [14], the density-variation data [8], and the noisy data [8] in Supp. Figs. 3 to 5, respectively. Besides, we provide visual comparisons with more baseline methods on the examples shown in the main paper; please see Supp. Figs. 6 to 10.

Ablation experiment. We provide more visual comparisons for the ablation study in Supp. Figs. 11 and 12, in which the transparent visualization results show that removing the signed supervision wrosens the accuracy on thin structures, thereby leading to undesired surfaces.

Generalization on scene-level data. We provide the qualitative comparison on one example from 3D Scene dataset [15]. As shown in Fig. 1, our methods could reconstructed more accurate surface compared with DiGS [4].

Results on Surface Reconstruction Benchmark [13] data. We conduct the experiments on the Surface Reconstruction Benchmark (SRB) [13] data. Note that, we randomly sample $50,000$ points, each on the reconstructed mesh and the ground-truth dense point clouds, respectively, to calculate the metrics, following SAP [12]. The quantitative results are provided in Supp. Tab. 1 and the visualization results are shown in Supp. Fig. 13. Overall, SSP achieves comparable performance with the current SOTA methods (DiGS [4]) on SRB [13]. We notice that the input point clouds in SRB [13] contain some missing regions, leading to unreliable unsigned supervisions. In this case, SSP may wrongly prioritize the optimization towards the missing regions according to the tracked loss values, thereby causing large errors. See the top example in Supp. Fig. 13. We regard this as a limitation of our method and leave this as a future work to introduce the data prior for accurate surface reconstruction from inputs with missing regions.

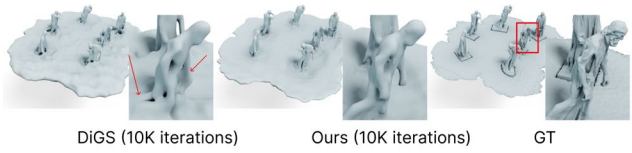


Figure 1. Visual comparison on 3D Scene [15].

2. Method details

2.1. Space partitioning

We use the same rule to determine the voxel size for all the shapes (ABC subset [6], Thingi10k [14], noisy data [8], density-varying data [8], and Surface Reconstruction Benchmark [13]). More concretely, for any given shape (represented in the point cloud format and normalized to fit into [-0.9, 0.9]), we first find the "distances" from each point to its 50th closest point. Then, we calculate the average "distance" (termed as "density_indicator") from all distance values since the average distance reflects the sampling density of a given shape. We use $10 * \mathrm{round}(1/(1.5 * \mathrm{density\_indicator} * 10))$ as the final voxel grid resolution. Note that we make sure the resolution is divisible by 10 for convenience. Then we mark a voxel as "occupied" if it contains at least one point. We provide the pseudo code of the space partitioning algorithm (Sec. 3.2 in the main paper) in Supp. Algo. 1.

*Work partially done during an internship at Tencent AI Lab.
†Corresponding author.

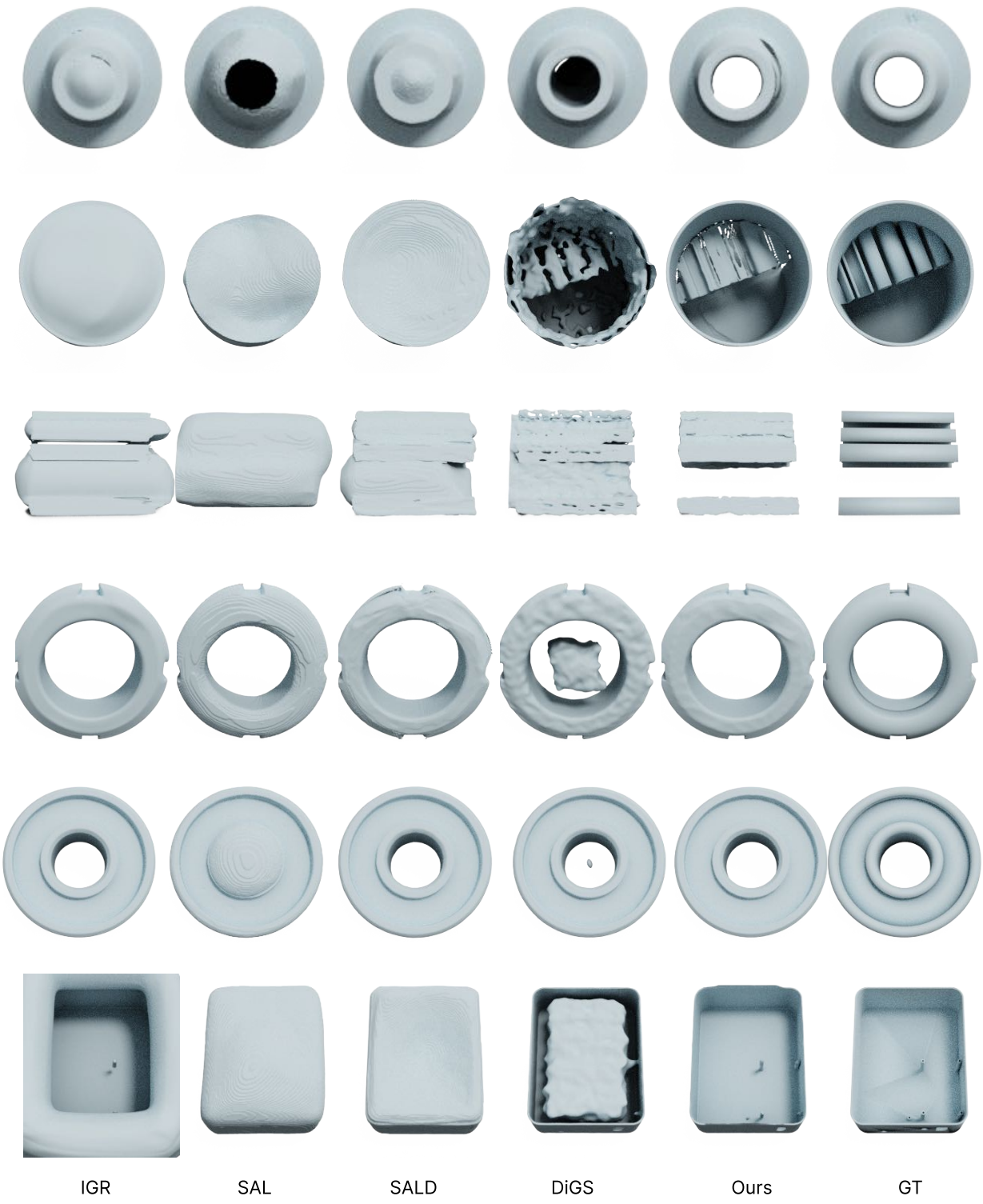


Figure 2. Visual comparisons for neural methods (IGR [7], SAL [1], SALD [2], and DiGS [4]) on the ABC subset [6].

2.2. Loss-based per-region sampling

Region-wise loss tracking. Different losses (i.e., $\mathcal{L}_{\text{dist}}^{\text{on}}$, $\mathcal{L}_{\text{dist}}^{\text{off}}$, $\mathcal{L}_{\text{grad}}^{\text{on}}$, $\mathcal{L}_{\text{grad}}^{\text{off}}$, $\mathcal{L}_{\text{signed}}$) are applicable to different voxels (Sec. 3.3 in the main paper). Specifically, we track $\mathcal{L}_{\text{dist}}^{\text{on}}$ and $\mathcal{L}_{\text{grad}}^{\text{on}}$ for the voxels that contain points. We track $\mathcal{L}_{\text{dist}}^{\text{off}}$ and $\mathcal{L}_{\text{grad}}^{\text{off}}$ for all voxels in region $\mathcal{V}_{\text{uncertain}}$. We track $\mathcal{L}_{\text{signed}}$ for all voxels in region $\mathcal{V}_{\text{known}}$.

Algorithm 1: Space Partitioning (Python style)

Input : 3D array V_{occ}, indicating if a voxel is occupied
Output : 3D array V_{outside}, indicating if a voxel is outside the object for sure
// shape: $N \times N \times N$

Initialize a visited list V_v with 0;
outside = {};

// Step 1: find all empty voxels on the six faces of the cube and label them as "outside"

// test voxels on two opposite faces of the cube
for i *in* $\{1, N\}$ **do**
  for $j \leftarrow 1$ **to** N **do**
    for $k \leftarrow 1$ **to** N **do**
      if $V_{\text{occ}}[i, j, k] == 0$ *and* $V_v[i, j, k] == 0$ **then**
        $V_{\text{outside}}[i, j, k] = 1$;
        outside.append([i,j,k]);
      end
      $V_v[i, j, k] = 1$;
    end
  end
end
// test the other two sets of opposite faces similarly
⋮
// Step 2: a BFS-like procedure to recursively find all "outside" voxels connected to the initial outside voxels on the six faces
offset $= \{[1,0,0], [-1,0,0], [0,1,0], [0,-1,0], [0,0,1], [0,0,-1]\}$;
while index < outside.*size()* **do**
  cur_pos = outside [i];
  for $i \leftarrow 1$ **to** 6 **do**
    // test all six faces
    neighbor_pos = {cur_pos [1]+ offset [i][1],cur_pos [2]+ offset [i][2],cur_pos [3]+ offset [i][3]};
    if check_valid *(*neighbor_pos*)* *and* V_{occ}[neighbor_pos] $== 0$ *and* V_v[neighbor_pos] $== 0$ **then**
      // if this is a valid && occupied && not visited position
      V_{outside}[neighbor_pos]= 1;
      // add the connected outside voxel
      outside.append(neighbor_pos);
    end
    V_v[neighbor_pos] $= 1$;
    index $=$ index $+ 1$
  end
end
return V_{outside}

Adaptive sampling. As mentioned in the main paper (Sec 3.3), we perform a two-step sampling for each loss: (i) adaptively sample a set of voxels based on the previous region losses and (ii) sample points within each region (voxels). Specifically, for the sampling of on-surface points and off-surface points, we additionally randomly sample voxels to better cover the whole $V_{\text{uncertain}}$ region. Besides, we keep a similar number of sample points (34588) as IGR [7] (34816). Specifically, we set $16384 * \frac{7}{9}$ for the sample number of on-surface points, $16384 * \frac{1}{3}$ as the sample number of off-surface points in region $V_{\text{uncertain}}$ and 16384 as the sample number in the outside region V_{outside}.

2.3. Motivation for using derivative supervision in the free space

According to the following proposition, the normal of point q's nearest point p' (denoted as $\frac{\nabla f_\theta(p')}{||\nabla f_\theta(p')||}$) on the underlying continuous surface equals to the derivative of q. Empirically, we found it helpful to utilize the normal of the nearest point p (denoted as n_p) in P to approximate $\frac{\nabla f_\theta(p')}{||\nabla f_\theta(p')||}$ as a supervision for point q in the free space.

Proposition 1 *Given a random point q in the free space and its closest point on the underlying surface p', we have $\frac{\nabla f(p')}{||\nabla f(p')||} = \frac{\nabla f(q)}{||\nabla f(q)||}$, where the derivative always exists for f.*

The proof: we first consider point q that is inside the underlying shape, which means $f(q) < 0$.

(1) First, we prove the following proposition:

$$\frac{\nabla f(p')}{\|\nabla f(p')\|} = \frac{q - p'}{\|q - p'\|}, \tag{1}$$

where $\|\cdot\|$ is L2 norm. Since p' is q's closest point on the underlying surface defined by $f(x) = 0$, then

$$p' = \underset{\mathrm{x}}{\operatorname{argmin}} \|x - q\| \;\; s.t. \;\; f(x) = 0.$$

And the corresponding Lagrange function is

$$L(x) = \|x - q\| + \lambda f(x),$$

whose gradient can be calculated as

$$\nabla_{(p',\lambda)} L = \left(\frac{-(q - p')}{\|q - p'\|} + \lambda \nabla f(p'), f(p') \right).$$

Let $\frac{-(q-p')}{\|q-p'\|} + \lambda \nabla f(p') = 0$ and we get $\nabla f(p') = \lambda \frac{q-p'}{\|q-p'\|}$. Hence, we know that $\nabla f(p')$ is collinear to vector $(q - p')$. Since the gradient denotes the direction of the greatest increase of the function, we have $\frac{\nabla f(p')}{\|\nabla f(p')\|} = \frac{q-p'}{\|q-p'\|}$.

(2) Then, we can prove that

$$\frac{\nabla f(q)}{\|\nabla f(q)\|} = \frac{q - p'}{\|q - p'\|} = \frac{\nabla f(p')}{\|\nabla f(p')\|}. \tag{2}$$

According to the definition of directional derivative,

$$\nabla_v f(x) = \lim_{h \to 0} \frac{f(x + hv) - f(x)}{h},$$

where v is the given unit vector. The relation between gradient and directional derivative at point q can be written as

$$\frac{\nabla f(q)}{\|\nabla f(q)\|} = \underset{v}{\operatorname{argmax}} \nabla_v f(q)$$

and

$$\|\nabla f(q)\| = \max_v \nabla_v f(q).$$

Since the signed distance function should satisfy $\|\nabla f(q)\| = 1$, we can know that the directional derivative always satisfies $\nabla_v f(q) \leq \|\nabla f(q)\| = 1$ for any direction v. Moreover, by definition, $\nabla_v f(q)$ can be expressed as

$$\nabla_v f(q) = \lim_{h \to 0} \frac{f(q + hv) - f(q)}{h}.$$

We can consider the specific direction $v' = \frac{q-p'}{\|q-p'\|}$ and let $q^* = q + hv'$. We also denote

$$p'' = \underset{p s.t. f(p)=0}{\operatorname{argmin}} \|q^* - p\|.$$

We can prove that $p'' = p'$ by contradiction, i.e., if $p'' \neq p'$, then $\|q^* - p''\| < \|q^* - p'\|$, we add the same value on each side of the inequality, $\|q^* - q\| + \|q^* - p''\| < \|q^* - q\| + \|q^* - p'\| = \|q - p'\|$, since q^* locates in the line segment (q, p'). However, $\|q^* - q\| + \|q^* - p''\| \geq \|q - p''\| > \|q - p'\|$ which leads to contradiction. Therefore, $p'' = p'$.

Thus,

$$\begin{aligned} \nabla_{v'} f(q) &= \lim_{h \to 0} \frac{f(q^*) - f(q)}{h} \\ &= \lim_{h \to 0} \frac{-1 * \|q^* - p'\| - (-1) * \|q - p'\|)}{h} \\ &= 1. \end{aligned}$$

Hence, the direction v' yields $\underset{v}{\operatorname{argmax}} \nabla_v f(q)$. Therefore, we obtain

$$\frac{\nabla f(q)}{\|\nabla f(q)\|} = \frac{q - p'}{\|q - p'\|}. \tag{3}$$

In the meantime, if q is outside the shape, it can be proved in a similar way that

$$\frac{\nabla f(p)}{\|\nabla f(p)\|} = \frac{\nabla f(q)}{\|\nabla f(q)\|} = \frac{-(q - p')}{\|q - p'\|}.$$

3. Experimental details

Implementation details of the proposed semi-signed prioritized (SSP) neural fitting. We adopt the MLP architecture that contains eight hidden layers, each with 512 hidden units, to represent the SDF function as IGR [7]. Besides, we adopt Geometry Network Initialization (GNI) [1] to initialize the implicit SDF function to a rough unit sphere. We use ADAM optimizer [10] with an initial learning rate of 0.005 and schedule the learning rate to decrease by a factor of 0.5 every 2000 epochs. For the whole experiment except for the noisy data (*i.e.*, ABC subset [6], Thingi10K [14], density-varying data [8], noisy data [8], and Surface Reconstruction Benchmark [13]) , the weight

Methods	SRB [13]	
	F-score ↑	CD-L_1 ↓ ($\times 100$)
SPSR [9]	0.735	2.232
SAP [12]	0.830	0.760
IMLS [11]	0.821	0.779
POCO [5]	0.858	0.760
N-P [3]	0.658	0.963
SAL [1]	0.831	0.909
SALD [2]	0.595	1.779
IGR [7]	0.723	2.794
DiGS [4]	**0.876**	0.686
Ours	0.874	**0.660**

Table 1. Quantitative results on SRB [13]. Note that the SRB dataset does not contain GT normal to calculate the normal consistency (NC) metric.

parameter w_i is $\{40, 20, 1, 1, 1, 10\}$ in Eq. (8) in the main paper. For the noisy data, the weight parameter is adjusted to $\{20, 10, 20, 10, 1, 10\}$ to rely less on the distance losses ($\mathcal{L}^{\text{on}}_{\text{dist}}, \mathcal{L}^{\text{free}}_{\text{dist}}$) and more on the gradient losses ($\mathcal{L}^{\text{on}}_{\text{grad}}, \mathcal{L}^{\text{free}}_{\text{grad}}$).

Optimization setting. We list the optimization epochs and fitting time for different methods in Supp. Tab. 1. We optimize our method on each shape for $10K$ epochs. For DiGS [7], we optimize the neural network using the default maximum number of epoch, *i.e.*, $10K$. For Neural-Pull [3], we optimize the neural network using the default maximum number of epoch $40K$ as in their official code. We optimize IGR [7], SAL [1], and SALD [2] for $20K$ epoch to keep a similar optimization time as our method. Since SALD [2] did not release the code for surface reconstruction, we implement the code based on SAL [1] by adding the on-surface derivative supervision [2] with the default loss weight according to the SALD [2] paper. For SAP [12], we just follow the provided optimization-based setting to report the results.

Method	Ours	DiGS [4]	IGR [7]	SAL [1]	SALD [2]	NP [3]
Epoch	10K	10K	20K	20K	20K	40K
Time	∼15	∼15	∼27	∼20	∼22	∼15

Table 2. The optimization epochs & approximate time (in minutes) for different optimization-based neural methods.

Evaluation metrics. We follow the same procedure as in SAP [12] to calculate the metrics, including the L1-based Chamfer Distance (denoted as CD-L_1), the Normal Consistency (NC), and the F1-score with default threshold 1% in Sec. 4.1 of the main paper. Specifically, for the ABC subset [6], we randomly sample $40,000$ points, each on the reconstructed mesh and the ground-truth mesh, respectively, to calculate the aforementioned metrics. For Thingi10K [14], we randomly sample $50,000$ points, each on the reconstructed mesh and the ground-truth mesh, respectively, to calculate the metrics, following SAP [12]. For the density-varying data and noisy data, we directly use the provided clean point cloud points (containing $100,000$ points) as GT and randomly sample the same number of points on the reconstructed meshes to calculate the metrics.

Details for applying proposed signed supervision to existing methods. To apply our proposed supervision, we use the space partition algorithm in Supp. Sec. 2.1 to find the V_{known} region. Then, we randomly sample points in the V_{known} region to apply the signed guidance in addition to the original constraints of the official methods (*i.e.*, IGR [7], DiGS [4]) without any other change. Specifically, we set the number of the sample points in the V_{known} region as 16348 for IGR [7] and DiGS [4].

References

[1] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 1, 2, 4, 5

[2] Matan Atzmon and Yaron Lipman. Sald: Sign agnostic learning with derivatives. In *International Conference on Learning Representations*, 2020. 1, 2, 5

[3] Ma Baorui, Han Zhizhong, Liu Yu-shen, and Zwicker Matthias. Neural-Pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. 2021. 5

[4] Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. DiGS: Divergence guided shape implicit neural representation for unoriented point clouds. In *CVPR*, 2022. 1, 2, 5

[5] Alexandre Boulch and Renaud Marlet. Poco: Point convolution for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6302–6314, 2022. 5

[6] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J. Mitra, and Michael Wimmer. Points2Surf: Learning implicit surfaces from point clouds. In *ECCV*, 2020. 1, 2, 4, 5, 7, 10, 11, 12, 15, 16

[7] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. 2020. 1, 2, 3, 4, 5

[8] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. PCPNet: Learning local shape properties from raw point clouds. In *Comput. Graph. Forum*. Wiley Online Library, 2018. 1, 4, 8, 9, 13, 14

[9] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. 2013. 5

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

[11] Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep implicit moving least-squares functions for 3D reconstruction. In *CVPR*, 2021. 5

[12] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. In *NeurIPS*, 2021. 1, 5

[13] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *CVPR*, 2019. 1, 4, 5, 17

[14] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3D-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 1, 4, 5

[15] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics (ToG)*, 32(4):1–8, 2013. 1

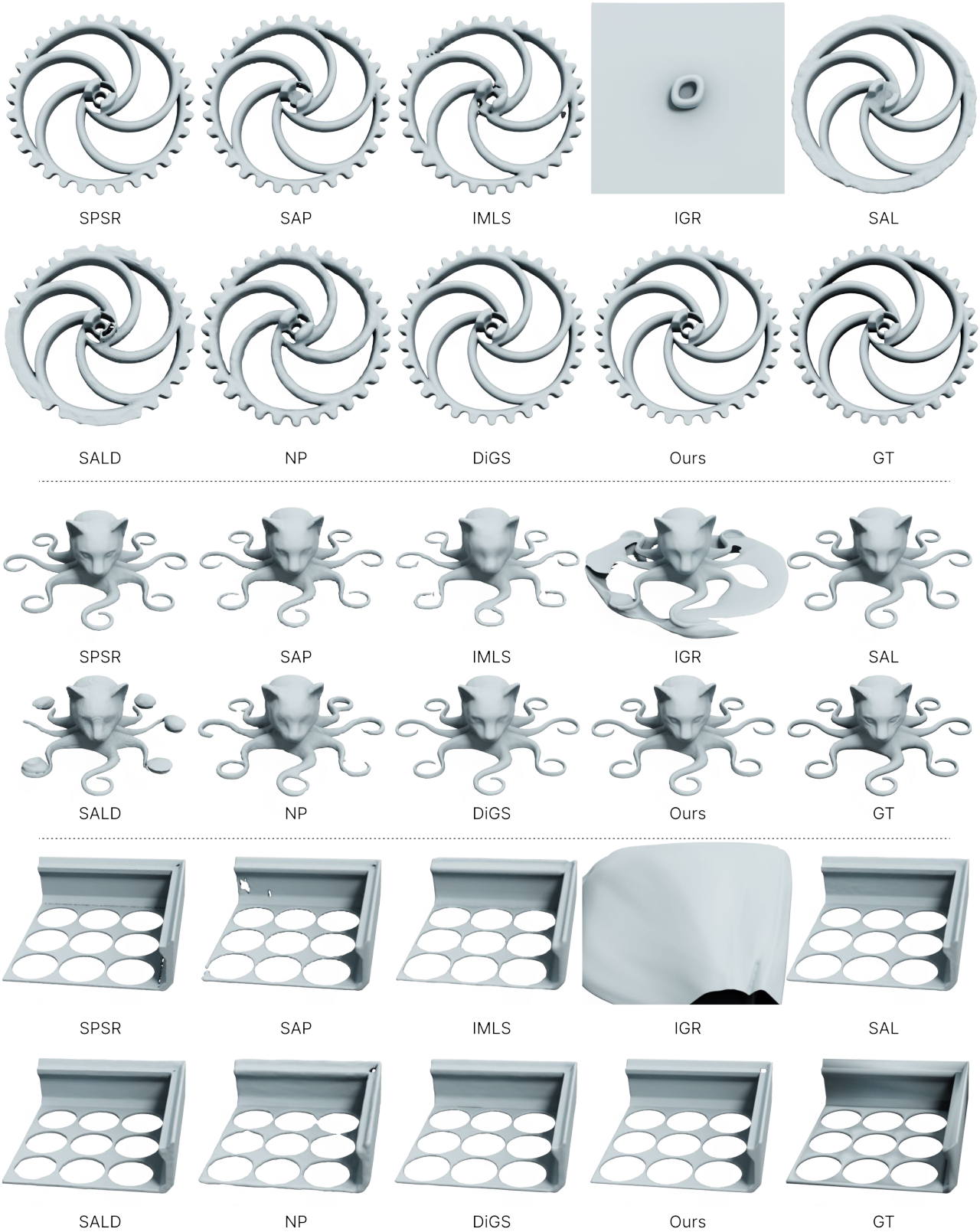


Figure 3. Visual comparisons on Thingi10K [6].

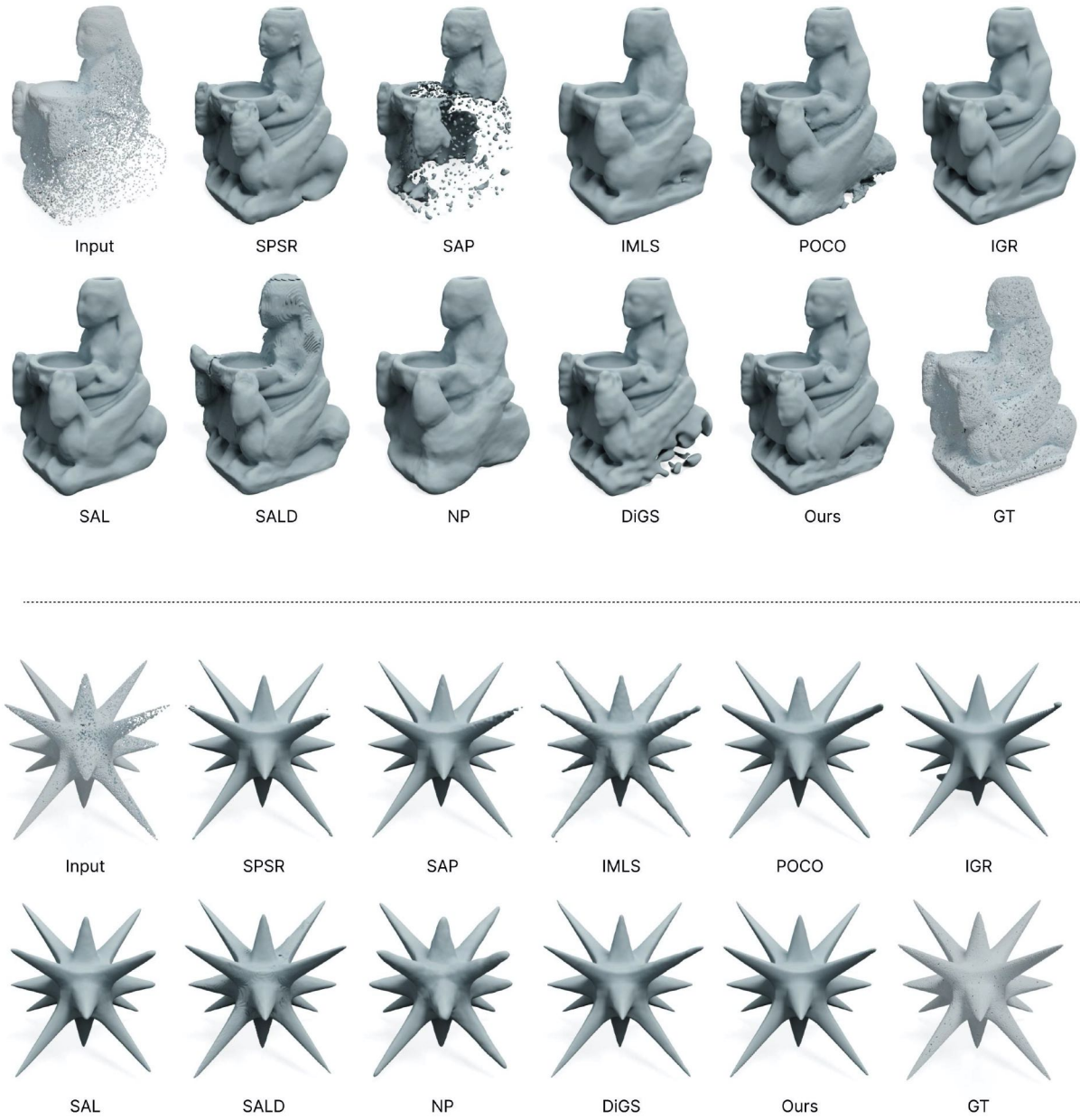
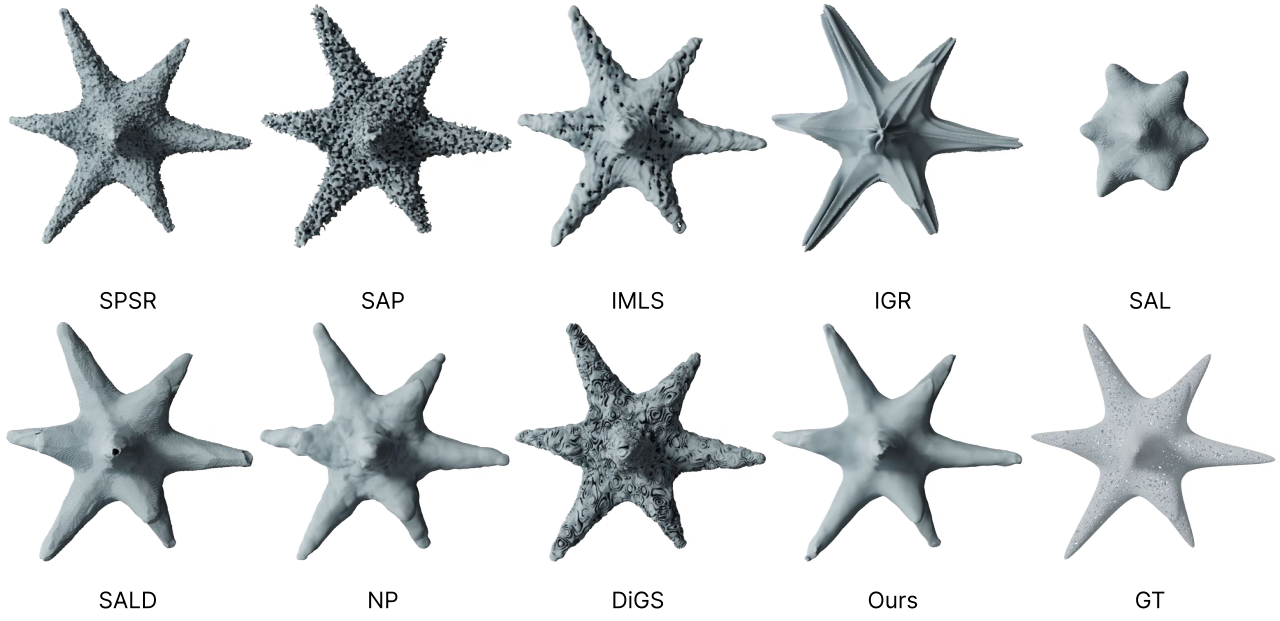


Figure 4. Visual comparisons on the density-variation data [8].



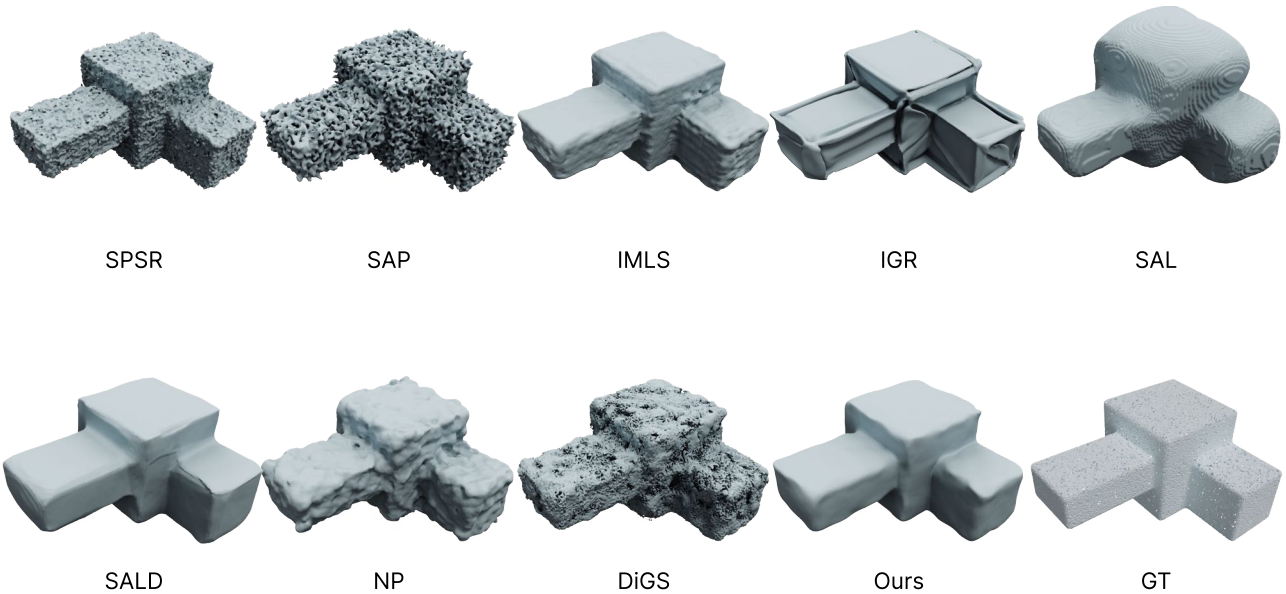


Figure 5. Visual comparisons on the noisy data [8].

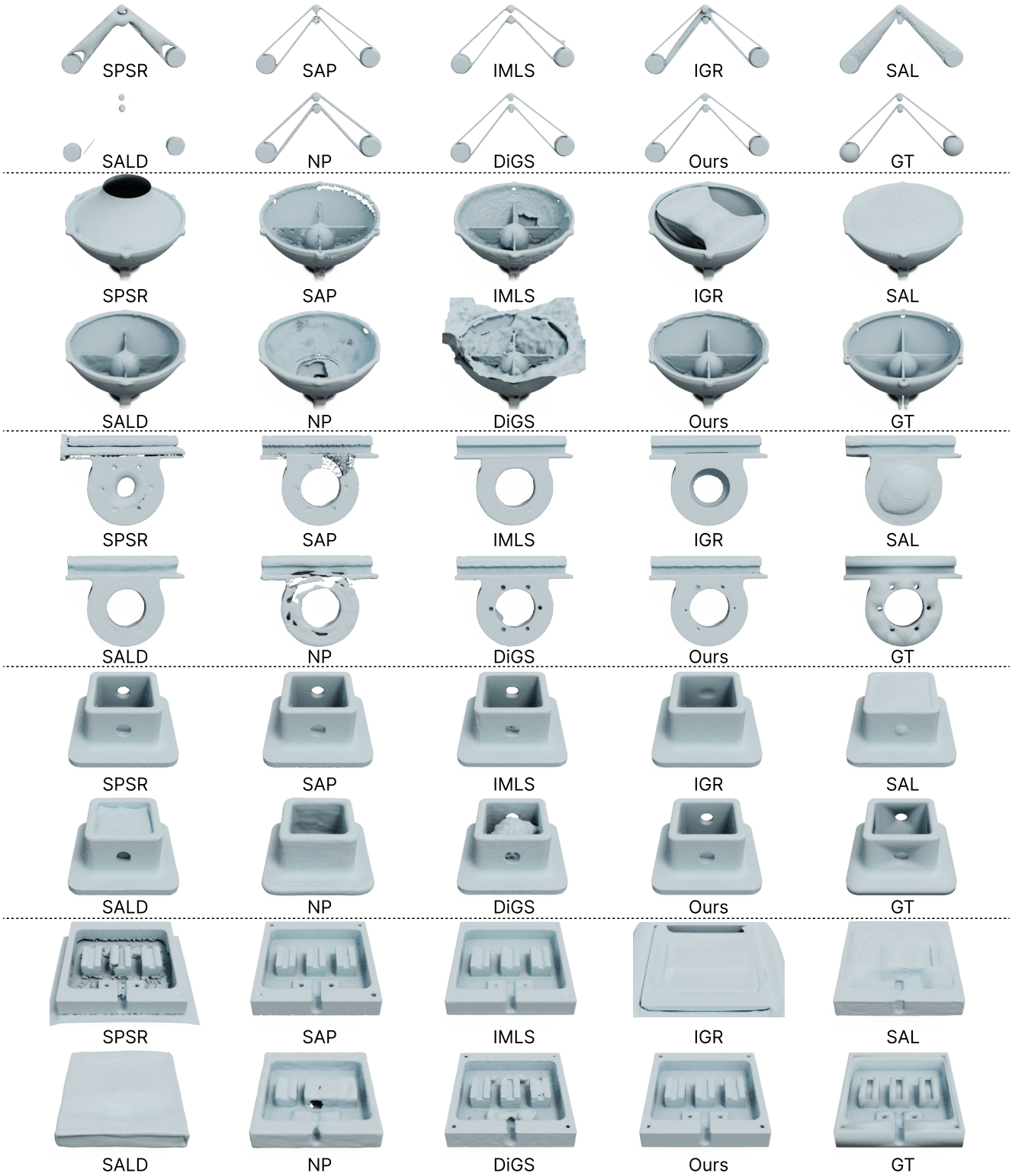


Figure 6. Visual comparisons with more methods on the ABC subset [6].

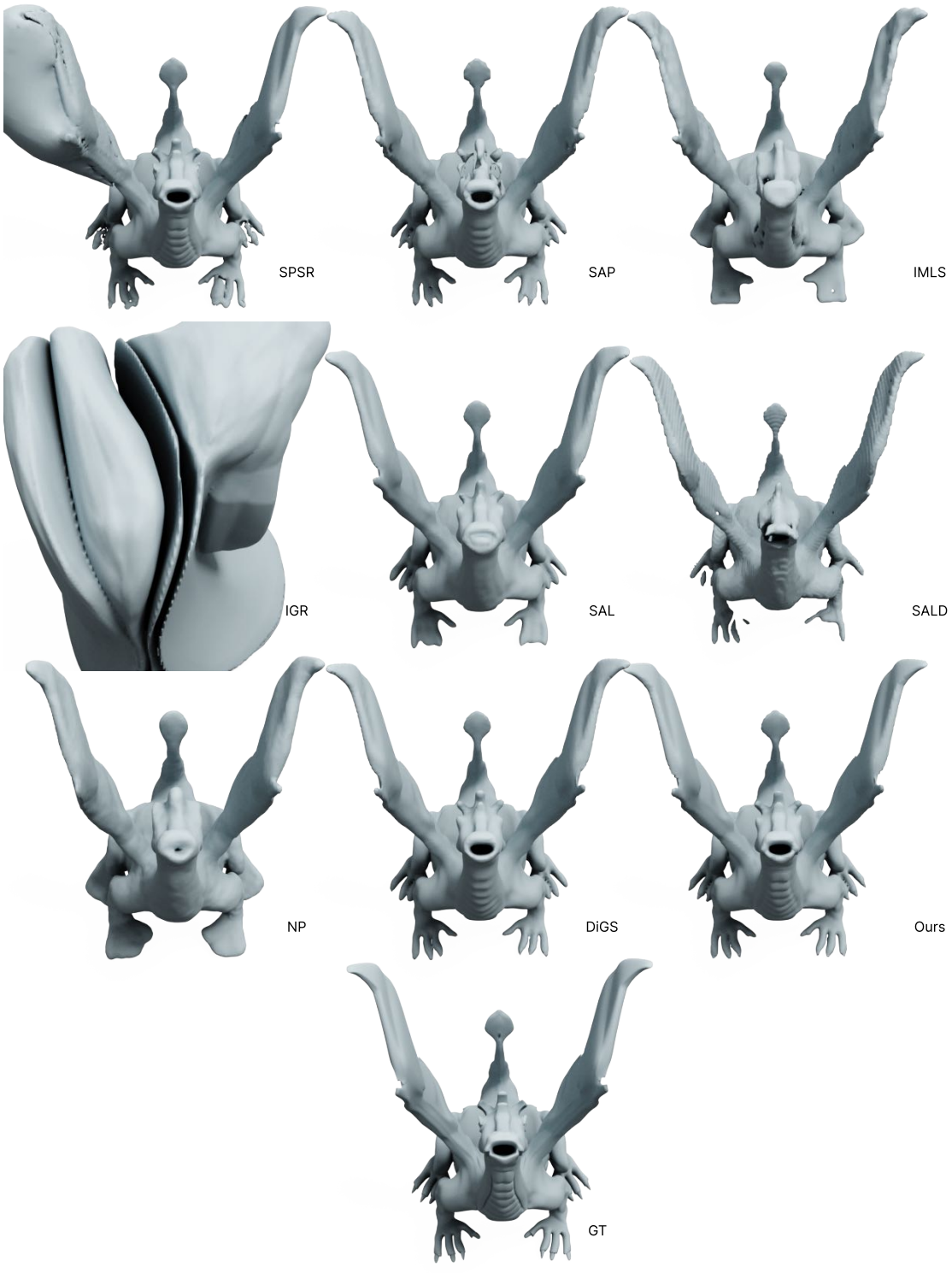


Figure 7. Visual comparisons on Thingi10K [6].

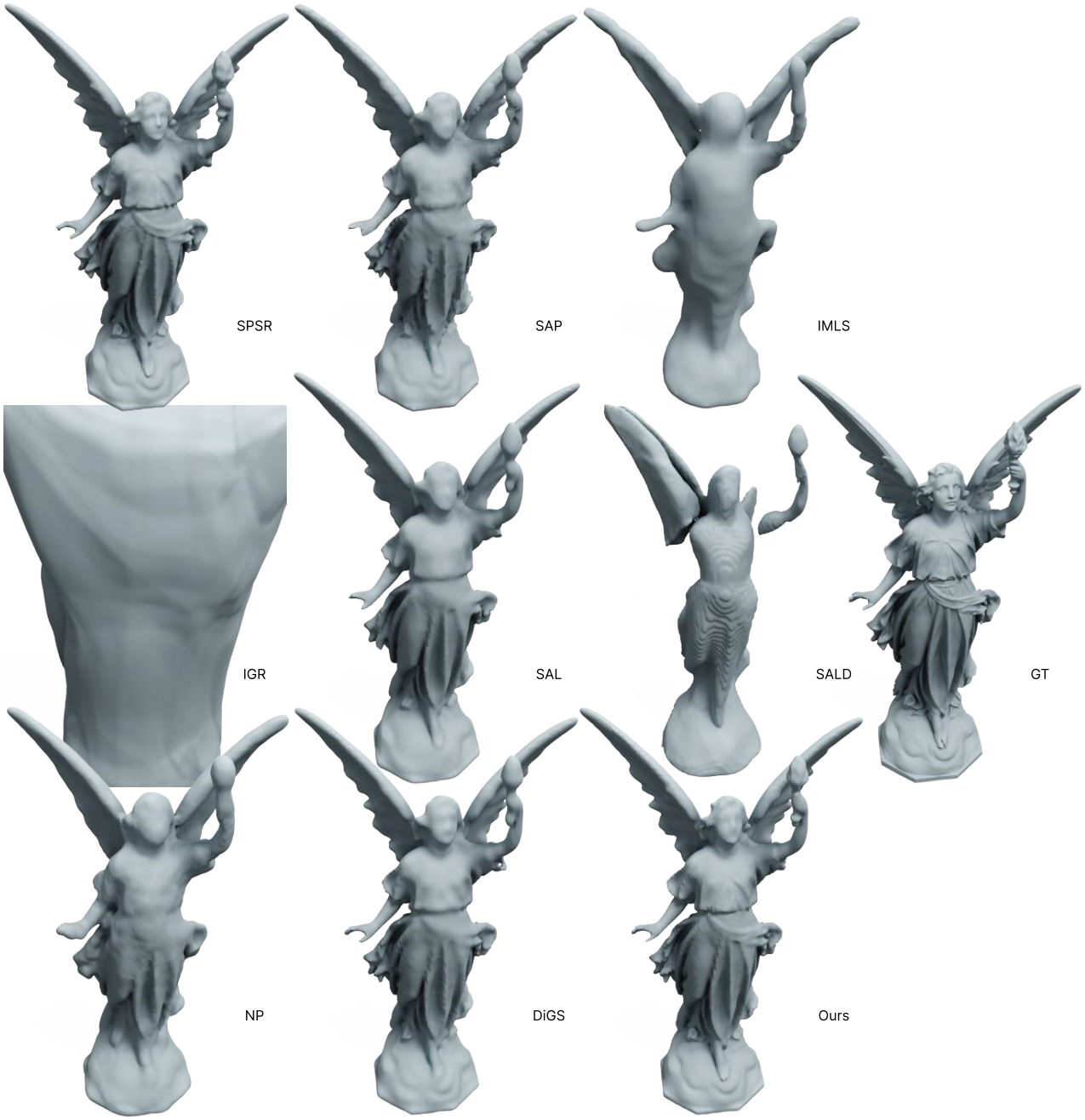


Figure 8. Visual comparisons on Thingi10K [6].

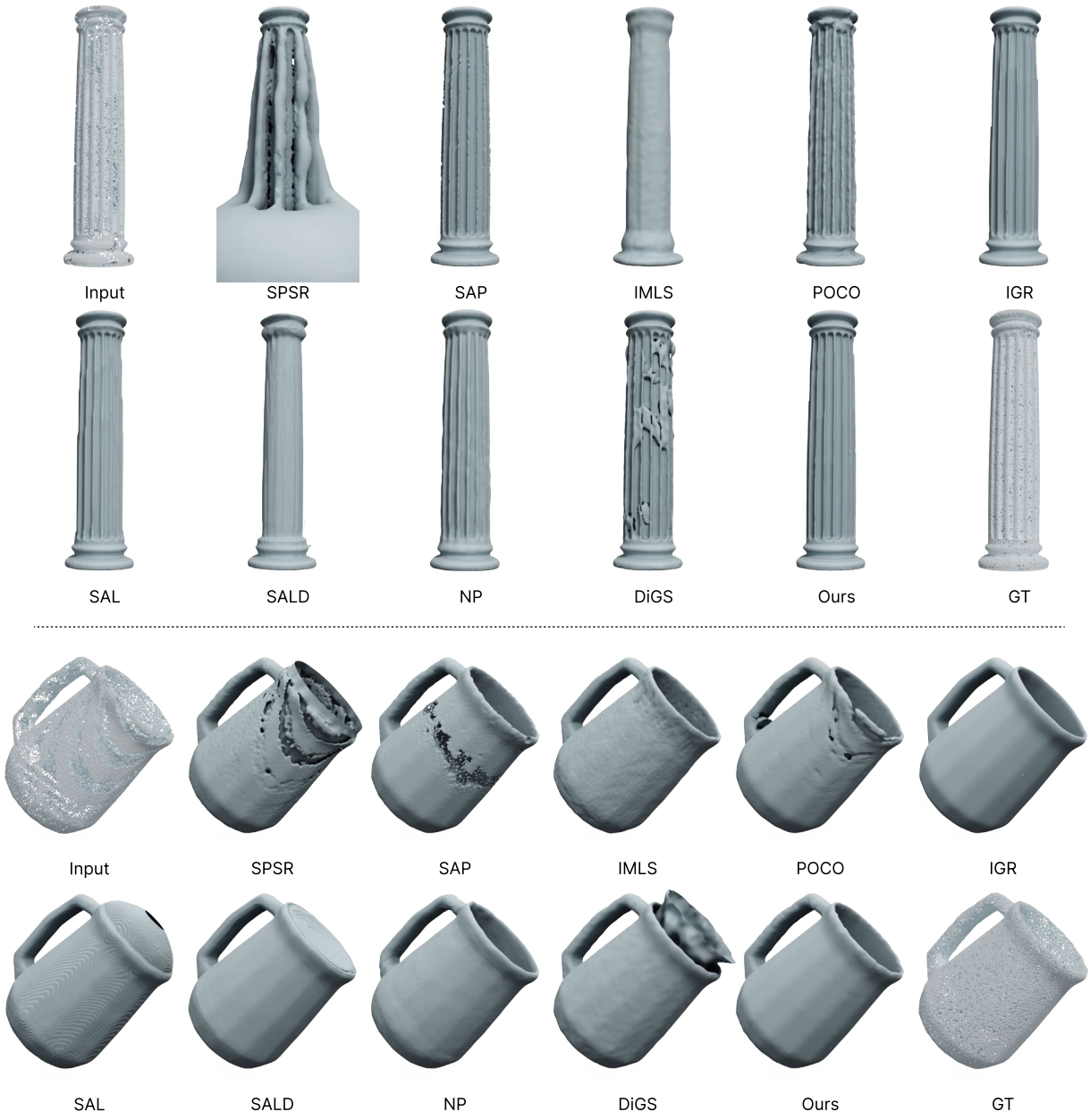


Figure 9. Visual comparisons on the density-variation data [8].

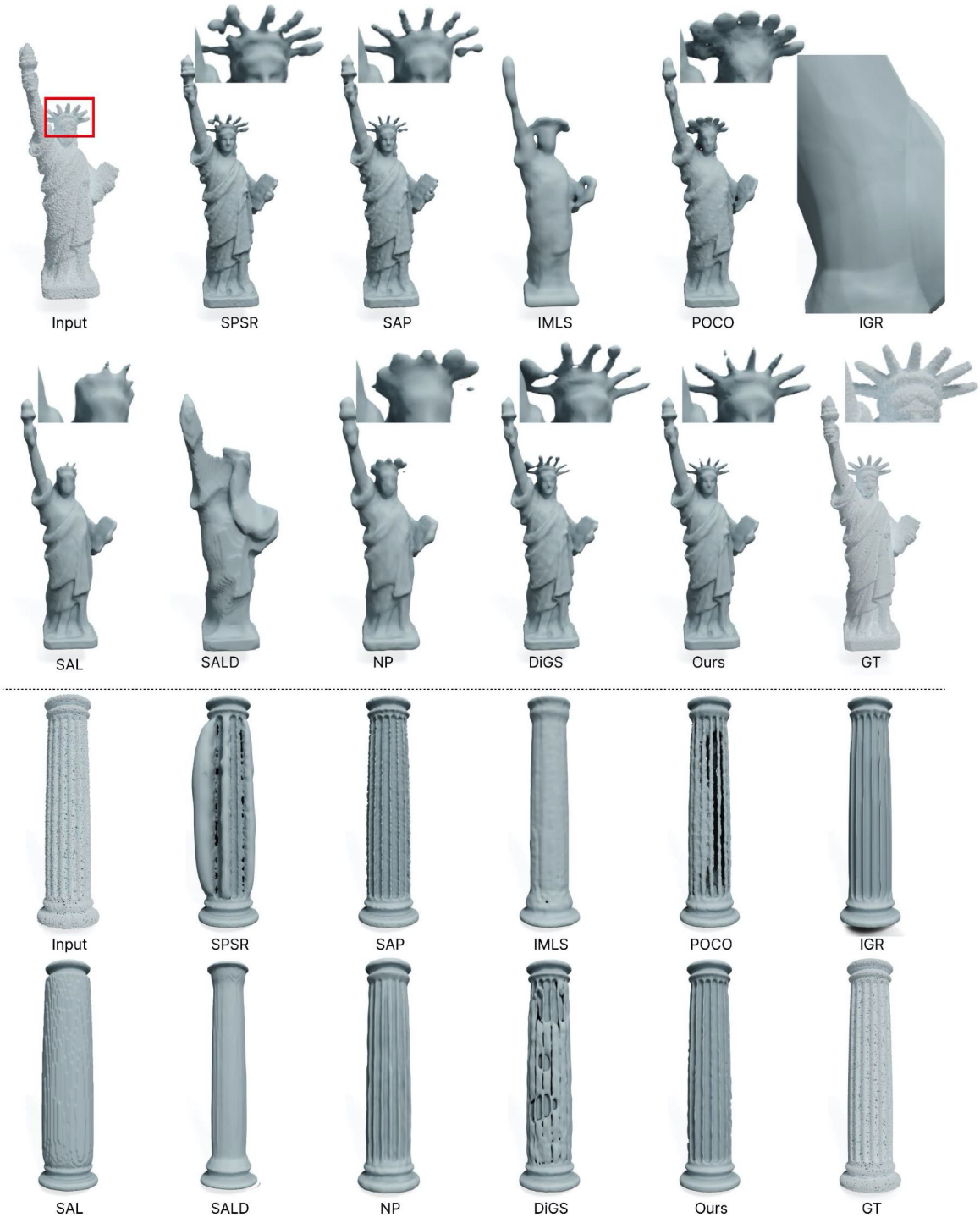


Figure 10. Visual comparisons on the noisy data [8].

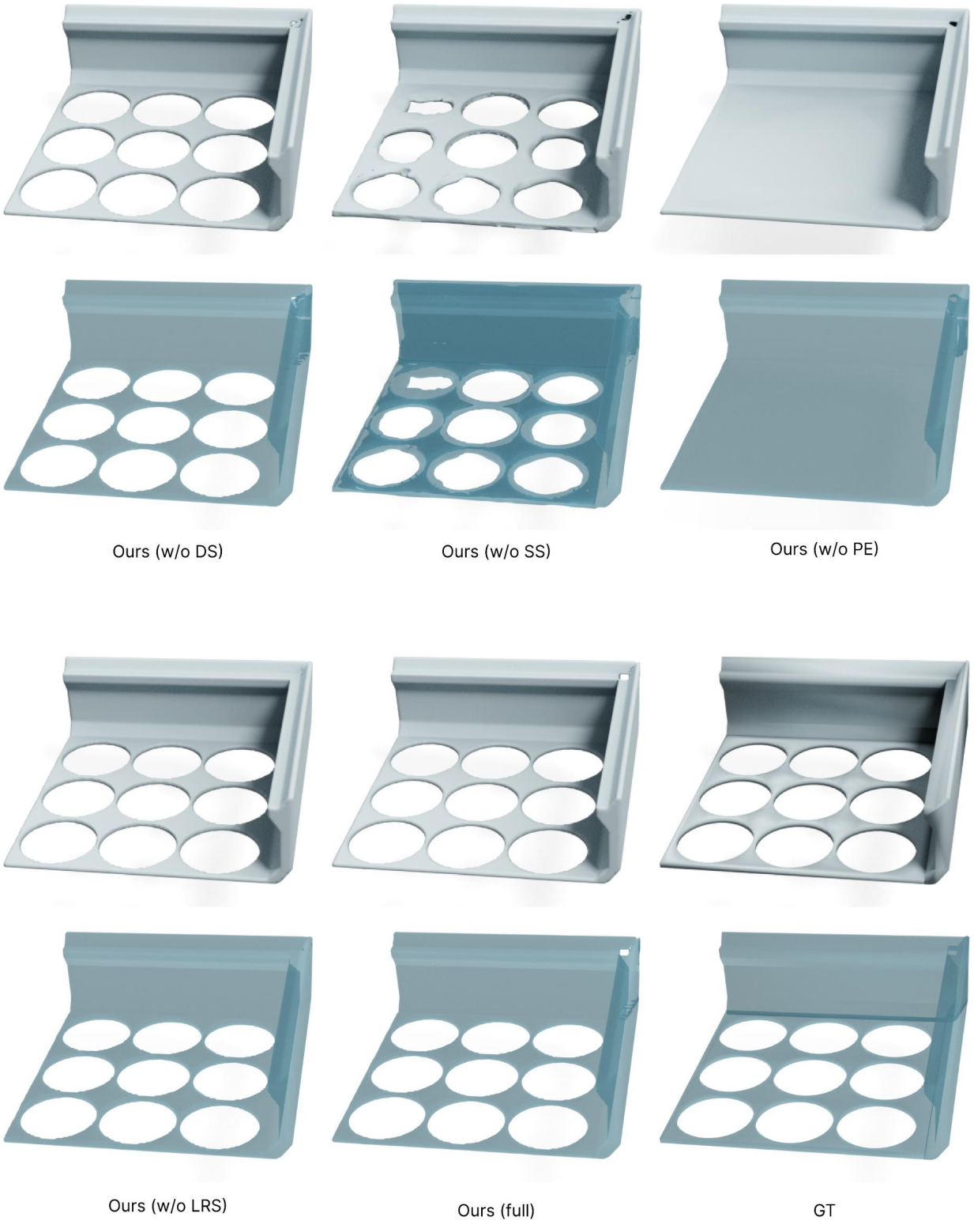


Figure 11. Ablation study on Thingi10K [6]. Note that, we provide two images (*i.e.*, non-transparent image (top) and transparent image (bottom)) for each result.

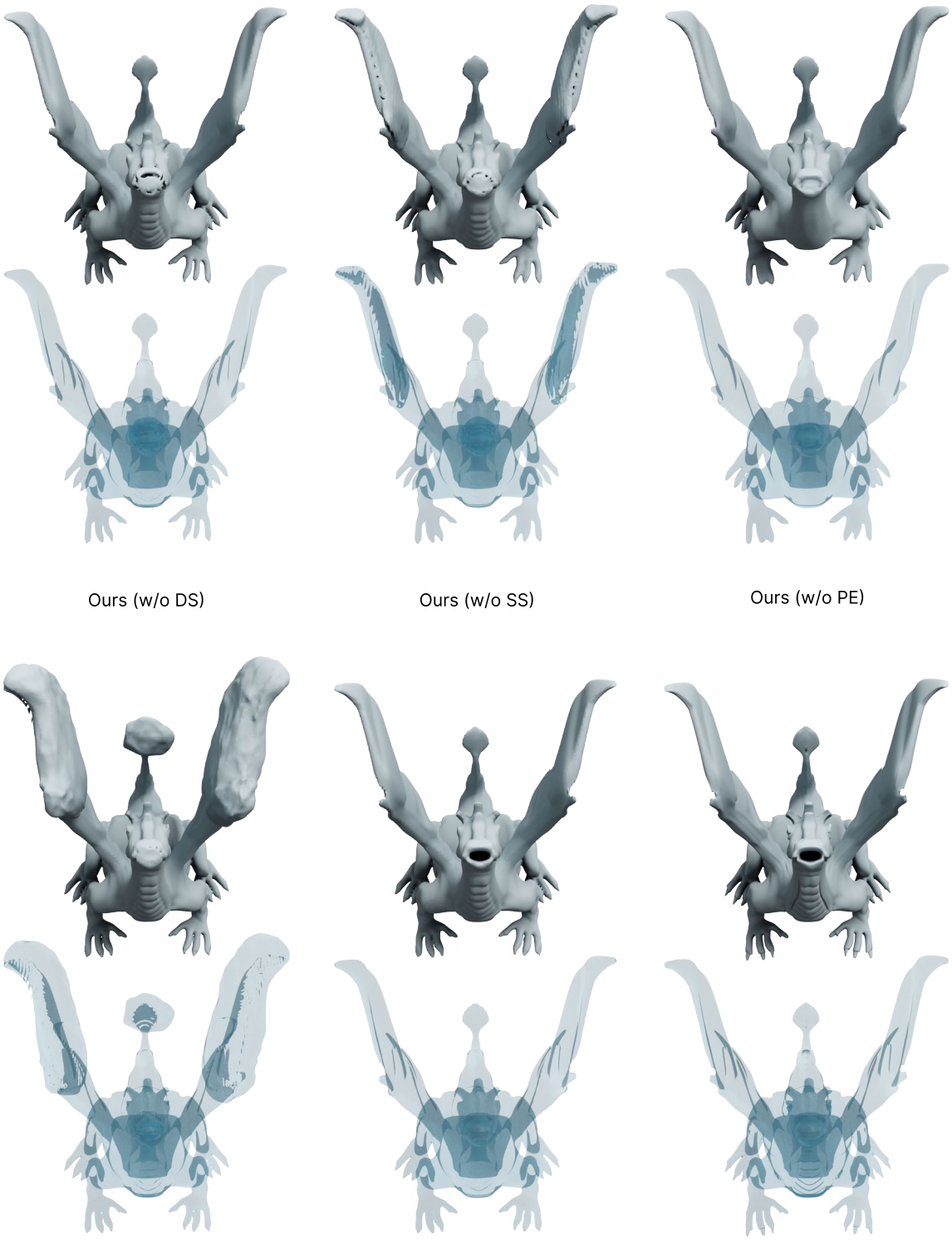


Figure 12. Ablation study on Thingi10K [6]. Note that, we provide two images (*i.e.*, transparent image (top) and non-transparent image (bottom)) for each result.

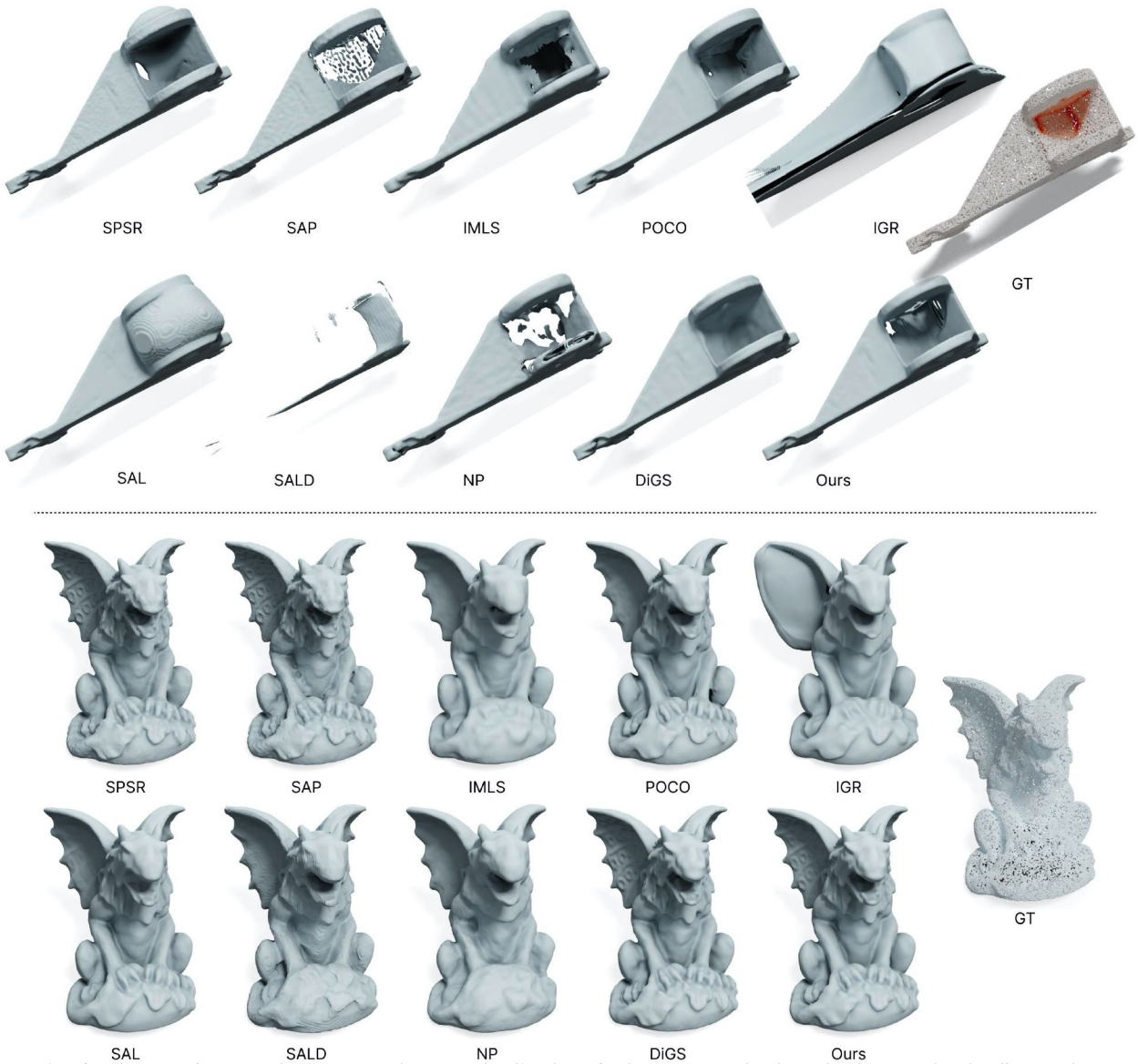


Figure 13. Visual comparisons on the SRB [13] data. For the GT shape in the top example, the red color encodes the distance between the GT point clouds with the input clouds. Deeper red color means the larger distance, thus indicating the missing regions.