Supplementary for "Learning Consistency-Aware Unsigned Distance Functions Progressively from Raw Point Clouds"

Junsheng Zhou[1*] **Baorui Ma**[1*] **Yu-Shen Liu**[1†] **Yi Fang**[2] **Zhizhong Han**[3]
School of Software, BNRist, Tsinghua University, Beijing, China[1]
Center for Artificial Intelligence and Robotics, New York University Abu Dhabi, Abu Dhabi, UAE[2]
Department of Computer Science, Wayne State University, Detroit, USA[3]
`zhoujs21@mails.tsinghua.edu.cn` `mbr18@mails.tsinghua.edu.cn`
`liuyushen@tsinghua.edu.cn` `yfang@nyu.edu` `h312h@wayne.edu`

A Implementation Details

Network design. Our network contains 8 layers of MLP where each layer has 256 nodes. We adopt a skip connection in the fourth layer as employed in DeepSDF [10] and ReLU activation functions in the last two layers of MLP. To make sure the network learns an unsigned distance, we further adopt a non-linear projection $g(x) = |x|$ before the final output.

Training settings. During training, we employ the Adam optimizer with an initial learning rate of 0.001 and a cosine learning rate schedule with 1k warm-up iterations. We start the training of next stage after the previous one converges. In experiment, we found the iteration of 40k, 60k and 70k suitable for the change of stages. Since the 3rd and 4th stages bring little advance as shown in Tab.7 of the main paper, we specify the number of stages as two in practical, so the network has been trained in total 60k iterations. For the surface reconstruction of real scanned complex scenes, we increase the iterations to 300k for a better convergence.

Evaluation settings. For experiments on ShapeNet and 3D Scenes [16], we employ the same evaluation settings as GIFS [15] and OnSurf [9]. We also follow SAP [11] to construct experiments on the SRB dataset [14], however, SAP didn't provide the evaluation settings of point numbers on the SRB dataset. For a fair comparison, we use the evaluation code provided by SAP and test the reconstructed results provided by the author of SAP under different evaluation settings where we found that sampling 50k points on the reconstructed shapes and 150k points on the ground truth shapes can completely reproduce the results of SAP.

B Proofs

Assumption: Given a raw point cloud which is a discrete representation of the surface, we have made a reasonable assumption: the closer the query location is to the given point cloud, the smaller the error of searching the target point on the given point cloud.

In the task of surface reconstruction, one of the key properties is local planarity, that is, the surface can be approximated by tangent plane with infinite accuracy. In graphics, the common form of expressing surfaces is polygon mesh. To complete our proof, we provide a few necessary definitions,

---

[*] Equal contribution.

[†] The corresponding author is Yu-Shen Liu. This work was supported by National Key R&D Program of China (2020YFF0304100), the National Natural Science Foundation of China (62272263, 62072268), and in part by Tsinghua-Kuaishou Institute of Future Media Data.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Definition 1. Polygon mesh is a collection of vertices, edges and faces that defines the surface of 3D shape $S = (V, F)$. Vertices $V = \{v_1, \dots, v_w\}$ determine the position of the faces in 3D space. The edges represent the connection between two vertices, thus a closed set of edges forming the faces $F = \{f_1, \dots, f_e\}$.

Definition 2. Given a point cloud $P = \{p_i, i \in [1, N]\}$, its ground truth surface model $S = (V, F)$ consists of vertices $V = \{v_1, \dots, v_w\}$ and faces $F = \{f_1, \dots, f_e\}$. Assume that given a set of query points $Q = \{q_i, i \in [1, M]\}$, the nearest point of query point q_i on the model $S = (V, F)$ is g_i, and g_i must be on one of the faces F. We define the meaning of error of searching the target point on the given point cloud, $E(S, P, Q) = \sum_{i \in [1,M]} \min_{j \in [1,N]} \|g_i - p_j\|_2$.

Review our strategy for sampling query points: We sample m queries around each point p_i on P. A Gaussian function $\mathcal{N}(\mu, \sigma^2)$ is adopt to calculate the sampling probability for generaing query location $q_\sigma^{i,j}, j \in [1, m]$, where $\mu = p_i$. Suppose we collect two sets of query points $Q_{\sigma_1} = \{q_{\sigma_1}^{i,j} \mid i \in [1, N], j \in [1, m]\}$ and $Q_{\sigma_2} = \{q_{\sigma_2}^{i,j} \mid i \in [1, N], j \in [1, m]\}$, where σ_1 and σ_2 are standard deviations of the sampled Gaussian function.

Theorem 1. Given a point cloud $P = \{p_i, i \in [1, N]\}$ and its ground truth surface model $S = (V, F)$. Assume two sets of query points Q_{σ_1} and Q_{σ_2} are sampled, where $\sigma_1 > \sigma_2$, then $E(S, P, Q_{\sigma_1}) > E(S, P, Q_{\sigma_2})$.

Proof 1. Queries $q_\sigma^{i,j}, j \in [1, m]$ are sampled around point p_i of the given point cloud P. Assume that the nearest surface point on the model $S = (V, F)$ to the query point $q_\sigma^{i,j}$ is $g_\sigma^{i,j}$, where $g_\sigma^{i,j}$ on the face $f_\sigma^{i,j,k}$. The point p_i must be on one of the faces F, assuming that this face is f_u. Let $Q_{\sigma_1}^i = \{q_{\sigma_1}^{i,j} \mid j \in [1, m]\}$,$Q_{\sigma_2}^i = \{q_{\sigma_2}^{i,j} \mid j \in [1, m]\}$, where $\sigma_1 > \sigma_2$. We decompose $Q_{\sigma_1}^i$ and $Q_{\sigma_2}^i$ to the two sides, $Q_{\sigma_1,i}^+ = \{q_{\sigma_1}^{i,j} \mid f_{\sigma_1}^{i,j,k} = f_u\}$,$Q_{\sigma_2,i}^+ = \{q_{\sigma_2}^{i,j} \mid f_{\sigma_2}^{i,j,k} = f_u\}$, and $Q_{\sigma_1,i}^- = \{q_{\sigma_1}^{i,j} \mid f_{\sigma_1}^{i,j,k} \neq f_u\}$, $Q_{\sigma_2,i}^- = \{q_{\sigma_2}^{i,j} \mid f_{\sigma_2}^{i,j,k} \neq f_u\}$. Therefore $E(S, P, Q_{\sigma_1}^i) = E(S, P, Q_{\sigma_1,i}^+) + E(S, P, Q_{\sigma_1,i}^-)$, $E(S, P, Q_{\sigma_2}^i) = E(S, P, Q_{\sigma_2,i}^+) + E(S, P, Q_{\sigma_2,i}^-)$. Since the sampling probability is calculated based on Gaussian function, that is defined by density function $\mu(q)$, assuming that $f_u = n^T x + d$,

$$
\begin{aligned}
E(S, P, Q_{\sigma_1,i}^+) &= \int ||g_{\sigma_1}^{i,j} - p_i\|_2 \mu(q_{\sigma_1}^{i,j}) d(q_{\sigma_1}^{i,j}) \\
&= \sum_{j \in [1,m]} ||q_{\sigma_1}^{i,j} - \frac{q_{\sigma_1}^{i,j} \cdot n^T + d}{n^T \cdot n^T} \cdot n^T - p_i\|_2 \frac{1}{\sqrt{(2\pi)^3 \sigma_1 |I|}} e^{-\frac{1}{2}(q_{\sigma_1}^{i,j} - p_i)^T \sigma_1 I (q_{\sigma_1}^{i,j} - p_i)}
\end{aligned}
\tag{1}
$$

Since $E(S, P, Q_{\sigma_1,i}^+)$ will decrease with decreasing σ_1, then we get $E(S, P, Q_{\sigma_1,i}^+) > E(S, P, Q_{\sigma_2,i}^+)$.

The geometric distribution of $f_\sigma^{i,j,k}$ appearing around f_u can be regarded as randomly distributed, the $E(S, P, Q_{\sigma,i}^-)$ is also randomly distributed and independent of the density function, so that in a statistical sense, $E(S, P, Q_{\sigma 1,i}^-) = E(S, P, Q_{\sigma_2,i}^-)$. Then we get $E(S, P, Q_{\sigma_1,i}) > E(S, P, Q_{\sigma_2,i})$. The case of arbitrary other $Q_{\sigma_1}^j$ and $Q_{\sigma_2}^j$ proven similarly. So we proof $E(S, P, Q_{\sigma_1}) > E(S, P, Q_{\sigma_2})$.

C Additional Experiments

The effect of training iterations. We further test the effect of training iterations of the first and second stage. Since we use an end-to-end training strategy, we set a relatively small number of training iterations for the second stage. In experiments, we set the numbers of iteration in the first stage to 30k, 40k and 50k, and the second stage to 15k, 20k and 25k. For testing the effect of the iteration numbers in the first stage, we set the iteration numbers of second stage to the default 20k, and the iteration numbers of first stage is set to the default 40k while testing the second stage. The results in Tab. 1 show that too few training iterations will lead to an under fitting problem and too many training iterations will result in network degradation.

Efficiency comparison on field learning. We make a comparison with Neural-Pull [8], IGR [5], Point2mesh [6] on the computational cost of optimizing for a single point cloud in Table 2. The

Stage1	30k	40k	50k
Chamfer-L2	0.1158	**0.1112**	0.1137
Stage2	15k	20k	25k
Chamfer-L2	0.1125	**0.1112**	0.1133

Table 1: Effect on training iterations of different stages.

results show that our proposed method converges faster than all the baselines with fewer memory requirements.

Methods	Neural-Pull	IGR	Point2mesh	Ours
Time (s)	1150	1212	4028	**667**
Memory (GB)	2.2	6.1	5.2	**2.0**

Table 2: Efficiency comparison on field learning.

Efficiency comparison on mesh extraction. We further evaluate the efficiency of our method. The comparison is shown in Tab. 3. For reproducing the mesh generation process of NDF [4], we use the default 1×10^6 points and the parameters provided by NDF to generate a mesh with ball-pivoting-algorithm (BPA) [1]. It's clear that our method achieves a tremendous advantage over NDF even with a relatively high resolution (e.g.256^3). The reason is that our method allows a straightforward surface extracting from the learned UDFs while the ball-pivoting-algorithm used by NDF requires a lot of calculations for neighbour searching and normal estimation.

Method	NDF [4]	Ours 64^3	Ours 128^3	Ours 256^3	Ours 320^3
Time	2080.7 s	3.0 s	21.9 s	**162.2 s**	307.6 s

Table 3: Efficiency comparison of surface generation.

The complete comparison under 3D Scene dataset. We also provide the complete comparison under all the five scenes of the 3D Scene dataset [16]. The results is shown in Tab. 4. We provide a comprehensive comparison with NDF by comparing both the generated point cloud ($*_{PC}$) and the generated mesh ($*_{mesh}$). We use L1 and L2 chamfer distance (L2CD, L1CD) as evaluation metrics.

D Additional Visualizations

Visualizations of the unsigned distance field. To further evaluate the effectiveness of our consistency-aware field learning, we visualize the learned unsigned distance fields with Neural-Pull [8] loss in Eq. (2) of our submission and our field consistency loss in Eq. (3) of our submission. Fig. 1 shows the visual comparison under two different shapes with complex inner structures. For the storey bus, training with the loss proposed by Neural-Pull fails to handle the rich structures, thus leads to a chaotic distance field where no details were kept. On the contrary, our proposed consistency-aware learning can build a consistent distance field where the detailed structures are well preserved. It's clear that our method learns a highly continuous unsigned distance field and has the ability to keep the field around complex structures correct (e.g. the seats, tires and stairs), which also helps to extract a high fidelity surface.

Visual comparison with ball-pivoting-algorithm. To further explore the advantage of our straightforward surface extraction algorithm, we use the same setting as NDF [4] to adopt ball-pivoting-algorithm (BPA) [1] to extract mesh from our generated point cloud and make a comparison with the mesh extracted using our method as shown in Fig. 2. Even with a carefully selected threshold, the mesh generated by BPA is still far from smooth and has a number of holes, and also fails to retain the detailed geometric information. On the contrary, our method allows to extract surfaces directly from the learned UDFs, thus is able to reconstruct a continuous and high-fidelity mesh where the geometry details is well preserved.

Visualizations of our generated intermediate and final point clouds. As shown in Fig. 3, we provide the visualizations of the raw input point cloud as 'Input' and the intermediate point cloud

		Burghers		Lounge		Copyroom		Stonewall		Totempole	
		L2CD	L1CD	L2CD	L1CD	L2CD	L1CD	L2CD	L1CD	L2CD	L1CD
$100/m^2$	COcc [12]	8.904	0.040	6.979	0.041	6.78	0.041	12.22	0.051	4.412	0.041
	LIG [7]	3.112	0.044	9.128	0.054	4.363	0.039	5.143	0.046	9.58	0.062
	DeepLS [3]	3.111	0.050	3.894	0.056	1.498	0.033	2.427	0.038	4.214	0.043
	NDF$_{PC}$ [4]	0.320	0.012	0.417	0.013	0.291	0.012	0.252	0.010	0.767	0.015
	NDF$_{mesh}$ [4]	0.463	0.014	0.484	0.015	0.439	0.015	0.248	0.011	0.624	0.014
	OnSurf [9]	0.544	0.018	0.435	0.013	0.434	0.017	0.371	0.016	3.986	0.040
	Ours$_{PC}$	**0.121**	**0.010**	**0.277**	**0.013**	**0.150**	**0.010**	**0.079**	**0.008**	**0.090**	**0.008**
	Ours$_{mesh}$	**0.212**	**0.011**	**0.245**	**0.011**	**0.214**	**0.012**	**0.118**	**0.009**	**0.145**	**0.009**
$500/m^2$	COcc [12]	26.97	0.081	9.044	0.046	10.08	0.046	17.70	0.063	2.165	0.024
	LIG [7]	3.080	0.046	6.729	0.052	4.058	0.038	4.919	0.043	9.38	0.062
	DeepLS [3]	0.714	0.020	10.88	0.077	0.552	0.015	0.673	0.018	21.15	0.122
	NDF$_{PC}$ [4]	0.304	0.014	0.261	0.011	0.162	0.010	0.239	0.012	0.919	0.021
	NDF$_{mesh}$ [4]	0.546	0.018	0.314	0.012	0.242	0.012	0.226	0.012	1.049	0.025
	OnSurf [9]	0.609	0.018	0.529	0.013	0.483	0.014	0.666	0.013	2.025	0.041
	Ours$_{PC}$	**0.072**	**0.008**	**0.146**	**0.011**	**0.072**	**0.008**	**0.038**	**0.007**	**0.064**	**0.008**
	Ours$_{mesh}$	**0.192**	**0.011**	**0.099**	**0.009**	**0.120**	**0.009**	**0.069**	**0.008**	**0.131**	**0.010**
$1000/m^2$	COcc [12]	27.46	0.079	9.54	0.046	10.97	0.045	20.46	0.069	2.054	0.021
	LIG [7]	3.055	0.045	9.672	0.056	3.61	0.036	5.032	0.042	9.58	0.062
	DeepLS [3]	0.401	0.017	6.103	0.053	0.609	0.021	0.320	0.015	0.601	0.017
	NDF$_{PC}$ [4]	0.575	0.019	0.303	0.012	0.186	0.011	0.407	0.016	1.333	0.026
	NDF$_{mesh}$ [4]	1.168	0.027	0.393	0.014	0.269	0.013	0.509	0.019	2.020	0.036
	OnSurf [9]	1.339	0.031	0.432	0.014	0.405	0.014	0.266	0.014	1.089	0.029
	Ours$_{PC}$	**0.087**	**0.010**	**0.057**	**0.009**	**0.083**	**0.010**	**0.043**	**0.007**	**0.086**	**0.010**
	Ours$_{mesh}$	**0.191**	**0.010**	**0.092**	**0.008**	**0.113**	**0.009**	**0.066**	**0.007**	**0.139**	**0.009**

Table 4: Surface reconstruction for point clouds under 3D Scene. L2CD×1000.

updated by the well-moved queries which serves as the target point cloud at the second stage, as shown by 'Stage2'. We also provide the generated dense point cloud by moving the queries to the approximated surface using Eq. (1) in our submission as shown by 'Ours'. The raw input point cloud is highly discrete and fails to provide detailed supervision for surface reconstruction. On the contrary, the intermediate point cloud updated by the well-moved queries shows great uniformity and continuity, thus can provide the supervision of fine geometries such as the detailed steering wheel and tires of the car. The quantitative comparison of using the raw input point cloud and the intermediate point cloud as target for surface reconstruction is shown in Tab. 6 of our submission, where leveraging the intermediate point cloud as target can achieve 10% improvement over leveraging the raw input point cloud. Furthermore, due to our progressive surface approximation strategy, our generated dense point cloud can maintain uniformity and keep the detailed geometric.

Visualizations of meshes extracted with different settings. We provide the visualizations of the extracted surfaces with different resolutions as shown in Fig. 4. It shows that a higher resolution leads to a more detailed reconstruction. Moreover, our designed mesh refinement operation brings great improvement in surface smoothness and local details due to the accurate unsigned distance values predicted by the neural network.

More visualization comparisons. We provide more visual comparisons under Surface Reconstruction Benchmark (SRB) dataset [14] in Fig. 5 and MGD dataset [2] in Fig. 6, respectively.

E Future works.

We consider two potential future works of our method. First, although the experiments under real scanned shapes and scenes proved the ability of our proposed method in handling noises with unknown distributions, we think it's an interesting future work to extend our method for reconstructing surfaces from high-noisy point clouds in an unsupervised way. Second, by transferring the sliding window strategy [3, 13] to our method, it's exciting to see the ability of extending our method for representing large scale data.

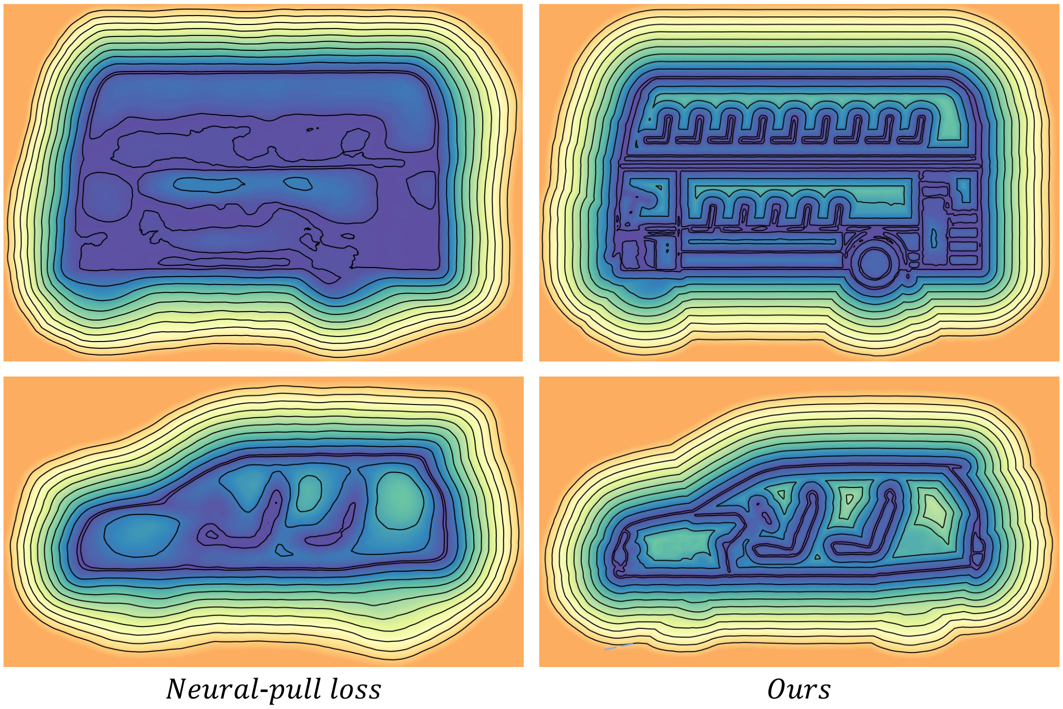

Figure 1: Visualizations of the unsigned distance field. The darker the color, the closer it is to the approximated surface. For a clear contrast, we set the color of the space far away from the approximated surface to orange.

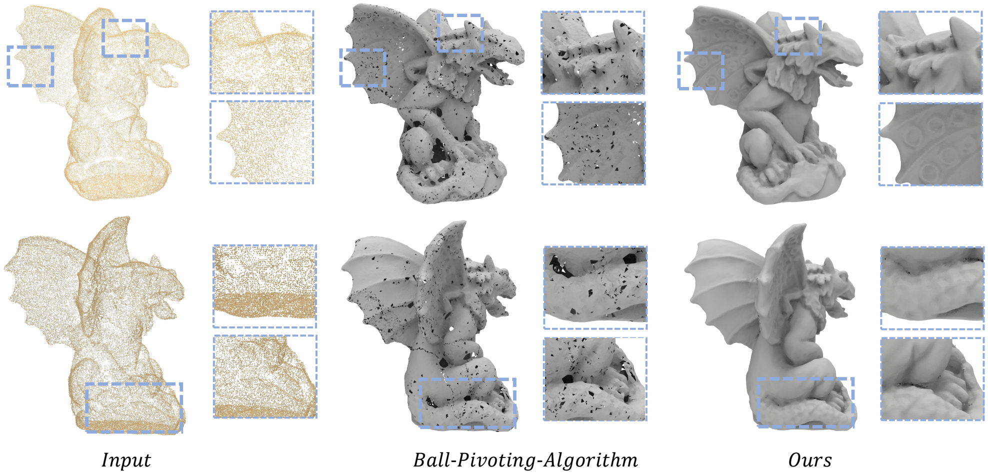

Figure 2: Visual comparison with ball-pivoting-algorithm.

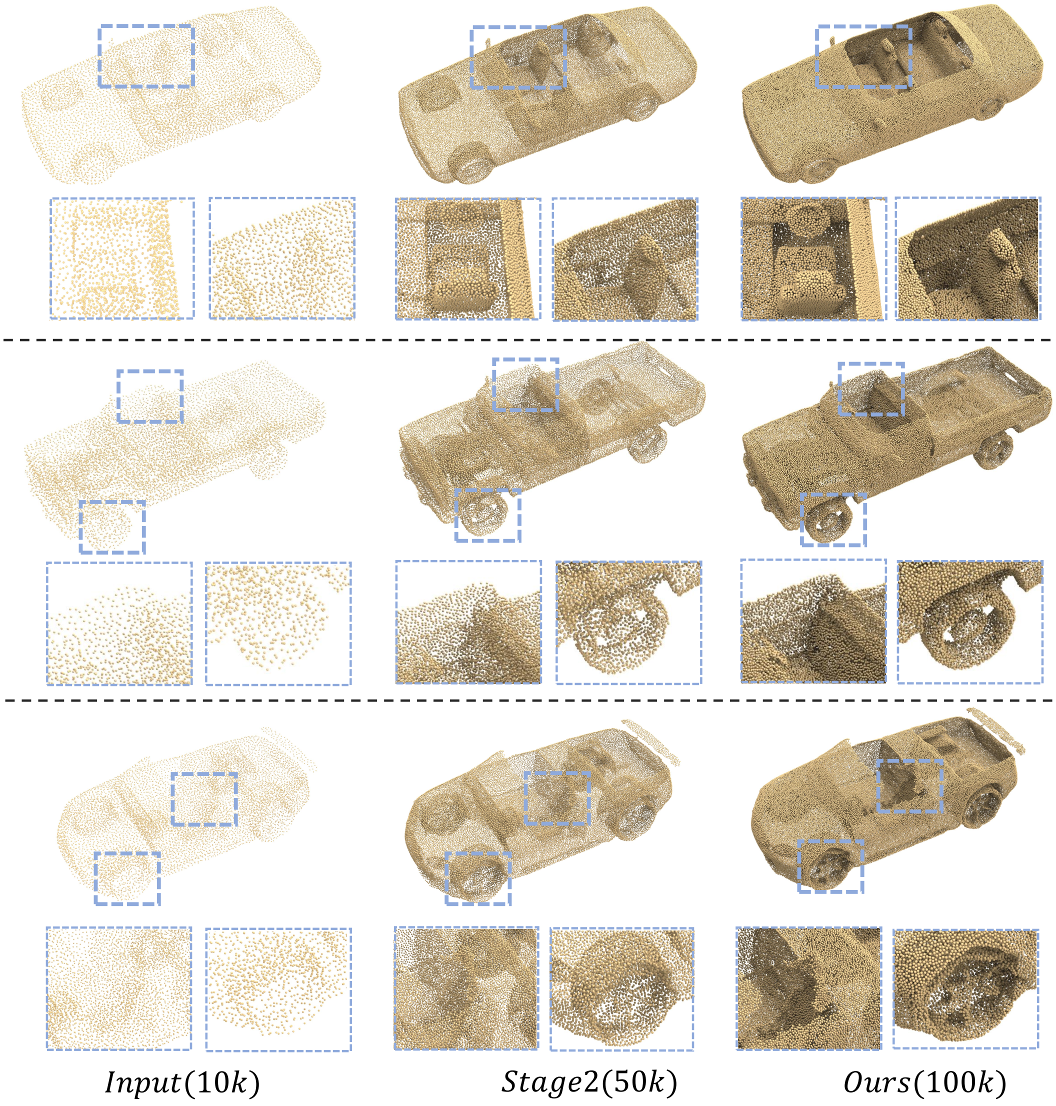

Figure 3: Visualizations of our generated intermediate and final point clouds.

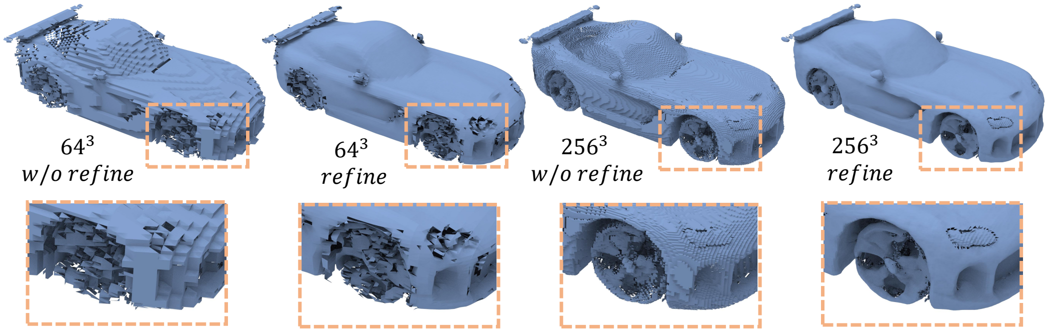

Figure 4: Visualizations of extracted mesh with different settings.

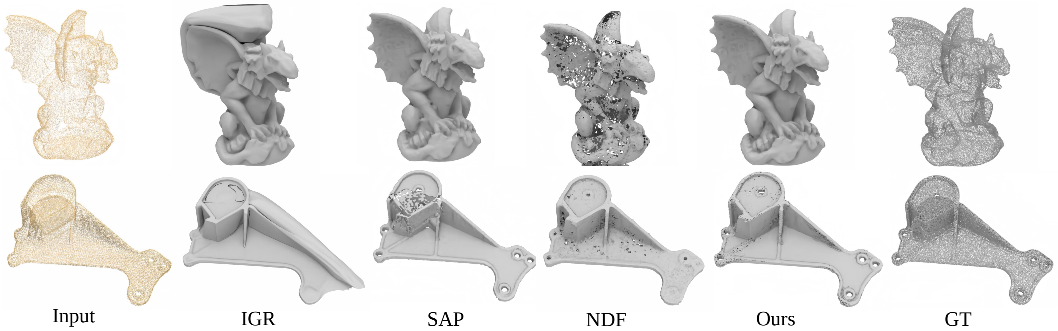

Figure 5: More visualizations under SRB dataset.

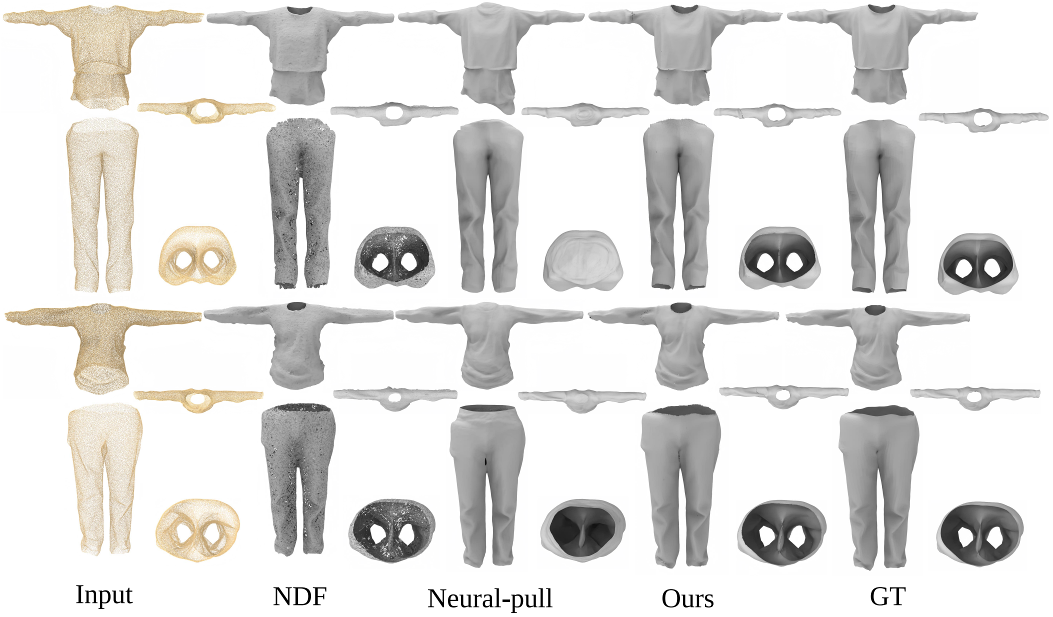

Figure 6: More visualizations under MGD dataset.

References

[1] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[2] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3D people from images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5420–5430, 2019.

[3] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3D reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020.

[4] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020.

[5] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020.

[6] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: a self-prior for deformable meshes. *ACM Transactions on Graphics (TOG)*, 39(4):126–1, 2020.

[7] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3D scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020.

[8] Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-Pull: Learning signed distance function from point clouds by learning to pull space onto surface. In *International Conference on Machine Learning*, pages 7246–7257. PMLR, 2021.

[9] Baorui Ma, Yu-Shen Liu, and Zhizhong Han. Reconstructing surfaces for sparse point clouds with on-surface priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[10] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[11] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. *Advances in Neural Information Processing Systems*, 34, 2021.

[12] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020.

[13] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022.

[14] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10130–10139, 2019.

[15] Jianglong Ye, Yuntao Chen, Naiyan Wang, and Xiaolong Wang. GIFS: Neural implicit function for general shape representation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

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