Supplementary Material of "LAKe-Net: Topology-Aware Point Cloud Completion by Localizing Aligned Keypoints"

Junshu Tang[1], Zhijun Gong[1], Ran Yi[1]*, Yuan Xie[2], Lizhuang Ma[1,2]*
[1]Shanghai Jiao Tong University, [2]East China Normal University
{tangjs, gongzhijun, ranyi}@sjtu.edu.cn, yxie@cs.ecnu.edu.cn, ma-lz@cs.sjtu.edu.cn

Abstract

This supplementary material consists of four parts. In Section 1 we introduce the proof of the alignment of keypoints detected by our proposed unsupervised multi-scale keypoint detector. Then we introduce the implementation details in Section 2. Later, we show the detailed quantitative comparison results on ShapeNet55 dataset using three widely-used metrics and more qualitative results on PCN dataset in Section 3. At last, in Section 4, we discuss the limitation of our method on extreme cases. For more details, we provide source code for closer inspection.

1. Proof of Aligned Keypoints Detected by Unsupervised Multi-scale Keypoint Detector

As mentioned in the main text, the keypoints extracted by our proposed unsupervised multi-scale keypoint detector (UMKD) D (shown in Figure 1) follow a theory that:

Theorem 1. *The coordinates of detected keypoints* $\boldsymbol{P}$ *are irrelevant to the order of original points* $\boldsymbol{X}$*. That is:* $\boldsymbol{P} = D(\boldsymbol{X}) = D(\mathcal{R}(\boldsymbol{X}))$*, where* $\mathcal{R}(\cdot)$ *denotes random permutation operation.*

Proof. Let $\mathbf{X} \in \mathbb{R}^{N\times 3}$ denote the sampled input point clouds. N is the number of points. The PointNet++ [4] based encoder captures point-wise features $H \in \mathbb{R}^{N\times D}$ where D denotes the feature dimension. Denote $\mathbf{X} = \{\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_N^T\}$ and $H = \{h_1^T, h_2^T, \cdots, h_N^T\}$, where $\mathbf{x}_i \in \mathbb{R}^{3\times 1}$ and $h_i \in \mathbb{R}^{D\times 1}$ are the coordinates and feature of the i-th point. Since the weights of the encoder are shared among all points, denote the weights of the shared encoder as $\Omega \in \mathbb{R}^{3\times D}$, the point feature $h_i = \Omega^T \cdot \mathbf{x}_i$. We apply a Linear-based block at the last of the keypoints detector, and denote its weights as: $\Phi \in \mathbb{R}^{D\times K}$, where K is the number of keypoints. The predicted convex combination weights are calculated as $\mathbf{W} = H \cdot \Phi$. We then predict keypoints $\mathbf{P}$ as the convex combination of the input points:

$$
\begin{aligned}
\mathbf{P} &= \mathbf{W}^T\mathbf{X} = (H\cdot\Phi)^T\mathbf{X} = \Phi^T \cdot H^T \cdot \mathbf{X} \\
&= \Phi^T(h_1\mathbf{x}_1^T + h_2\mathbf{x}_2^T + \cdots + h_N\mathbf{x}_N^T) \\
&= \Phi^T(\Omega^T\mathbf{x}_1\mathbf{x}_1^T + \Omega^T\mathbf{x}_2\mathbf{x}_2^T + \cdots + \Omega^T\mathbf{x}_N\mathbf{x}_N^T) \\
&= \Phi^T\Omega^T\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T.
\end{aligned} \tag{1}
$$

According to Equation 1, the coordinates of predict keypoints $\mathbf{P}$ are irrelevant to the order of original points $\mathbf{X}$. That is to say, objects with similar shapes or topology have almost the same ordered keypoints. Therefore, our proposed UMKD can capture aligned keypoints within a subcategory.

2. Implementation Details

Our proposed method LAKe-Net is implemented with PyTorch on AMD Ryzen 3700X with 2 NVIDIA GTX 3090 GPUs. We utilize Adam optimization to train the whole architecture and the betas are set as 0.9 and 0.999. The learning rate is initialized as 0.001. For point cloud reconstruction, we set the batch size to 64 and train the model for 60 epochs with the continuous learning rate decay of 0.7 for every 20 epochs. And for point cloud completion, we resume the learning rate and train the model for 100 epochs with the continuous learning rate decay of 0.7 for every 20 epochs.

The architecture of our proposed LAKe-Net consists of four parts: auto-encoder, asymmetric keypoint locator, surface-skeleton generation and shape refinement subnet. We input 2,048 points and output 16,384 and 8,192 points for PCN dataset and ShapeNet dataset, respectively.

The auto-encoder consists of a pointnet-based [3] PCN [11] encoder and a coarse generator. The dimension of the global feature embedding is 1024. We implement the $ConvTranspose$ layer (shown in Figure 2 in main paper) using a special one-dimensional deconvolutional strategy proposed in [6], i.e. the point-wise splitting operation,

*Corresponding authors.

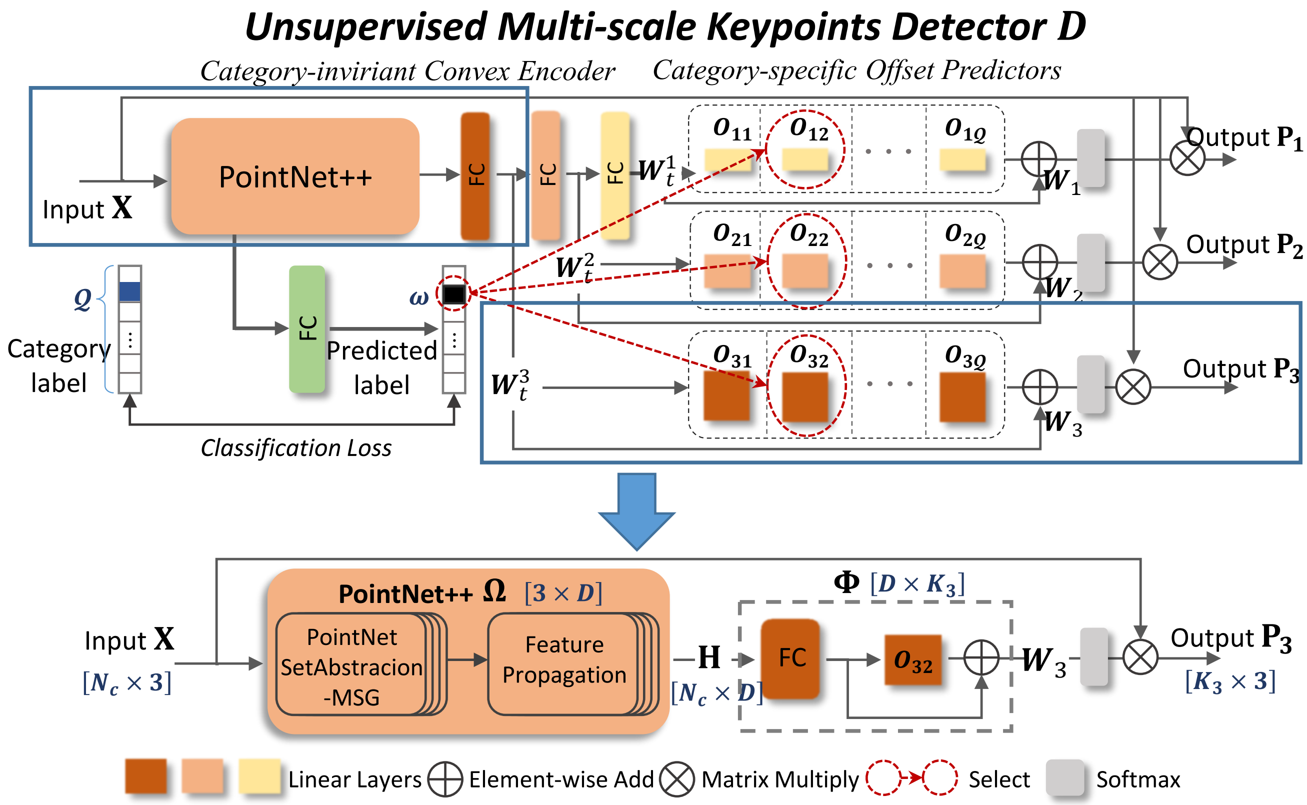

Figure 1. The architecture of our proposed unsupervised multi-scale keypoints detector UMKD. We show the detailed calculation of $\mathbf{P}_3$ which is boxed in blue. Ω and Φ denote the weights of PointNet++ and linear block (boxed in grey and dotted box), respectively.

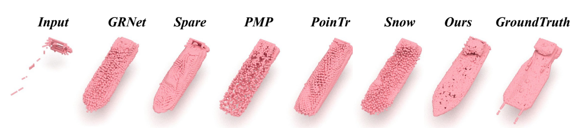

Figure 2. Illustration of failure case and completed results by different point cloud completion methods.

Figure 3. More results of original geometry and corresponding surface-skeletons pairs.

to generate multiple child point features for each global feature. The number of coarse output $\mathbf{X}_c$ and $\mathcal{X}_c$ is 1024.

As for the detailed architecture of the refinement subnet, we propose three recursive skeleton-assisted refinement (RSR) modules which is shown in Figure 2 of the main text. The dimensions of f_i in multi-step are set as [1024, 512, 256, 128]. For each step, we integrate multi-scale surface-skeletons where the numbers of skeleton points are 1024, 2048, 4096. And the up-factors of RSR modules u_i are [1,1,2] for PCN dataset and [1,1,1] for ShapeNet55 dataset.

3. Detailed Experimental Results

We report the detailed results for FoldingNet [9], PCN [11], GRNet [8], PoinTr [10] and our proposed LAKe-Net on ShapeNet55 dataset in Table 2. Each row in the table stands for a category of objects. We list numbers of samples in each category using three widely-used metrics: CD-l_1, CD-l_2 and EMD. We also test the inference time per sample of various methods. The results are shown in Table 1, which shows that our method is comparable with most of other methods.

Besides, we present more qualitative comparison results of point cloud completion on PCN dataset. Figure 4 shows the extra visualization results of our method and other methods on PCN dataset. Experimental results show that our method has the best performance on completing the missing topology and geometric information.

We also provide more results of original geometry and surface-skeletons pairs, generated by 128 keypoints of objects with different topologies (chairs with and without arm) in Figure 3. It shows that our proposed surface skeleton can reconstruct different topological structures. We can flexibly control the fineness of skeletons by adjusting the number of keypoints.

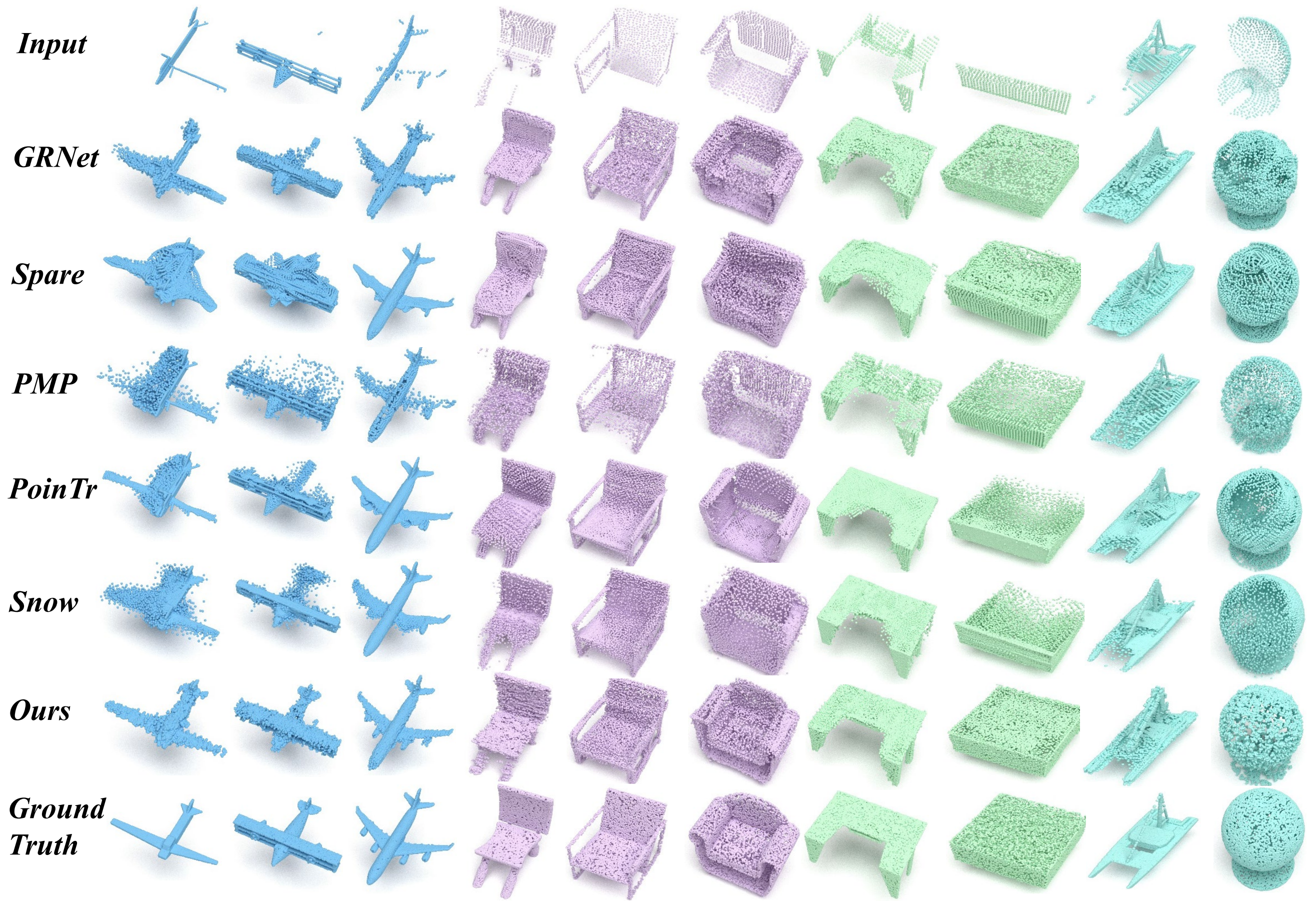

Figure 4. Illustration of point cloud completion comparison with previous methods on different categories of PCN dataset. From top to bottom are the partial input, completed results from GRNet [8], SpareNet [7], PMP-Net [5], PoinTr [10], Snowflake-Net [6], our method and the last row is the ground truth.

Methods	Times(ms)
Folding [9]	1.79
PCN [11]	6.54
AtlasNet [1]	11.1
MSN [2]	63.7
GRNet [8]	23.4
PMP-Net [5]	18.4
SpareNet [7]	1030
PoinTr [10]	18.0
Snowflake [6]	30.4
Ours	34.5

Table 1. Comparison results with other methods on inference time per sample.

4. Limitations

We discuss the limitation of our proposed LAKe-Net on some extreme cases. We show a sampled failure case of point cloud completion in Figure 2. When the partial input misses most of geometric and topological information, all methods including ours may fail to recover the original local geometric details. According to the qualitative comparison results in Figure 2, our method can generate more plausible and clear structure compared with other methods.

References

[1] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 3

[2] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu. Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11596–11603, 2020. 3

[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1

[4] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 1

[5] X. Wen, P. Xiang, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu. Pmp-net: Point cloud completion by learn-

	Samples	CD-l_1($\times 10^3$) $\downarrow$					CD-l_2($\times 10^3$) $\downarrow$					EMD($\times 10^3$) $\downarrow$				
		Folding	PCN	GRNet	PoinTr	Ours	Folding	PCN	GRNet	PoinTr	Ours	Folding	PCN	GRNet	PoinTr	Ours
airplane	6472	21.294	14.301	15.795	10.893	**9.996**	1.391	0.643	2.626	0.658	**0.463**	55.318	24.909	28.661	**21.269**	23.990
bag	136	25.502	21.836	16.501	16.667	**14.643**	2.141	1.535	**0.862**	1.400	1.359	62.715	34.343	**27.208**	29.452	37.897
basket	184	27.857	23.641	20.518	18.798	**18.674**	2.208	1.557	1.188	1.637	**0.954**	63.666	35.671	33.641	**31.971**	34.864
bathtub	1376	27.212	21.900	19.032	17.297	**16.114**	2.208	1.425	1.133	1.565	**1.129**	67.201	33.514	31.577	**30.767**	34.219
bed	376	32.311	27.647	17.526	19.951	**15.894**	3.171	2.495	0.934	2.180	**0.717**	73.551	49.438	29.706	37.562	**28.360**
bench	2904	20.803	17.205	15.186	12.221	**12.120**	1.453	0.956	0.863	0.935	**0.707**	50.075	28.857	25.850	21.445	**18.160**
birdhouse	120	32.276	29.176	19.498	23.980	**18.250**	3.095	2.443	**1.069**	2.917	1.239	81.111	46.768	**31.373**	43.747	34.811
bookshelf	728	28.352	26.917	17.746	19.419	**16.212**	2.483	2.388	0.928	1.862	**0.891**	64.409	44.744	**29.595**	37.062	29.714
bottle	800	21.701	16.893	15.686	15.852	**13.393**	1.431	0.859	**0.826**	1.371	1.010	53.550	**24.915**	25.606	28.306	33.461
bowl	304	30.318	23.288	22.952	**18.265**	19.305	2.637	1.427	1.557	1.331	**0.803**	64.767	33.013	36.404	**30.976**	31.233
bus	1504	24.890	17.820	16.037	**13.009**	14.015	1.716	0.894	0.838	**0.745**	1.060	62.856	31.193	26.847	**25.096**	35.784
cabinet	2520	23.800	21.179	18.915	**16.039**	17.328	1.594	1.236	**1.011**	1.102	1.015	59.865	34.189	31.184	**28.239**	33.237
camera	184	36.762	30.297	18.459	27.252	**17.059**	4.233	2.792	**1.012**	4.118	1.134	83.065	49.564	**30.534**	51.290	36.959
can	176	24.991	21.527	20.286	19.443	**18.715**	1.761	1.303	1.151	2.134	**0.633**	60.234	30.878	32.272	31.228	**29.032**
cap	96	33.213	19.539	18.662	17.445	**15.275**	3.279	1.009	1.277	1.290	**0.971**	74.738	30.302	**29.678**	31.702	32.842
car	5624	26.678	21.451	19.090	17.358	**17.387**	1.929	1.222	**1.054**	1.189	1.069	65.968	36.668	**31.020**	32.398	34.389
cellphone	1336	16.567	14.398	15.884	**11.293**	12.711	0.728	**0.549**	0.891	0.556	0.594	42.215	24.520	26.273	**21.418**	35.284
chair	10848	24.096	20.094	15.862	**0.416**	13.977	1.816	1.209	**0.800**	1.324	1.014	60.203	33.513	**26.685**	31.580	34.133
clock	1048	25.727	21.942	17.877	17.081	**15.863**	1.996	1.484	**0.976**	1.537	1.005	57.494	35.244	**29.268**	31.153	30.092
keyboard	104	17.760	14.226	14.835	**9.821**	11.847	0.878	0.584	0.755	**0.444**	0.934	44.448	22.136	24.017	**16.820**	31.681
dishwasher	152	23.315	20.718	19.294	**17.466**	18.092	1.483	1.168	1.024	1.422	**0.210**	50.122	30.906	31.479	29.311	**16.392**
display	1752	22.821	19.745	16.270	15.079	**14.229**	1.559	1.135	**0.798**	1.218	1.250	51.455	32.232	26.770	**26.356**	35.903
earphone	120	34.411	26.904	16.521	25.901	**14.340**	4.012	2.326	**1.018**	4.547	1.191	76.522	48.377	**27.574**	49.357	32.760
faucet	1192	29.935	23.018	14.203	21.963	**10.721**	3.186	1.955	1.569	1.754	**0.338**	66.154	40.282	24.903	42.439	**20.544**
file cabinet	480	25.942	22.775	18.778	18.112	**17.541**	1.942	1.492	1.005	3.227	**0.965**	65.258	37.221	30.949	42.723	**27.470**
guitar	1280	12.090	8.996	12.144	7.947	**6.777**	0.437	**0.239**	1.426	0.385	0.699	26.276	**18.313**	22.347	18.514	29.145
helmet	264	34.702	29.376	21.077	27.375	**19.186**	3.783	2.578	1.276	4.291	**1.198**	77.935	45.700	**33.110**	45.843	37.541
jar	960	31.069	24.743	19.996	22.614	**17.999**	3.049	1.858	1.192	2.684	**1.112**	71.250	37.405	**31.485**	39.210	33.153
knife	680	13.033	10.277	12.828	9.360	**6.553**	0.543	**0.339**	2.885	0.544	1.213	31.929	**18.662**	23.922	20.264	36.936
lamp	3712	27.753	23.019	15.000	20.443	**10.766**	2.804	2.188	2.445	3.381	**1.282**	60.129	41.450	**26.144**	36.941	37.052
laptop	736	18.469	14.384	15.556	**11.529**	13.178	0.855	**0.506**	0.790	1.551	1.435	40.658	21.525	24.916	**19.728**	40.856
loudspeaker	2560	28.254	24.627	19.158	20.241	**17.635**	2.411	1.833	**1.070**	2.079	1.148	67.650	39.619	**31.280**	36.650	35.478
mailbox	152	22.713	17.961	13.536	18.510	**10.800**	1.640	0.990	0.848	2.250	**0.661**	54.655	29.430	**23.312**	35.135	27.312
microphone	112	23.040	20.631	14.018	21.982	**9.258**	1.974	1.731	2.798	3.985	**1.524**	50.595	33.966	**25.097**	41.002	41.198
microwaves	248	26.206	22.422	19.855	**18.005**	18.308	1.956	1.381	**1.087**	1.591	1.306	63.709	36.277	**32.883**	39.564	37.046
motorbike	544	29.535	22.890	16.233	19.519	**14.931**	2.608	1.542	0.827	1.753	**0.337**	68.983	40.384	27.609	40.384	**20.469**
mug	344	32.684	25.790	22.169	21.857	**20.417**	3.056	1.805	1.301	2.184	**0.307**	72.250	37.005	34.851	37.001	**19.663**
piano	384	29.993	24.023	17.658	19.002	**15.754**	2.845	1.767	0.927	2.169	**0.376**	69.687	40.847	29.207	34.482	**21.198**
pillow	160	24.430	20.216	18.165	17.811	**15.782**	1.640	1.086	1.055	1.404	**0.973**	63.224	31.447	**28.795**	31.769	29.848
pistol	496	20.901	15.997	12.408	14.506	**10.375**	1.275	0.750	**0.653**	1.189	1.069	54.767	32.784	**21.855**	30.082	36.188
flowerpot	968	36.622	28.910	20.387	22.951	**18.600**	4.106	2.576	1.239	2.677	**1.194**	82.931	48.663	**33.464**	42.715	35.906
printer	272	32.040	27.407	17.810	20.495	**16.462**	3.012	2.238	0.921	2.235	**0.858**	75.801	44.694	**29.497**	39.077	30.358
remote	112	18.344	15.208	14.662	12.160	**11.854**	0.875	0.617	0.756	0.675	**0.601**	43.824	23.872	24.275	**21.395**	28.322
rifle	3800	17.733	11.867	12.175	10.491	**7.498**	1.021	**0.462**	2.487	0.729	0.925	41.483	24.095	**22.826**	24.463	32.302
rocket	136	20.458	12.680	10.702	11.864	**7.636**	1.317	**0.511**	0.912	0.918	0.688	47.203	22.858	**19.205**	22.491	28.999
skateboard	248	21.164	14.556	13.753	10.626	**10.483**	1.384	0.622	0.875	**0.593**	0.665	56.167	24.293	23.936	**19.771**	27.137
sofa	5080	25.277	21.194	17.793	**15.228**	15.937	1.750	1.213	**0.928**	1.084	1.277	64.620	36.472	30.066	**28.314**	43.099
stove	352	26.358	22.565	18.159	17.806	**16.740**	2.038	1.445	0.964	1.618	**0.824**	64.162	36.998	29.846	30.074	**26.681**
table	13496	23.052	19.185	16.451	**0.483**	13.752	1.786	1.181	0.898	1.395	**0.564**	55.483	31.098	27.082	31.238	**22.177**
telephone	1744	16.080	14.176	15.824	**11.040**	12.614	0.690	0.544	0.872	0.553	**0.478**	39.098	23.954	26.184	**20.767**	20.838
tower	216	24.196	20.261	15.182	17.883	**13.302**	1.831	1.344	0.874	1.731	**0.595**	59.455	33.713	25.310	31.909	**21.915**
train	624	25.545	18.670	14.791	14.305	**12.824**	1.895	1.055	0.759	1.083	**0.517**	60.620	33.822	25.058	26.239	**21.500**
trash bin	552	28.235	24.318	21.501	21.316	**20.074**	2.283	1.624	1.248	**1.107**	1.209	65.054	36.620	**34.132**	34.752	37.648
watercraft	3104	23.740	17.670	12.740	13.562	**10.818**	1.711	1.001	0.689	1.012	**0.397**	58.678	32.644	22.239	24.141	**18.701**
washer	272	27.989	24.620	20.579	20.915	**19.277**	2.391	1.754	1.164	2.250	**1.033**	59.057	38.436	33.572	37.290	**30.029**
Average		25.459	20.674	16.977	16.597	**14.532**	2.060	1.361	1.151	1.701	**0.893**	60.169	34.008	**28.200**	31.724	30.943

Table 2. Detailed comparison results with other methods of point cloud completion on ShapeNet55 dataset. We list numbers of samples in each category in test set and calculate three metrics: CD with L1 norm (CD-l_1, lower is better) multiplied by 10^3, L2 norm (CD-l_2, lower is better) multiplied by 10^3 and EMD multiplied by 10^3. Note that the metrics are computed by 8,192 points.

ing multi-step point moving paths. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7443–7452, 2021. 3

[6] P. Xiang, X. Wen, Y.-S. Liu, Y.-P. Cao, P. Wan, W. Zheng, and Z. Han. Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5499–5509, 2021. 1, 3

[7] C. Xie, C. Wang, B. Zhang, H. Yang, D. Chen, and F. Wen. Style-based point generator with adversarial rendering for point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4619–4628, 2021. 3

[8] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun. Grnet: Gridding residual network for dense point cloud completion. In *European Conference on Computer Vision*, pages 365–381. Springer, 2020. 2, 3

[9] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 2, 3

[10] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12498–12507, 2021. 2, 3

[11] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018. 1, 2, 3