Supplementary Material for Optimization over Disentangled Encoding: Unsupervised Cross-Domain Point Cloud Completion via Occlusion Factor Manipulation

Jingyu Gong[1⋆], Fengqi Liu[1⋆], Jiachen Xu[1], Min Wang[2], Xin Tan[3], Zhizhong Zhang[3], Ran Yi[1], Haichuan Song[3], Yuan Xie[3⋆⋆], and Lizhuang Ma[1,3,4⋆⋆]

[1] Shanghai Jiao Tong University, Shanghai, China
[2] SenseTime Research, Shanghai, China
[3] East China Normal University, Shanghai, China
[4] Qing Yuan Research Institute, SJTU, Shanghai, China
{gongjingyu,liufengqi,xujiachen,tanxin2017,ranyi}@sjtu.edu.cn, wangmin@sensetime.com, {zzzhang,hcsong}@cs.ecnu.edu.cn, yxie@cs.ecnu.edu.cn, ma-lz@cs.sjtu.edu.cn

Abstract. This supplementary material consists of ten parts. First, we describe our real-time algorithm for generating partial point cloud in detail in Sec 1, and provide more details in training strategy and hyper-parameters in Sec 2. In Sec 3, we show the influence of domain gap on supervised methods quantitatively. Then, we check how far our proposed cross-domain method away from supervised methods trained on target domain in Sec 4. After that, we analyse the impact of iteration steps on completion performance in Sec 5 and investigate the influence of domain factor disentanglement in Sec 6. Next, we conduct extra ablation study of view -point prediction and show the results in Sec 7. Finally, we visualize domain factor and domain-invariant shape factor in Sec 8, provide more visual results in Sec 9, and discuss about the potential of realism metrics for real scan completion in Sec 10.

1 Algorithm for Real-time Partial Point Cloud Generation

In order to scan the complete point cloud from various view-points and generate partial shape alongside the training procedure, we design a real-time implementation of z-buffering [10] for point cloud. The detailed algorithm is listed in Algorithm 1. In our algorithm, we first transform the point cloud into camera coordinate and obtain the depth of each point (Algorithm 1(1-5)). Then, for each pixel, we obtain the minimum distance of each point that can be rendered on this pixel (Algorithm 1(6-13)). Finally, we check the visibility of each point and generate the partial point cloud (Algorithm 1(14-20)).

⋆ Equal Contribution.
⋆⋆ Corresponding Author.

In the training procedure, the average run-time for generating a batch of partial point clouds is 0.0273s on a single GTX 1080Ti, only 12.05% of average run-time for training one batch (*i.e.*0.2265s). This indicates our algorithm can make real-time partial point cloud generation possible for training, and this is because we circumvent the time-consuming loops in z-buffering and utilize the parallelism in tensor operation. We also provide the code for batch partial point cloud generation and more implementation details in "realtime_render.py".

Algorithm 1: Real-time partial point cloud generation.

Input: Complete Shape $\mathcal{P}_c$, Resolution R, Tolerance ϵ, Number of Point N, Box Size L
Output: Partial Shape $\mathcal{P}_p$

▷ Change to camera coordinate.
  az, el = random_view()
  $\mathcal{P}_c$ = rotate($\mathcal{P}_c$,az,el)
▷ Calculate the depth of point to camera plane.
  depth = -L - $\mathcal{P}_c[z]$
▷ Initialize pixel distance to camera plane.
  plane_dis = -2L × ones(R,R,N)
▷ Get the index in plane_dis for changing.
  grid_idx = concat([grid($\mathcal{P}_c[x]$),grid($\mathcal{P}_c$[y])],1)
  idx = concat([grid_idx,arange(N)],1)
▷ Get the plane depth.
  plane_dis[idx] = depth
  plane_depth = max(plane_dis,axis=2)
▷ Obtain the index of visible point.
  plane_mask = plane_depth <= -2L
  plane_depth = 2L ×plane_mask + plane_depth - ϵ
  point_vis = max(plane_dis >= plane_depth,axis=(0,1))
  point_vis_idx = choice(where(point_vis > 0.5),N)
▷ Generate partial point cloud.
  $\mathcal{P}_p$ = $\mathcal{P}_c$[point_vis_idx]
  return $\mathcal{P}_p$

2 Implementation Details

For better understanding of our method, we provide more implementation details of our training strategy and hyper-parameters in Algorithm 2. Here, $\mathcal{L}^s_{rec}$ and $\mathcal{L}^t_{rec}$, $\mathcal{L}^s_{com}$ and $\mathcal{L}^t_{com}$ are the reconstruction losses and completion losses defined by Eq. (5) and Eq. (6) in main paper. $\mathcal{L}_{vp}$ is the view-point prediction loss given in Eq. (1) of our main manuscript, while $\mathcal{L}_{ds}$ and $\mathcal{L}_{di}$ are both binary cross entropy loss for domain classification where a gradient reverse layer [5] is utilized between $\mathcal{L}_{di}$ and domain-invariant shape factor. $\mathcal{L}_{cons}$ and $\mathcal{L}_{op}$ are the factor permutation consistency loss and optimization stage loss formulated in

Eq. (8) and Eq. (12) of our main paper. As shown in our method, we first learn an disentangled representation for cross-domain completion (Algorithm 2(4-23)). Here, we only add factor permutation consistency loss after 120 epoch, when the three factors have been learned preliminarily in encoder. In case of over-fitting to partial shapes in target domain, we only add completion loss for target domain $\mathcal{L}^t_{com}$ after 160 epoch. The batch size can be set to 10 on a single GTX 1080Ti. In the second stage, we further adapt our prediction to each specific instance within the target domain to ensure the consistency between completed point cloud and input partial shapes (Algorithm 2(25-33)). Only 4 iterations in the optimization stage are enough to obtain the desirable performance.

Algorithm 2: OptDE.

Input: Complete Source Shapes P^s_c, Partial Target Shapes P^t_p, Partial Test Shapes P_{test}
Output: Completed Test Shapes P_{com}

▷ Initialization.
Init $P_{com} = []$, init(Enc, Dec, VP, $Dis1$, $Dis2$)
▷ Learn disentangled factors for cross-domain completion.
for *epoch* $\in [0, 1, \cdots, 199]$ **do**
  for *iter_i* $\in [0, 1, \cdots, 144]$ **do**
    $\mathcal{L}_s = \mathcal{L}^s_{com} + 0.5\mathcal{L}^s_{rec}$; backward($\mathcal{L}_s$)
    $\mathcal{L}_f = 0.004 \times (\mathcal{L}_{vp} + \mathcal{L}_{ds} + 0.01\mathcal{L}_{di})$; backward($\mathcal{L}_f$)
    if *epoch* >= 120 **then**
      $\mathcal{L}_c = 0.06\mathcal{L}_{cons}$; backward($\mathcal{L}_c$)
    end
    if *epoch* >= 160 **then**
      $\mathcal{L}_t = \mathcal{L}^t_{com} + 0.15\mathcal{L}^t_{rec}$; backward($\mathcal{L}_t$)
    else
      $\mathcal{L}_t = 0.15\mathcal{L}^t_{rec}$; backward($\mathcal{L}_t$)
    end
    step()
  end
end
save($Enc^\dagger$,$Dec^\dagger$)
▷ Adapt to each input instance with target domain.
for $\mathcal{P} \in P_{test}$ **do**
  $Dec_{init} = Dec^\dagger$, $z_{init} = \mathbf{0} \otimes f_d \otimes f_s$
  for *iter_i* $\in [0, 1, 2, 3]$ **do**
    $\mathcal{L}_{op} = \mathcal{L}_{op}(Dec(z), \mathcal{P}, z)$; backward($\mathcal{L}_{op}$)
    step()
  end
  append(P_{com}, $Dec^*(z^*)$)
end
return P_{com}

3 Comparison with Supervised Methods on Cross-Domain Completion

In order to show the influence of domain gap on supervised point cloud completion methods and the advantage of our cross-domain completion method, several prevailing supervised methods trained on source domain are evaluated on the target domain. Without loss of generality, we take CRN as our source domain and evaluate the performance on target domain ModelNet. The results are reported in Table 1.

Methods	Plane	Car	Chair	Lamp	Sofa	Table	Avg.
PCN [15]	4.25	**9.32**	22.60	70.01	16.69	139.61	43.75
TopNet [11]	4.21	9.36	23.60	67.00	17.71	131.20	42.18
MSN [7]	3.20	11.42	19.81	92.30	23.14	81.15	38.50
VRCNet [8]	5.14	9.34	22.91	46.40	**15.75**	67.65	27.87
OptDE(Ours)	**2.18**	9.80	**14.71**	**39.74**	19.43	**9.75**	**15.94**

Table 1. Cross-domain completion performance of supervised methods and our method on ModelNet. We take [CD↓] as our metric to evaluate the performance of each methods which is scaled by 10^4.

We can see these supervised method trained on CRN cannot generalize well to predict the complete shape in ModelNet due to the domain gap. This also reflects the influence of domain gap on completion performance which usually occurs in real scan completion. On the contrary, our method only needs complete shape from source domain and partial scan from target domain which is usually available for training, and perform much better on cross-domain shape completion.

4 Distance to Supervised Methods Trained on Target Domain

In order to check how far away our method from prevailing supervised methods trained on the target domain, we treat ModelNet as our source domain and evaluate on the shared categories in the test set of target domain CRN. It is noteworthy, we only utilize complete shapes from ModelNet and partial clouds from CRN for training. Meanwhile, all the supervised methods are trained by partial-complete pairs from training set of target domains CRN. We also train our backbone in a supervised manner and test on CRN for better evaluation. We report the results in Table 2.

We can see in this table, with no complete shapes from CRN used for training, there are still some gaps between the completion performance of our method and prevailing supervised methods trained on paired partial-complete shapes from

Methods	pair	compl.	Plane	Car	Chair	Lamp	Sofa	Table	Avg.
PCN [15]	✓	✓	3.5	6.4	11.0	11.6	11.5	10.4	9.1
TopNet [11]	✓	✓	4.1	7.8	13.4	14.8	16.0	12.9	11.5
MSN [7]	✓	✓	2.9	7.1	10.6	9.3	12.0	9.6	8.6
CRN [12]	✓	✓	2.3	6.2	8.8	8.5	11.3	9.3	7.7
Supervised(Ours)	✓	✓	3.5	6.2	12.8	12.7	11.9	12.0	9.9
OptDE(Ours)	✗	✗	5.0	10.0	18.1	20.2	20.2	18.5	15.3

Table 2. Completion results on CRN. We take [CD↓] as our method to evaluate the performance of each methods which has been scaled by 10^4. pair: paired partial-complete point clouds from CRN are required for training. compl.: complete point clouds from CRN are required for training.

CRN. This may spur more works studying how to better handle the output domain gap and complete cross-domain shapes under no knowledge of complete shapes from target domain.

5 Iteration Steps in Optimization

Here, we attempt to analyse the influence of iteration steps during optimization on final completion performance. Following the setting of our main manuscript, we take CRN as our source domain and ModelNet as our target domain. Given the good initial prediction provided by disentangled encoding, we take different iteration steps (*e.g.*$1, 2, 4, 8, 16$) in optimization stage to make the output predictions consistent to input shapes. We report the Chamfer Distance ($\times 10^4$) between the Ground Truth and initial predictions as well as optimized predictions after different iteration steps in Table 3.

Iterations	Plane	Car	Chair	Lamp	Sofa	Table	Avg.
Step 0	2.19	9.80	15.11	42.94	21.45	10.26	16.96
Step 1	2.25	9.80	14.90	41.69	20.47	9.88	16.50
Step 2	2.16	9.80	14.78	41.06	19.88	9.77	16.24
Step 4	2.18	9.80	14.71	39.74	19.43	9.75	15.94
Step 8	2.21	9.80	14.73	39.18	19.83	9.87	15.94
Step 16	2.20	9.79	14.74	39.15	20.31	9.85	16.01

Table 3. Ablation study on iteration steps in optimization procedure where [CD↓]($\times 10^4$) is taken to evaluate the performance.

It shows optimization can converge within a few iterations and 4 steps are usually enough for most categories to obtain desirable performance, and this is

attributed to the good initialization provided by the disentangled encoding as claimed in the main paper.

6 Disentanglement of Domain Factor

Compared with previous cross-domain completion methods [2,13] which only learnt domain-invariant features for completion, we additionally disentangle domain factor to preserve domain patterns in the output prediction and show the advantage in the main paper. Another choice to preserve domain features in output prediction is keeping domain and shape features entangled while disentangling occlusion factor. Here, we remove the discriminators which disentangle the domain factor and domain-invariant shape factor, and only permute the occlusion factor in factor permutation consistency regularization. We take CRN as source domain and report the results of entangled domain-shape encoding (ent.) and disentangled domain, domain-invariant shape encoding (dist.) on ModelNet in Table 4.

Methods	Plane	Car	Chair	Lamp	Sofa	Table	Avg.
Pcl2pcl [2]	18.53	17.54	43.58	126.80	38.78	163.62	68.14
ShapeInversion [16]	3.78	15.66	22.25	60.42	22.25	125.31	41.61
Cycle4Compl. [13]	5.77	11.85	26.67	83.34	22.82	21.47	28.65
OptDE(ent.)	2.45	10.39	14.81	40.61	21.99	11.27	16.92
OptDE(dist.)	2.18	9.80	14.71	39.74	19.43	9.75	15.94

Table 4. Ablation study of disentangled domain, domain-invariant shape factor on ModelNet dataset. We take [CD↓]($\times 10^4$) as our metric for performance evaluation.

We can see only manipulating occlusion factor while keeping domain and shape feature entangled can still outperform previous methods which just extract domain-invariant features because domain information can also be preserved in the output prediction to some extent. Even though, disentangling domain factor and domain-invariant shape factor can explicitly preserve domain information in the completed point cloud, thus perform better than entangled representation of domain and shape factor.

7 View-Point Prediction.

We choose to directly predict the rotation angle for view-point prediction. Meanwhile, another choice is to predict the rotation matrix $\hat{\mathcal{M}}_{rot}$ which is continuous. Here, we predict the rotation matrix from an intermediate 6D representation through a Gram-Schmidt-like process [17]. After that, the Ground Truth rotation matrix of view-point $\mathcal{M}_{rot}$ will guide the learning of occlusion factor through

supervising the view-point predictor module $\mathcal{L}_{vp} = \|\mathcal{M}_{rot} - \hat{\mathcal{M}}_{rot}\|_2^2$. Here, we report the results of supervising through rotation matrix (mat.) and rotation angle (ang.) prediction in Table 5.

Methods	Plane	Car	Chair	Lamp	Sofa	Table	Avg.
OptDE(mat.)	2.10	11.26	14.82	50.07	23.63	10.25	18.69
OptDE(ang.)	2.18	9.80	14.71	39.74	19.43	9.75	15.94

Table 5. Ablation study of occlusion factor supervision strategy on ModelNet. We use [CD↓]($\times 10^4$) to compare these two strategies.

We can see the supervision through rotation angles works better, and this is because the parameters of rotation angles (*i.e.*, azimuth and elevation) are independent and more explicit than the rotation matrix given partial shapes.

8 Factor Visualization

Here, we visualize the learnt domain factor and domain-invariant shape factor for better understanding of our method. To visualize the learnt domain factor, we first obtain the domain factors (f_d^M and f_d^C) of samples from ModelNet $\mathcal{P}^M$ and CRN $\mathcal{P}^C$ through trained $Enc^\dagger$. Then, we swap domain factors between samples from ModelNet and CRN and generate new shapes. Generated point clouds with ModelNet/CRN domain factors are denoted as $\mathcal{P}^{\hat{M}}/\mathcal{P}^{\hat{C}}$. Later, we feed $\mathcal{P}^{\hat{M}}$ and $\mathcal{P}^{\hat{C}}$ into $Enc^\dagger$ again to obtain the domain factors of newly generated samples $f_d^{\hat{M}}$ and $f_d^{\hat{C}}$. Here, we visualize f_d^M, f_d^C, $f_d^{\hat{M}}$ and $f_d^{\hat{C}}$ after t-SNE in Fig 1 (a). It shows the domain distribution of shapes can be changed through manipulating the learnt domain factors. Besides, we show the learnt domain-invariant shape factors of samples from ModelNet f_s^M and CRN f_s^C in Fig 1 (b) which shows the learnt shape factor is invariant to domain.

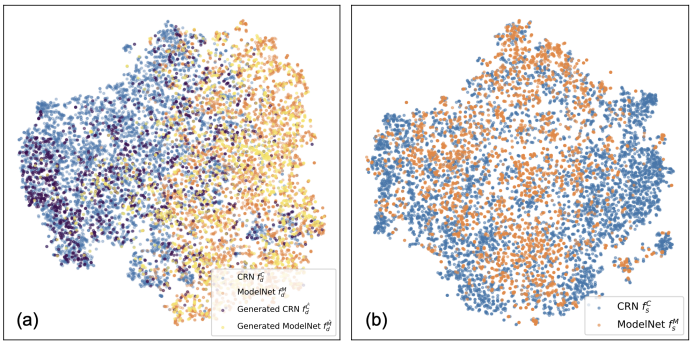

Fig. 1. T-SNE visualization of (a) domain factor f_d and (b) domain-invariant shape factor f_s.

9 Visualization Results

In this section, we present more visualization results of our method on real scans from ScanNet [3], MatterPort3D [1] and KITTI [6] as well as on synthesized datasets ModelNet [14] and 3D-FUTURE [4]. We present additional visual results of our method and previous cross-domain completion methods on real scans from ScanNet, MatterPort3D and KITTI in Figure 2. In Figure 3, we provide more qualitative results on the test set of 3D-FUTURE. Finally, we visualize more completion results on test set of ModelNet in Figure 4.

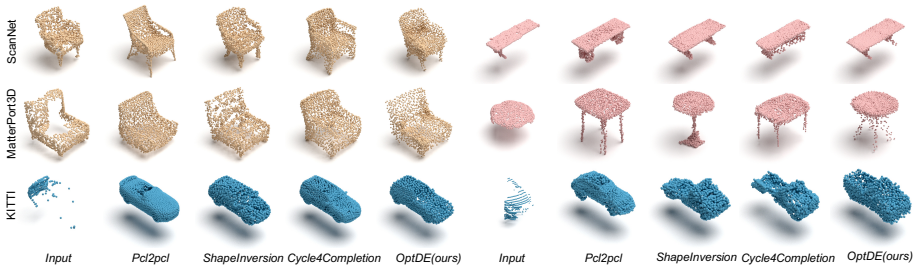

Fig. 2. More visualization results on the data of ScanNet, MatterPort3D and KITTI. Partial point clouds, results of pcl2pcl ShapeInversion, Cycle4Completion and our methods are presented separately from the left to the right.

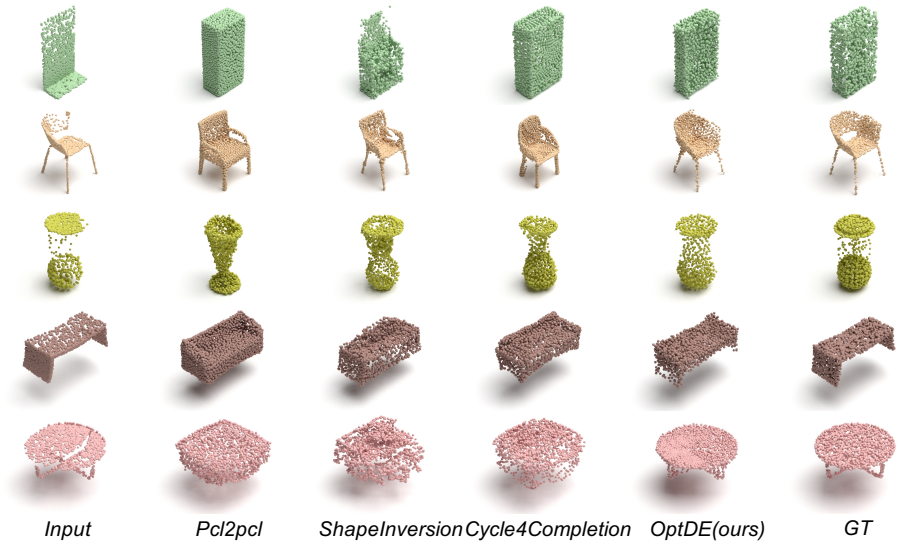

Fig. 3. More visualization results on the test set of 3D-FUTURE. The left-most images are input partial shapes and the following images are completed point cloud predicted by pcl2pcl, ShapeInversion, Cycle4Completion and our method as well as Ground Truth respectively.

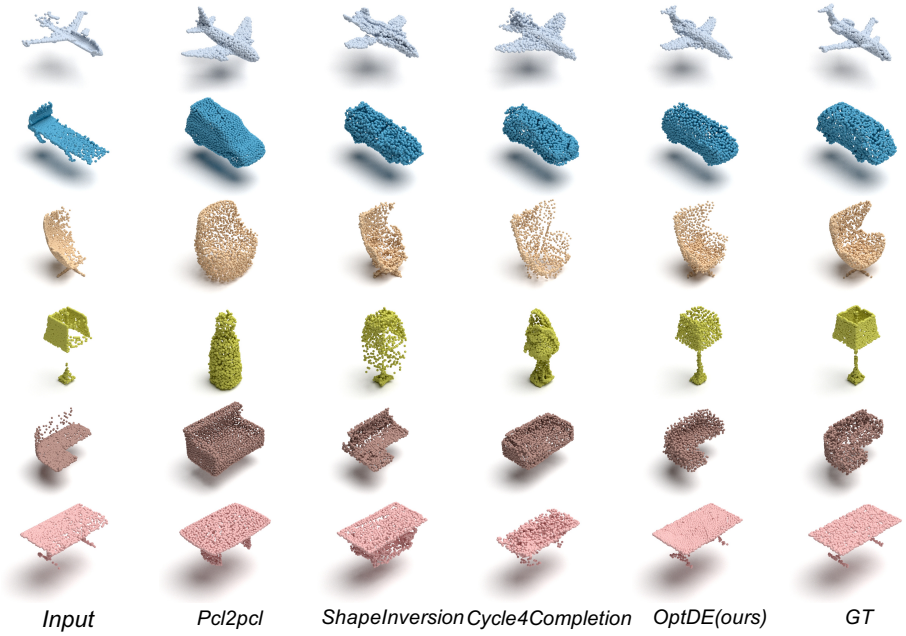

Fig. 4. More visualization results on the test set of ModelNet. The images from the left to the right are input partial clouds, predictions given by pcl2pcl, ShapeInversion, Cycle4Completion and ours, as well as target shapes separately.

10 Future Work Discussion

As complete shapes of real scans are usually unavailable for the evaluation of cross-domain point cloud completion, previous works [16] utilized metrics like UCD and UHD to check the fidelity of completed point clouds. As suggested by our reviewers, we attempt to take realism metric like Frechet Distance (FD) to measure the similarity between predicted complete shapes and realistic complete ones. Concretel, we take the form of Frechet Point Cloud Distance (FPD) [9] for evaluation which is a common metric in point cloud generation, and FPD used here is formulated as:

$$\mathrm{FPD}(\mathbb{R},\mathbb{P}) = \|\mu_{\mathbb{R}} - \mu_{\mathbb{P}}\|_2^2 + \mathrm{Tr}\left(\Sigma_{\mathbb{R}} + \Sigma_{\mathbb{P}} - 2\left(\Sigma_{\mathbb{R}}\Sigma_{\mathbb{P}}\right)^{\frac{1}{2}}\right) \tag{1}$$

where $\mathbb{R}$ and $\mathbb{P}$ are the feature distributions of realistic 3D shapes and predicted complete shapes, while μ and Σ are the mean and covariance matrix of corresponding features. Here, we take complete shapes in 3D-FUTURE as realistic ones due to the lack of complete real shapes. Results show that ours is more similar to realistic complete shapes than ShapeInversion in ScanNet (chair: 1.74 v.s. 1.80, table: 2.14 v.s. 2.54) and MP3D (chair: 1.60 v.s. 1.74, table: 2.33 v.s. 3.19). We believe that further researches on the realism metrics for real scan completion will be a promising future work.

References

1. Chang, A., Dai, A., Funkhouser, T., Halber, M., Niebner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3d: Learning from rgb-d data in indoor environments. In: 2017 International Conference on 3D Vision (3DV). pp. 667–676. IEEE Computer Society (2017)
2. Chen, X., Chen, B., Mitra, N.J.: Unpaired point cloud completion on real scans using adversarial training. In: International Conference on Learning Representations (2020)
3. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5828–5839 (2017)
4. Fu, H., Jia, R., Gao, L., Gong, M., Zhao, B., Maybank, S., Tao, D.: 3d-future: 3d furniture shape with texture. arXiv preprint arXiv:2009.09633 (2020)
5. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: International conference on machine learning. pp. 1180–1189. PMLR (2015)
6. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 3354–3361. IEEE (2012)
7. Liu, M., Sheng, L., Yang, S., Shao, J., Hu, S.M.: Morphing and sampling network for dense point cloud completion. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 11596–11603 (2020)
8. Pan, L., Chen, X., Cai, Z., Zhang, J., Zhao, H., Yi, S., Liu, Z.: Variational relational point completion network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8524–8533 (2021)
9. Shu, D.W., Park, S.W., Kwon, J.: 3d point cloud generative adversarial network based on tree structured graph convolutions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3859–3868 (2019)
10. Straßer, W.: Schnelle kurven-und flächendarstellung auf grafischen sichtgeräten. Ph.D. thesis (1974)
11. Tchapmi, L.P., Kosaraju, V., Rezatofighi, H., Reid, I., Savarese, S.: Topnet: Structural point cloud decoder. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 383–392 (2019)
12. Wang, X., Ang Jr, M.H., Lee, G.H.: Cascaded refinement network for point cloud completion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 790–799 (2020)
13. Wen, X., Han, Z., Cao, Y.P., Wan, P., Zheng, W., Liu, Y.S.: Cycle4completion: Unpaired point cloud completion using cycle transformation with missing region coding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13080–13089 (2021)
14. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
15. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. In: 2018 International Conference on 3D Vision (3DV). pp. 728–737. IEEE (2018)
16. Zhang, J., Chen, X., Cai, Z., Pan, L., Zhao, H., Yi, S., Yeo, C.K., Dai, B., Loy, C.C.: Unsupervised 3d shape completion through gan inversion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1768–1777 (2021)

17. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5745–5753 (2019)