Point Cloud Upsampling via Disentangled Refinement

Ruihui Li Xianzhi Li* Pheng-Ann Heng Chi-Wing Fu
The Chinese University of Hong Kong
{lirh,xzli,pheng,cwfu}@cse.cuhk.edu.hk

Figure 1. In the top row, we show (a) a sparse real-scanned point set from [23], followed by upsampled results (16×) produced by (b) MPU [26], (c) PU-GAN [12], and (d) our method. In the bottom row, we show the associated reconstructed 3D meshes produced by the ball-pivoting algorithm [2]. Clearly, our method outperforms others on the local uniformity, contributing a better surface reconstruction.

Abstract

Point clouds produced by 3D scanning are often sparse, non-uniform, and noisy. Recent upsampling approaches aim to generate a dense point set, while achieving both distribution uniformity and proximity-to-surface, and possibly amending small holes, all in a single network. After revisiting the task, we propose to disentangle the task based on its multi-objective nature and formulate two cascaded sub-networks, a dense generator and a spatial refiner. The dense generator infers a coarse but dense output that roughly describes the underlying surface, while the spatial refiner further fine-tunes the coarse output by adjusting the location of each point. Specifically, we design a pair of local and global refinement units in the spatial refiner to evolve a coarse feature map. Also, in the spatial refiner, we regress a per-point offset vector to further adjust the coarse outputs in fine scale. Extensive qualitative and quantitative results on both synthetic and real-scanned datasets demonstrate the superiority of our method over the state-of-the-arts. The code is publicly available at https://github.com/liruihui/Dis-PU.

1. Introduction

Point clouds, as a compact representation of 3D data, are widely explored by both traditional and deep-learning-based methods for many applications [6, 4, 18], *e.g.*, self-driving cars, robotics, rendering, and medical analysis, etc. However, raw point clouds produced by 3D scanning are often locally sparse and non-uniform, possibly with small holes on the object surface; see a real-scanned example shown on the top of Figure 1(a). Obviously, we need to amend such raw data, before we can effectively use it for rendering, analysis, or general processing.

The goal of point cloud upsampling is not limited to generating a dense point set from the sparse input. Very importantly, the generated points should also faithfully locate on the underlying surface and cover the surface with a uniform distribution. As an inference-based problem, these goals are very demanding, due to the limited information available in the sparse input. Besides being sparse, the input points can be non-uniform and noisy, and they may not well represent fine structures (if any) on the underlying surface.

Benefited from data-driven machine learning and deep neural network models, several deep-learning-based methods [28, 27, 26, 12, 21] have been proposed for point cloud upsampling and they demonstrated superior performance, compared with traditional methods [1, 17, 10]. The general approach taken in existing learning-based methods is that they first design an upsampling module to expand the number of points in the feature space, then formulate losses to constrain the output points for the distribution uniformity and proximity-to-surface. In other words, the designed upsampling module is expected not only to infer and generate dense points from the sparse input, but also to produce points that are uniform, clean, and faithfully located on the

*corresponding author

underlying surface. However, it is very hard for a network to meet all the requirements at the same time. Therefore, the dense points produced by existing works still tend to be non-uniform or retain excessive noise (see the top results in Figures 1 (b) & (c)), thus resulting in low-quality reconstructed meshes (see results in the bottom row).

After revisiting the point cloud upsampling task, we propose to disentangle the task into two sub-goals: (i) to first generate a *coarse but dense point set* with less details to roughly describe the underlying surface, and then (ii) to *refine the coarse points* to better cover the underlying surface for distribution uniformity and proximity-to-surface. To do so, we formulate an end-to-end disentangled refinement framework, which consists of two cascaded sub-networks, *a dense generator* and *a spatial refiner*, which are designed to aim for sub-goals (i) and (ii), respectively. Particularly, we design the spatial refiner with a pair of local and global refinement units to evolve the coarse feature map inside the refiner to take into account both the local and global geometric structures. Further, inspired by the residual-learning strategy [8], we formulate the spatial refiner to regress a per-point offset vector for fine-tuning the coarse outputs by adjusting the location of each point. Compared with directly predicting the final refined 3D point coordinates, regressing a small residual is much easier for the network.

Compared with current upsampling methods [28, 26, 12], our disentangled refinement pipeline assigns lower requirements to each sub-network, so that both the dense generator and the spatial refiner could be more focused on their own sub-goals. In addition, the cascading scheme allows the two sub-networks to complement each other during network learning, thus leading to a substantial performance boost; see Figure 1(d). Extensive experimental results demonstrate that our method outperforms others on both real-scanned and synthetic inputs.

2. Related Work

Optimization-based upsampling. To generate new points from the inputs, optimization-based methods typically rely on hand-crafted priors. Alexa *et al.* [1] introduced an early work that inserts new points at the vertices of the Voronoi diagram, which is computed based on the moving-least-squares surface. Later, a locally optimal projection operator was proposed by Lipman *et al.* [17], where points are resampled based on the L_1 norm. To upsample point cloud in an edge-aware manner, Huang *et al.* [10] proposed to first upsample points away from the edges then progressively move points towards the edge singularities. Later, Wu *et al.* [25] introduced a point-set consolidation method by augmenting surface points into deep points that lie on the meso-skeleton of the shape. Overall, optimization-based methods may fail when the prior assumptions are not satisfied.

Deep learning-based upsampling. Inspired by the success of PointNet [20], many deep learning methods were proposed for assorted tasks on point cloud processing, from high-level tasks like classification [16, 31, 14] and object detection [11, 19] to low-level tasks like completion [29, 3], denoising [22, 9], and other applications [24, 5, 13, 15].

For the point cloud upsampling task, Yu *et al.* [28] introduce PU-Net, the first attempt based on deep learning to learn to extract multi-scale features and expand a point set via a multi-branch convolution in the feature space. Later, they introduce EC-Net [27] to achieve edge-aware point cloud upsampling, which further enhances the surface reconstruction quality. Soon after that, Wang *et al.* [26] proposed MPU, a network that progressively upsamples point patches in multiple steps, while Li *et al.* [12] proposed PU-GAN by leveraging the generative adversarial network to learn to synthesize points with a uniform distribution in the latent space. Very recently, Qian *et al.* [21] proposed PU-GeoNet to first generate samples in a 2D domain and then use a linear transform to lift up the samples to 3D.

After revisiting deep-learning-based methods for point cloud upsampling, we found that existing works all rely on a single network to meet all the various goals of point cloud upsampling, *i.e.*, dense point set generation, faithfulness to underlying surface, distribution uniformity, hole amendment, etc. To better meet the multi-objective nature of the task, we propose a new approach to disentangle the task into two cascaded networks and demonstrate substantial improvements over the prior works.

3. Method

3.1. Overview

Essentially, 3D scanning is a sampling problem in the 3D physical space, while upsampling is a prediction problem, aiming to infer more samples on the original surface, given the sparse samples obtained in the scanning. Given a point set $\mathcal{P}$ of N points, which is typically sparse, non-uniform, and noisy, the point cloud sampling task aims to generate a dense point set, say $\mathcal{Q}$ of rN points, for a given upsampling rate r. These upsampled points should (i) faithfully describe the underlying object surface and (ii) cover the surface with a uniform distribution. This upsampling task is very challenging, since we need to infer new knowledge from the sparse input, in which the information about the original geometry is not completely represented.

Different from the existing approaches that try to meet the various goals in upsampling all in a single network, we propose to disentangle the upsampling task into two sub-goals, where we first generate a coarse but dense point set and then refine these points over the underlying surface to improve the distribution uniformity. Before giving the details of our method, we first discuss our key insights:

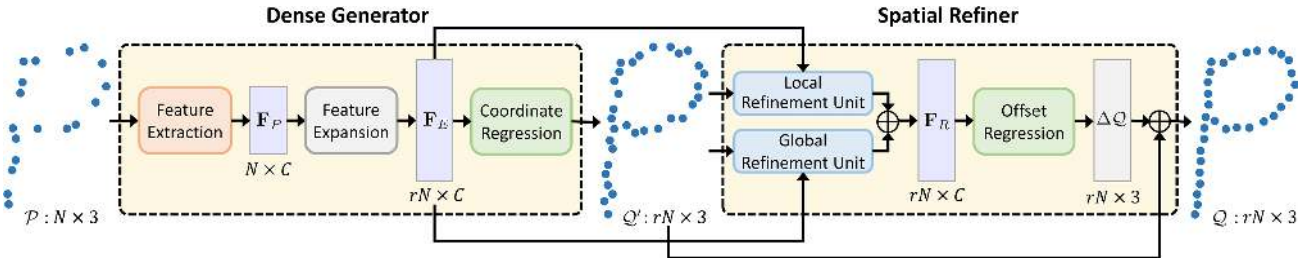

Figure 2. An illustration of our framework. Given sparse input $\mathcal{P}$ of N points, the dense generator extracts feature map $\mathbf{F}_P$ from the input, generates the expanded feature map $\mathbf{F}_E$, then produces a coarse but dense point set $\mathcal{Q}'$ of rN points, where r is the upsampling rate. Next, the spatial refiner consumes both $\mathcal{Q}'$ and the associated $\mathbf{F}_E$ to obtain the refined feature map $\mathbf{F}_R$ via a pair of local and global refinement units. We then regress offsets $\Delta\mathcal{Q}$ from $\mathbf{F}_R$, and output the final refined dense points $\mathcal{Q}$ by $\mathcal{Q}' + \Delta\mathcal{Q}$.

- First, we propose an end-to-end disentangled refinement framework with two cascaded sub-networks: one to generate dense points that roughly locate on the underlying surface and the other to aim for proximity-to-surface and distribution uniformity. Thus, each sub-network can better focus on its specific sub-goal, while complementing each other in the upsampling task.
- Second, with the help from the spatial refiner sub-network, the dense generator sub-network does not need a complicated structure. Hence, having a simple yet effective structure can enable it to expand features with higher flexibility and increase the upsampling rate without introducing extra network parameters.
- Third, we design the spatial refiner with both local and global refinements, such that we can leverage their complementary strengths to locally improve the distribution uniformity and proximity-to-surface, and globally explore similar structures on the surface.

Figure 2 shows the overall framework of our method. Given a sparse point set $\mathcal{P}$, we first feed it into our *dense generator* to generate the dense output $\mathcal{Q}'$ with rN points. At present, $\mathcal{Q}'$ may still be non-uniform and noisy like $\mathcal{P}$, as illustrated in the figure's toy example. Next, we feed it into our *spatial refiner* to further regress a per-point offset vector $\Delta\mathcal{Q}$, which is used to adjust the location of each point in $\mathcal{Q}'$, such that the refined dense points $\mathcal{Q}$ can faithfully locate on the underlying surface, while being more uniform. In the following, we first present the details of the dense generator and spatial refiner in Sections 3.2 and 3.3, respectively. Then, we give the details of the patch-based end-to-end network training in Section 3.4.

3.2. Dense Generator

Given $\mathcal{P} \in \mathbb{R}^{N\times 3}$, our dense generator produces the upsampled coarse points $\mathcal{Q}' \in \mathbb{R}^{rN\times 3}$. Similar to existing upsampling approaches [28, 26, 12], we also expand the number of points in the feature space.

Specifically, as shown in the left-side of Figure 2, we first employ a feature extraction unit to embed the feature map $\mathbf{F}_P \in \mathbb{R}^{N\times C}$ from $\mathcal{P}$, where C is the number of feature channels. Here, we follow [26] to use the same feature extraction unit by considering the efficiency and effectiveness. Please refer to [26] for the details of this unit. Next, we feed $\mathbf{F}_P$ into a feature expansion unit to generate the expanded feature map $\mathbf{F}_E \in \mathbb{R}^{rN\times C}$. As discussed in Section 3.1, with the help from the cascaded spatial refiner, the dense generator only needs to generate a dense point set to roughly locate on the underlying surface. Hence, in the feature expansion unit, we adopt the commonly-used expansion operation by duplicating $\mathbf{F}_P$ with r copies and concatenating with a regular 2D grid to obtain $\mathbf{F}_E$. Although such operation may introduce redundant information or extra noise, these problems could be rectified by the subsequent spatial refiner. Lastly, $\mathcal{Q}'$ is generated by regressing the point coordinates from $\mathbf{F}_E$ via multi-layer perceptrons (MLPs).

3.3. Spatial Refiner

Considering that $\mathcal{Q}'$ may still be noisy and non-uniform, we thus design a spatial refiner to further fine-tune the spatial location of each point in $\mathcal{Q}'$ and generate a high-quality dense point set $\mathcal{Q}$, which lies on the underlying surface and also distributes uniformly.

To do so, as shown in the right-side of Figure 2, we first feed the coarse $\mathcal{Q}'$ and the associated coarse feature map $\mathbf{F}_E$ into both local and global refinement units, which are detailed later. Next, we sum the two outputs generated by the two refinement units to obtain the refined feature map $\mathbf{F}_R \in \mathbb{R}^{rN\times C}$. Then, instead of directly regressing the refined point coordinates, we adopt residual learning to regress the per-point offset $\Delta\mathcal{Q}$. The reason behind is that, compared with the relative offset, the absolute point coordinates are more diverse and have a wide distribution in 3D space. Hence, it is difficult for the network to synthesize points without introducing extra noise, while still preserving the uniformity and shape structures. Lastly, the final output $\mathcal{Q}$ is obtained by $\mathcal{Q}' + \Delta\mathcal{Q}$.

Local refinement unit aims to evolve $\mathbf{F}_E$ by considering the local geometric structures. Figure 3(a) shows the detailed architecture. Specifically, we first employ KNN

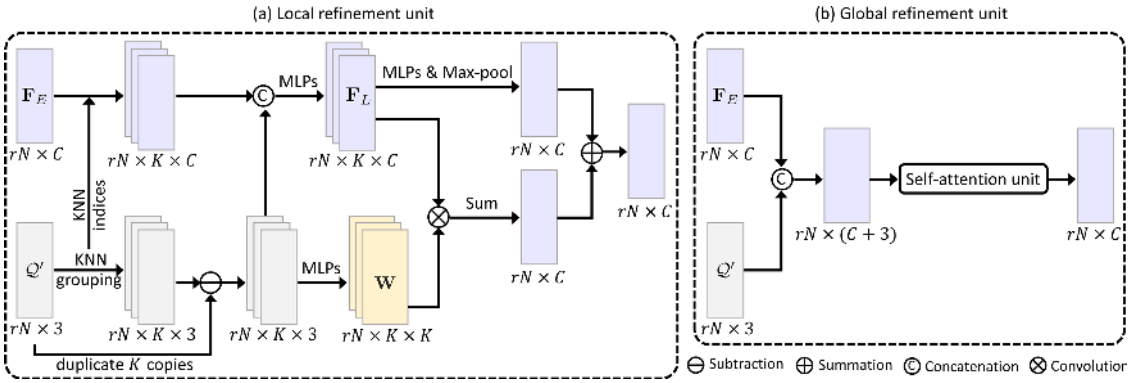

Figure 3. The architecture of (a) the local refinement unit and (b) the global refinement unit.

grouping on $\mathcal{Q}'$ to search K-nearest neighbors and group the associated neighbor points together to obtain a stacked $rN \times K \times 3$ point volume. At the same time, we employ the same nearest neighbor indices to group $\mathbf{F}_E$ into an $rN \times K \times C$ feature volume. Then, we duplicate $\mathcal{Q}'$ with K copies and apply a subtraction operation on the duplicated and the grouped point volumes, which helps encode local information. We then concatenate the subtracted point volume with the grouped feature volume and apply MLPs to obtain the encoded local feature volume $\mathbf{F}_L \in \mathbb{R}^{rN \times K \times C}$.

Next, to obtain the local point feature over $\mathbf{F}_L$, a common routine is to apply MLPs followed by a max-pooling along the K-dimension. However, to account for the relative importance among the K neighbors, we further regress a spatial weight $\mathbf{W}$ (see the light yellow volume in Figure 3 (a)) from the subtracted point volume. Then, we modify $\mathbf{F}_L$ via a convolution with $\mathbf{W}$, followed by a summation along the K-dimension to obtain the weighted $rN \times C$ feature map. Lastly, we sum the weighted feature map as the final refined local features.

Global refinement unit aims to refine $\mathbf{F}_E$ by considering the overall shape structure. As shown in Figure 3(b), instead of feeding only $\mathbf{F}_E$ to the refinement unit, we concatenate $\mathbf{F}_E$ and $\mathcal{Q}'$ together as the input to avoid losing the overall shape structure. Next, we adopt the widely-used self-attention unit [30] to obtain the refined global feature map, since this unit regresses attention weights among all the rN points, thus introducing long-range context dependencies. For brevity, we will not describe the details of this attention unit; please refer to [30] if needed.

3.4. Patch-based End-to-end Training

Since point cloud upsampling is a low-level task that requires us to focus more on the local geometric structures, we thus adopt the patch-based training strategy, as all the existing upsampling approaches did. During training, for each input sparse point set $\mathcal{P}$ and its associated target dense point set $\hat{\mathcal{Q}}$, our framework predicts both $\mathcal{Q}'$ and $\mathcal{Q}$. Hence, we formulate our objective function to encourage the geometric consistency between $\mathcal{Q}'\&\hat{\mathcal{Q}}$, and between $\mathcal{Q}\&\hat{\mathcal{Q}}$:

$$\mathcal{L} = \mathcal{L}_{\text{CD}}(\mathcal{Q}', \hat{\mathcal{Q}}) + \lambda \mathcal{L}_{\text{CD}}(\mathcal{Q}, \hat{\mathcal{Q}}), \tag{1}$$

where $\mathcal{L}_{\text{CD}}(\cdot)$ means the Chamfer Distance (CD) [7] to measure the average closest point distance between two point sets. The parameter λ controls the relative importance of each term. In the early stage of network training, we set a small λ, so that the network focuses more on the training of the dense generator to produce a more reliable $\mathcal{Q}'$. As the training progresses, we gradually increase λ to let our spatial refiner to be fully trained.

Note that, we also tried to combine Eq. (1) with the repulsion loss [28] to encourage the distribution uniformity. However, we found that this repulsion loss does not contribute too much in our work, because our method can already generate a relatively uniform dense point set benefited from the disentangled refinement scheme, even without any losses to constrain the uniformity distribution.

4. Experiments

4.1. Experimental Settings

Datasets. We employ both synthetic and real-scanned datasets in our experiments. For the synthetic dataset, we use the benchmark dataset provided by [12] with 120 training and 27 testing objects. For each training object, we follow [12] to crop 200 overlapped patches, thus resulting in totally 24,000 training surface patches. On each surface patch, we uniformly sample rN points as target $\hat{\mathcal{Q}}$, and then randomly downsample N points from $\hat{\mathcal{Q}}$ as the training input $\mathcal{P}$. For each testing shape, we follow MPU [26] and PU-GAN [12] to sample ~20,000 uniform points using Poisson disk sampling as $\hat{\mathcal{Q}}$ for quantitative evaluation, and generate 1,024 non-uniform points for testing.

For real-scanned dataset, we use ScanObjectNN [23], which contains 2,902 point cloud objects in 15 categories. Each object has 2,048 points. Since no target dense points are provided in ScanObjectNN, we just use this dataset for testing. Hence, during testing, we have both synthetic and

real-scanned point clouds. For each testing point cloud with 2,048 points, we use farthest sampling to pick seeds and extract a local patch of N points per seed. We then feed these patches to the network for testing and combine the upsampled results as the final output.

Evaluation metrics. For quantitative evaluation, we consider three widely-used evaluation metrics: (i) Chamfer distance (CD), (ii) Hausdorff distance (HD), and (iii) Point-to-surface (P2F) distance using the original testing objects. A lower evaluation metric indicates a better performance.

Comparison methods. To demonstrate the effectiveness of our method, we compare it with three state-of-the-art point cloud upsampling methods, including PU-Net [28], MPU [26], and PU-GAN [12]. We use their released public code and follow the same setting in the original papers to retrain their networks using our prepared training data. Note that, for the recent work PUGeoNet [21], we cannot provide the comparison results without available code so far.

Implementation details. In experiments, we set $N = 256$. We train our network with a batch size of 28 for 400 epochs on the TensorFlow platform. For each patch, we apply random scaling, rotation, and point perturbation to avoid overfitting. The Adam optimizer is used with the learning rate of 0.001, which is linearly decreased by a decay rate of 0.7 per 40 epochs until 10^{-6}. The parameter λ in Eq. (1) is linearly increased from 0.01 to 1.0 as the training progresses.

4.2. Results on Real-scanned Dataset

First, we compared our method with state-of-the-arts on real-scanned test inputs. Besides the results shown earlier in Figure 1, we further show more visual comparison results in Figure 7 (see page 8), where we set $r = 16$. For each object, the top row shows the upsampled points by each method, and the bottom row shows the associated reconstructed 3D meshes using ball-pivoting surface reconstruction algorithm [2]. Note that, since PU-Net is an early work with not very promising results, we thus omit its results in visual comparisons. As shown in the top row of Figure 7(a), to upsample the real-scanned sparse inputs is very challenging, since these points are not only noisy and non-uniform, but also exhibit many small holes and structural defects. Thus, reconstructing meshes directly from sparse inputs often results in incomplete surfaces with many holes; see the bottom results in (a). Comparing the upsampled points produced by various methods, the other methods tend to retain noise in their results, or fail to generate a uniform output, thus resulting in low-quality reconstructed meshes with small holes or rough surfaces. On the contrary, our method enables to produce uniform dense points with low deviations to the underlying object surface. Hence, the reconstructed meshes from our upsampled points can well describe the geometric structures with smooth and complete surfaces. More real-scanned comparisons can be found in our supplementary material.

Table 1. Quantitative comparisons by using our method and state-of-the-arts. The units of CD, HD, and P2F are all 10^{-3}.

Methods	4X				16X			
	Size	CD	HD	P2F	Size	CD	HD	P2F
PU-Net [28]	10.1M	0.844	7.061	9.431	24.5M	0.699	8.594	11.619
MPU [26]	23.1M	0.632	6.998	6.199	92.5M	0.348	7.187	6.822
PU-GAN [12]	9.57M	0.483	5.323	5.053	9.57M	0.269	7.127	6.306
Our	13.2M	**0.315**	**4.201**	**4.149**	13.2M	**0.199**	**4.716**	**4.249**

4.3. Results on Synthetic Dataset

Next, we compared our method with state-of-the-arts on synthetic test models provided by [12]. Figure 4 shows the visual comparisons on three sparse inputs, where we set r=16. Comparing the dense points produced by our method (e) and others (b-d) with the target (a), we can see that other methods tend to introduce excessive noise (*e.g.*, (b)), cluster points together with a non-uniform distribution (*e.g.*, (c)), or destroy some tiny structures (*e.g.*, (d)) in the results. In contrast, our method produces the most similar visual results to the target points, and our dense points can well preserve tiny local structures with a uniform point distribution; see particularly the blown-up view in Figure 4. Besides, we show also the associated error maps, where the colors reveal the nearest distance for each point in target point set to the predicted point set. We can see that the errors of our upsampled results are the lowest (*i.e.*, most points are blue), which is also verified by both CD and HD values. More comparison results can be found in the supplemental material.

Table 1 shows the quantitative comparisons on all the synthetic test models under different upsampling rates. We can see that our method achieves the lowest values on all the evaluation metrics in terms of both upsampling rates. Note that, different from PU-Net and MPU, the number of learnable parameters in our network will not increase as r increases. Hence, our method has good scalability to a large upsampling rate. More importantly, when r increases, the advantages of our method compared with others become more obvious. The reason behind is that, the prediction difficulty will significantly increase given a large r for existing approaches. However, thanks to the disentangled refinement scheme, our method has better adaptability.

4.4. Noise Robustness Test

We explored the noise robustness of our method by adding Gaussian noise of different levels to the synthetic test inputs. Figure 5 shows the visual comparisons. Clearly, our method achieves more uniform upsampling results (d & g) without excessive noise, under both upsampling rates. The quantitative comparisons are summarized in Table 2, where $r = 16$ and we show the CD values. Obviously, our method produces the lowest values across all the noise levels with a significant margin, compared to others.

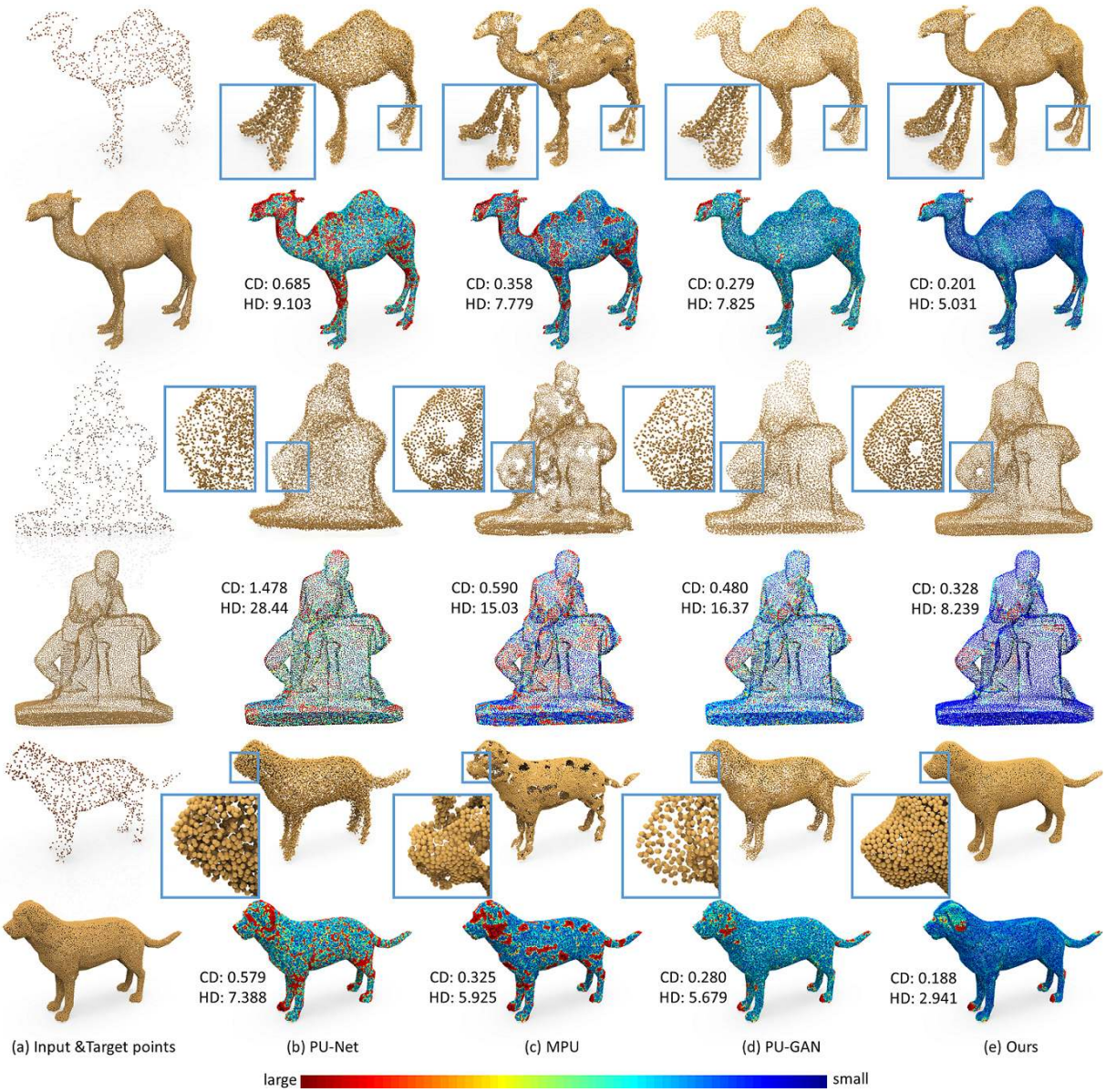

Figure 4. Comparing point set upsampling (x16) results from synthetic sparse inputs (a) using different methods (b-e). We also show the associated error maps, where the colors reveal the nearest distance for each target point to the predicted point set generated by each method.

4.5. Ablation Study

To evaluate the effectiveness of the major components in our framework, we conducted an ablation study by simplifying our full pipeline in the following four cases: (A) removing the spatial refiner and only keeping the dense generator (see Figure 2); (B) removing the local refinement unit; (C) removing the global refinement unit; and (D) removing offsets and directly regressing the point coordinates. In each case, we re-trained the network and tested the performance using synthetic data. Table 3 summarizes the results of each case in terms of CD value, compared to our full pipeline (bottom row). Clearly, our full pipeline performs the best with the lowest CD value, and removing any component reduces the overall performance, meaning that each component in our framework contributes. We also present a visual result associated with the ablation study in Figure 6. The supplemental material provides more results with various settings, verifying the effectiveness of our disentangled design. Noted that since the dense generator adopts an identical or simplified design from the previous works [26, 12], our spatial refiner is generally applicable to other networks.

5. Conclusion

In this paper, we present a disentangled refinement framework for point cloud upsampling. Different from existing approaches that try to meet the various upsampling

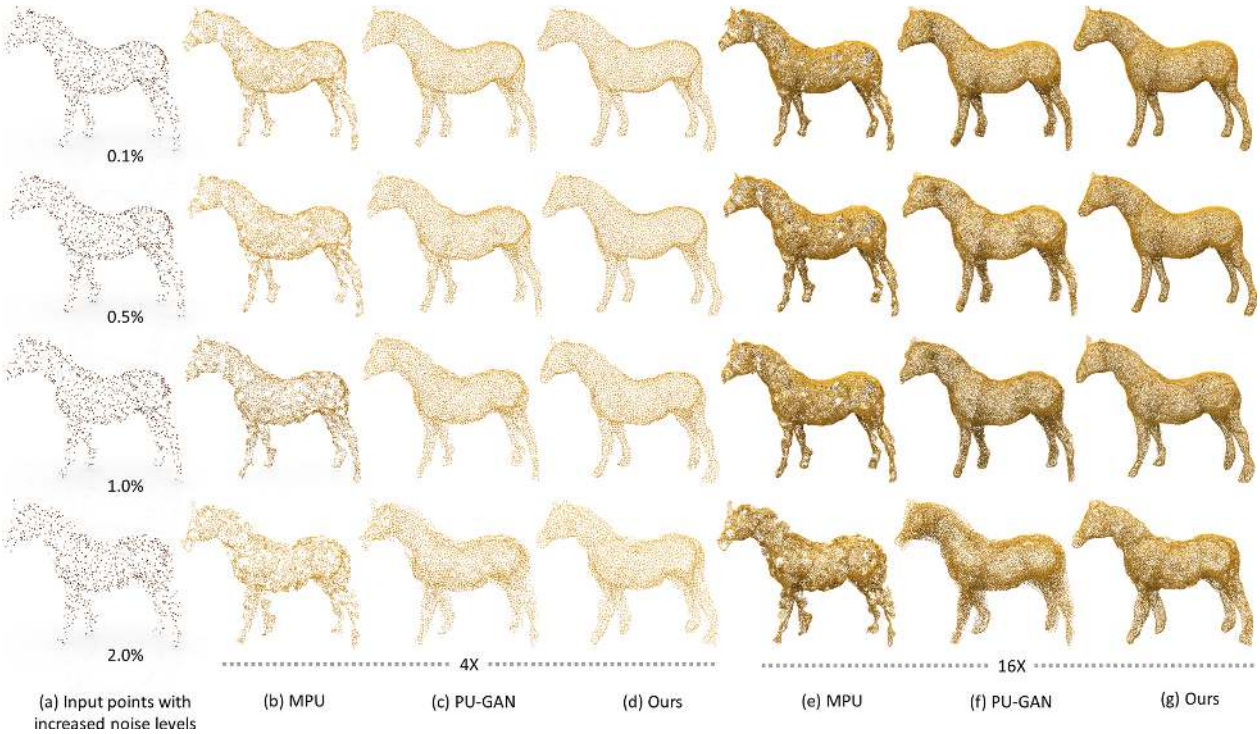

Figure 5. Comparing point set upsampling results produced using different methods under different upsampling rate, when given noisy sparse inputs with increasing noise level, *i.e.*, 0.1%, 0.5%, 1.0%, and 2.0%.

Table 2. Quantitative comparisons by using our method and state-of-the-arts to upsample noisy inputs with increasing noise level ($r = 16$). Here we show CD values with the unit of 10^{-3}.

Methods	Perturbation with different noise levels				
	0%	0.1%	0.5%	1.0%	2.0%
PU-Net [28]	0.699	0.717	0.794	0.860	0.945
MPU [26]	0.348	0.364	0.426	0.524	0.831
PU-GAN [12]	0.269	0.309	0.381	0.562	0.899
Our	**0.199**	**0.213**	**0.229**	**0.310**	**0.592**

Table 3. Comparing the upsampling performance of our full pipeline with various cases in the ablation study (r=16). Here we show CD values with the unit of 10^{-3}.

Model	Spatial Refiner	Local	Global	Offset	CD
A					0.684
B	✓		✓	✓	0.378
C	✓	✓		✓	0.343
D	✓	✓	✓		0.228
Full					**0.199**

goals all in a single network, we propose to disentangle the upsampling task into two sub-goals, where we first generate coarse but dense points, and then refine these points by adjusting the location of each point. To this end, we formulate an end-to-end disentangled refinement framework with two cascaded sub-networks: a dense generator and a spatial refiner. In the spatial refiner, we introduce a pair of local and global refinement units to evolve the coarse feature map by considering both local and global geometric structures. Also, we design our spatial refiner to regress offset vectors to adjust the coarse outputs in fine scale. Experimental results demonstrate the superiority of our method over others.

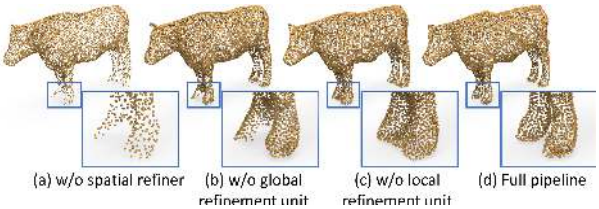

Figure 6. Visualization results for the ablation study.

Actually, this work aims to provide a generic framework to disentangle the point cloud upsampling task. In the future, we may continue to explore a more comprehensive architecture for dense generator and spatial refiner. We may further explore the possibility of designing a region adaptive refiner, meaning that we only fine-tune regions that are non-uniform and noisy, thus improving the overall efficiency. Lastly, designing the refiner to be aware of edges may be helpful for downstream tasks like mesh reconstruction.

Acknowledgments. We thank anonymous reviewers for the valuable comments. This work is supported by the Hong Kong Centre for Logistics Robotics, and Research Grants Council of the Hong Kong Special Administrative Region (Project No. CUHK 14201717 & 14201918 &14201620).

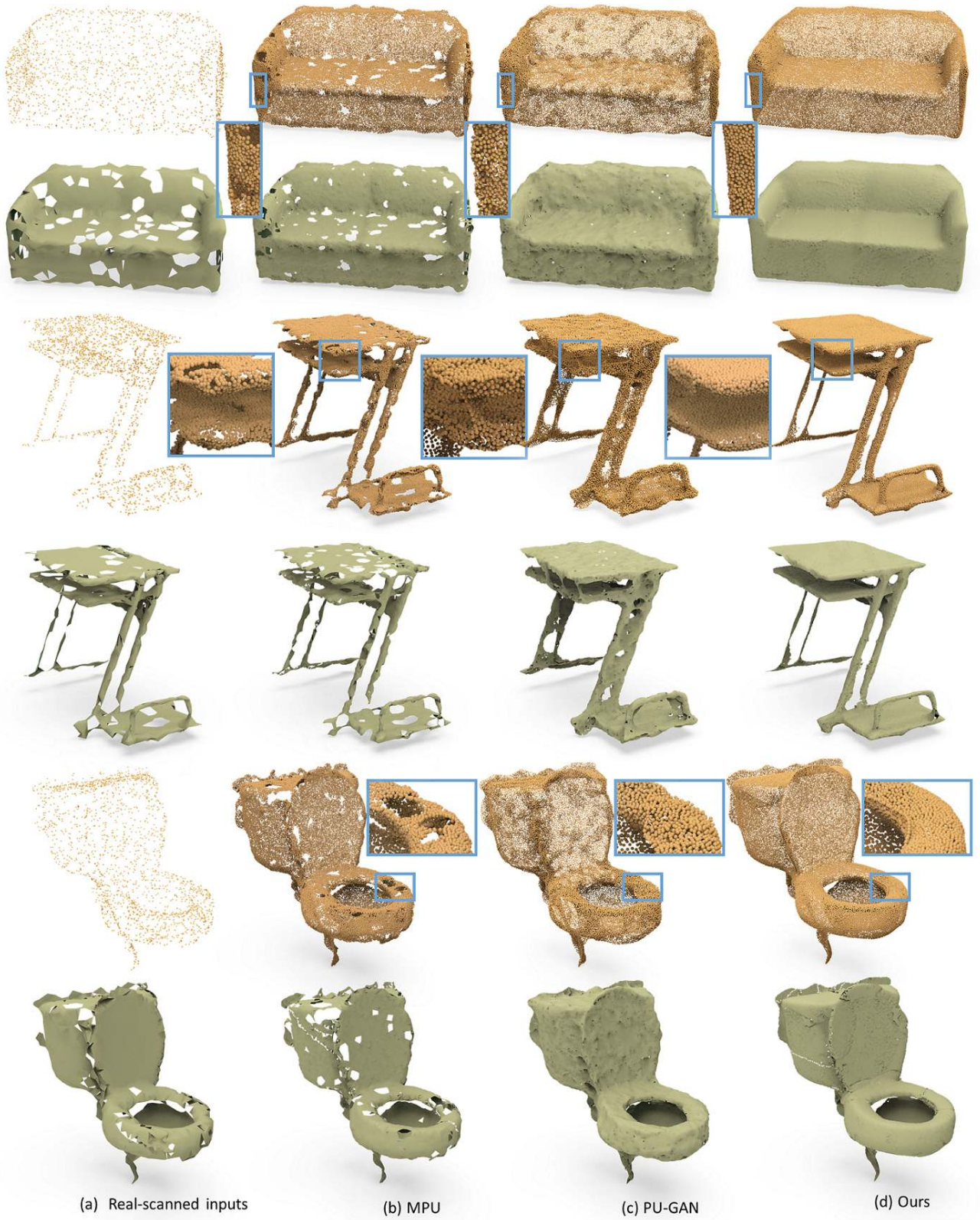

Figure 7. Comparing point set upsampling (16×) results and reconstructed 3D meshes using different methods (b-d) from real-scanned sparse inputs (a), while the bottom row shows the reconstructed meshes. Clearly, our method outperforms others on the local uniformity, contributing a better surface reconstruction.

References

[1] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Trans. Vis. & Comp. Graphics (TVCG)*, 9(1):3–15, 2003. 1, 2

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

[3] Xuelin Chen, Baoquan Chen, and Niloy J. Mitra. Unpaired point cloud completion on real scans using adversarial training. *Int. Conf. on Learning Representations (ICLR)*, 2020. 2

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D object detection network for autonomous driving. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1907–1915, 2017. 1

[5] Zhutian Chen, Wei Zeng, Zhiguang Yang, Lingyun Yu, Chi-Wing Fu, and Huamin Qu. LassoNet: Deep Lasso-selection of 3D point clouds. *IEEE Trans. Vis. & Comp. Graphics (TVCG)*, 26(1):195–204, 2020. 2

[6] David M. Cole and Paul M. Newman. Using Laser range data for 3D SLAM in outdoor environments. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1556–1563, 2006. 1

[7] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3D object reconstruction from a single image. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 605–613, 2017. 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[9] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. Total Denoising: Unsupervised learning of 3D point cloud cleaning. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 52–60, 2019. 2

[10] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM Trans. on Graphics (TOG)*, 32(1):9:1–12, 2013. 1, 2

[11] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 12697–12705, 2019. 2

[12] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-GAN: A point cloud upsampling adversarial network. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 7203–7212, 2019. 1, 2, 3, 4, 5, 6, 7

[13] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. PointAugment: An auto-augmentation framework for point cloud classification. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6378–6387, 2020. 2

[14] Xianzhi Li, Ruihui Li, Guangyong Chen, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. A rotation-invariant framework for deep point cloud analysis. *arXiv preprint arXiv:2003.07238*, 2020. 2

[15] Xianzhi Li, Lequan Yu, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Unsupervised detection of distinctive regions on 3d shapes. *ACM Trans. on Graphics (TOG)*, 39(5):158:1–14, 2020. 2

[16] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on $\mathcal{X}$-transformed points. In *Conference and Workshop on Neural Information Processing Systems (NeurIPS)*, pages 828–838, 2018. 2

[17] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Trans. on Graphics (SIGGRAPH)*, 26(3):22:1–5, 2007. 1, 2

[18] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. Holoportation: Virtual 3D teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 741–754, 2016. 1

[19] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3D object detection in point clouds. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 9277–9286, 2019. 2

[20] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 2

[21] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. PUGeo-Net: A geometry-centric network for 3D point cloud upsampling. In *European Conf. on Computer Vision (ECCV)*, 2020. 1, 2, 5

[22] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J. Mitra, and Maks Ovsjanikov. PointCleanNet: Learning to denoise and remove outliers from dense point clouds. *Computer Graphics Forum (CGF)*, 39(1):185–203, 2020. 2

[23] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1588–1597, 2019. 1, 4

[24] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 3523–3532, 2019. 2

[25] Shihao Wu, Hui Huang, Minglun Gong, Matthias Zwicker, and Daniel Cohen-Or. Deep points consolidation. *ACM Trans. on Graphics (SIGGRAPH Asia)*, 34(6):176:1–13, 2015. 2

[26] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3D point set upsampling. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5958–5967, 2019. 1, 2, 3, 4, 5, 6, 7

[27] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. EC-Net: An edge-aware point set consolidation network. In *European Conf. on Computer Vision (ECCV)*, pages 386–402, 2018. 1, 2

[28] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net: Point cloud upsampling network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2799, 2018. 1, 2, 3, 4, 5, 7

[29] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point completion network. In *Int. Conf. on 3D Vision (3DV)*, pages 728–737, 2018. 2

[30] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *Int. Conf. on Machine Learning (ICML)*, pages 7354–7363. PMLR, 2019. 4

[31] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1607–1616, 2019. 2