Parametric Surface Constrained Upsampler Network for Point Cloud

Pingping Cai, Zhenyao Wu, Xinyi Wu, Song Wang

University of South Carolina, USA
{pcai,zhenyao,xinyiw}@email.sc.edu, songwang@cec.sc.edu

Abstract

Designing a point cloud upsampler, which aims to generate a clean and dense point cloud given a sparse point representation, is a fundamental and challenging problem in computer vision. A line of attempts achieves this goal by establishing a point-to-point mapping function via deep neural networks. However, these approaches are prone to produce outlier points due to the lack of explicit surface-level constraints. To solve this problem, we introduce a novel surface regularizer into the upsampler network by forcing the neural network to learn the underlying parametric surface represented by bicubic functions and rotation functions, where the new generated points are then constrained on the underlying surface. These designs are integrated into two different networks for two tasks that take advantages of upsampling layers – point cloud upsampling and point cloud completion for evaluation. The state-of-the-art experimental results on both tasks demonstrate the effectiveness of the proposed method. The implementation code will be available at https://github.com/corecai163/PSCU.

Introduction

Point cloud is an efficient data structure to represent 3D objects. But, due to the limitation of sensors, the collected point clouds are usually sparse and incomplete. Therefore, point cloud upsampling (Li et al. 2019; Qian et al. 2021; Yu et al. 2018; Qian et al. 2020; Wang et al. 2019) is introduced to generate denser point clouds for better scene representation, which benefits many computer vision applications such as autonomous driving (Zeng et al. 2018; Li et al. 2021), 3D object classification (Li et al. 2020; Qi et al. 2017b), semantic segmentation (Zhang et al. 2020; Landrieu and Simonovsky 2018; Engelmann et al. 2017) and robotics (Rusu et al. 2008). For point cloud upsampling, it is expected that the generated dense points can well represent the shape and surface underlying the point cloud. However, obtaining such property is challenging, and even previous state-of-the-art (SOTA) methods may generate many noisy and outlier points (see Figure 1).

Many sophisticated methods have been proposed to solve this challenging problem. Traditional optimization based methods (Alexa et al. 2003; Lipman et al. 2007; Huang et al.
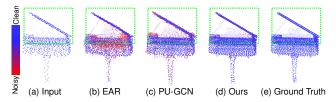
Figure 1: The double-layer lid can be clearly upsampled by our method, while the traditional EAR (Huang et al. 2013) fails to distinguish the two nearby layers and the previous SOTA method PU-GCN (Qian et al. 2021) generates many noisy and outlier points. Please zoom in for more details.

2013; Wu et al. 2015; Preiner et al. 2014) rely on geometric priors to upsample the point cloud but often fail when the input is complex and have been outperformed by recent deep learning methods. These well-designed deep learning methods can be further divided into two categories based on their adopted upsampler: 1) feature-based upsampling methods and 2) folding-based upsampling methods. The feature-based methods (Zhang, Yan, and Xiao 2020; Huang et al. 2020; Wu, Zhang, and Huang 2019; Ye et al. 2022; Qian et al. 2021; Yu et al. 2018; Xiang et al. 2021) first extract shape features from input points and expand the number of points by upscaling the shape features in the feature space. These upsampled features are then fed into a coordinate regression block (MLPs) to predict their coordinates. A point-wise loss function, *i.e.,* Chamfer Distance (Fan, Su, and Guibas 2017), is usually used to train the network. However, this loss function only measures the point-wise distance and can not measure the difference of underlying surfaces between point clouds. As a result, these methods often fail to generate points that are located accurately on the underlying surfaces. Folding-based methods (Yang et al. 2018; Yuan et al. 2018; Liu et al. 2020, 2022; Long et al. 2021; Luo and Hu 2020) expand the number of points by introducing a predefined 2D grid for each point and then concatenating them with shape features to regress for their coordinates via MLPs – they can be viewed as a mimic of a morphing/surface function that transforms the 2D grid to target surfaces. While they attempt to preserve a better surface structure, they can only learn an overfitted point-point mapping with point-wise loss functions (Williams et al. 2019), but not the accurate

representation of the surface.

To achieve a better surface representation, PUGeo-Net (Qian et al. 2020) introduced a parameterized method that incorporates discrete differential geometry into network design, where it models the small surface around each input point via the first and second fundamental forms (do Carmo 1976) to generate the upsampled points. However, this method relies heavily on the correctness of point normals and needs additional ground truth point normals to train the network. Such normals are not directly available in many point cloud datasets. Besides, it still uses MLPs to predict the point displacement and lacks explicit surface-level constraints. Also, it follows the patch-based upsampling pipeline, which does not consider the smoothness between two adjacent patches.

We extend PUGeo-Net by using explicit parametric functions to model the local surface for each input point without the required supervision of point normal information. Besides, instead of dividing the input point cloud into multiple patches, we directly upsample the entire input to avoid the discrepancy between adjacent patches. Specifically, we design a novel parametric surface constrained upsampler network that can extract the spatial surface features from discrete and unordered input points, predict explicit bicubic functions and rotation functions to express the underlying surfaces, and constrain the new generated points on these surfaces. To further improve its performance, we also introduce a displacement loss for generating better parametric functions. The proposed upsampler can be also used for other related tasks such as point cloud completion.

We evaluate our proposed method in both point cloud upsampling and completion tasks on three standard datasets, PU1K, KITTI, and ShapeNet-PCN. The experiment results demonstrate that by using the proposed surface constrained upsampler, we can achieve new SOTA results by outperforming previous methods. The main contributions of this paper are as follows.

1. We propose a novel surface-level constraint that uses parametric surfaces as regularizers to ensure the smoothness of the upsampled point clouds.

2. We design a new parametric surface constrained upsampler, that estimates the surface parametric functions from input points and generates new points on each surface.

3. We evaluate the proposed upsampler on both point cloud upsampling and point cloud completion tasks, and our proposed method achieves new SOTA performance.

Related Work

Point Cloud Upsampling

Yu et al. (2018) proposed the first deep learning based point cloud upsampling algorithm PU-Net at patch level. For each patch, multilevel features are extracted at each input point and expanded via a multibranch convolution unit in the feature space. Then, these expanded features are reconstructed for upsampled point cloud via an MLP based coordinate regression block. However, when the upsampling rate is high, *e.g.* $\times 16$, it needs to define multiple branches of convolution layers, which is quite inefficient. To address this issue, Wang et al. (2019) proposed 3PU, a multistep patch-based method to progressively upsample the points by breaking the upsampling task into multiple $\times 2$ small tasks, each solved by a sub-network that focuses only on a particular level of detail. Each sub-network has the same structure with the feature extractor unit, feature expansion unit, and an MLP to regress for the point coordinates. To consider more spatial information among neighboring points, Qian et al. (2021) proposed PU-GCN by introducing a multi-scale Inception DenseGCN feature extractor to extract the spatial features among nearby points and another graph convolution network to better encode local point information from its neighbors. Although PU-GCN achieves the SOTA result by utilizing spatial information, it relies on an MLP-based coordinate regression module which often fails to learn the accurate representation of the surface.

Point Cloud Completion

Point cloud completion can be considered as a more challenging version of upsampling where the input point cloud is sparse and incomplete. It aims to not only upsample the input point cloud but also infer the missing underlying shapes and surfaces (Pan et al. 2021; Yuan et al. 2018; Xie et al. 2020; Xiang et al. 2021; Wen et al. 2021; Cai and Sur 2022).

Yuan et al. (2018) proposed PCN, a novel coarse to fine point cloud completion network under an encoder-decoder framework. It first extracts the global shape code that describes the coarse shape of the input points using a PointNet (Qi et al. 2017a) feature extractor. Then, a coarse but complete point cloud is generated from the global shape code and fed into a folding-based upsampling block to generate the dense point cloud. However, PCN's feature extractor can not extract sufficient structural information from input point clouds and may not well capture the geometric detail. To this end, Xie et al. (2020) proposed GRNet, using 3D grids as intermediate representations to regularize unordered point clouds and 3D convolution networks to explore the structural context and infer the underlying shape of the input point clouds. However, the resolution of intermediate point clouds is limited by the size of 3D feature maps, making it hard to reveal fine local geometric details. As a result, it still needs additional MLPs to refine and upsample the point clouds. To preserve more local geometric details, Xiang et al. (2021) designed SnowFlakeNet. It first extracts the global shape code from input point clouds using Point-Net++ (Qi et al. 2017b) and Point Transformer (Zhao et al. 2021) blocks, which are fed into a seed generator to generate a course but complete point cloud. The coarse point cloud is then upsampled to a denser one via SnowFlake Point Deconvolution blocks. These blocks consist of 1D deconvolution layers to upsample the points in the feature space and skip-transformers to preserve better structure and shape. Although SnowFlakeNet achieves the state-of-the-art result on point cloud completion tasks, it still relies on an MLP to regress for the coordinates of each upsampled point without constraining the upsampled points to be on the underlying surfaces accurately.
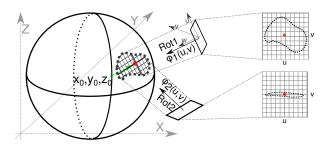
Figure 2: An example to describe the surface of a small region by the parametric functions. There exist multiple projections and surface functions to express this small region.

Parametric Surface Constrained Upsampler

Designing an upsampler for point clouds is nontrivial due to the difficulty of inferring accurate underlying surfaces from discrete points and elegantly constraining the upsampled points on these surfaces. We solve these challenges by forcing the network to predict an explicit representation of the underlying surface via parametric surface functions and then generate points directly on the surface via the predicted parametric functions.

Parametric Surfaces

Ideally, given points on the 2D plane X-Y, we can map them into 3D space via an explicit function $\Phi(x, y) \to (x, y, z)$. However, if the target surface is perpendicular to the X-Y plane, *i.e.*, $X = 0$, we cannot calculate the coordinates z for $x \neq 0$, limiting its representation ability. To solve this problem, we use local coordinate systems with rotations to improve the representation ability with better projection planes. Figure 2 shows an example of local parametric surfaces around the coordinate (x_0, y_0, z_0). In particular, we define the parametric surface function as $Rot(\phi(u, v)) + (x_0, y_0, z_0)$, where $\phi(u, v) \to (u, v, w)$ is the surface function that maps points in 2D projection planes to 3D surfaces on a local coordinate system U-V-W, and $Rot(u, v, w) \to (x, y, z)$ is the rotation function with the rotation center $(u = 0, v = 0, w = 0)$ to rotate local coordinates into the global X-Y-Z coordinate system. Mathematically, given these two functions, we can generate an arbitrary number of points on the local surface. Thus, we introduce this idea into the design of the surface constrained upsampler network, *i.e.*, we propose to learn and utilize the surface function ϕ and the rotation function Rot to constrain the upsampled points.

Explicit Surface Function Previous methods used the concatenation-based folding technique to model the surface function via MLPs (Yang et al. 2018; Yuan et al. 2018; Qian et al. 2020; Luo and Hu 2020). However, as mentioned in the introduction, such methods only learn an overfitted point-point mapping and induce a bottleneck that limits its capacity to represent different 3D surfaces. Different from these methods, we propose to use the explicit polynomial function to model the underlying surface, which can be expressed as:

$$w = \phi(u, v) = \sum_i \sum_j a_{ij} u^i v^j, \quad (1)$$

where a_{ij} is the coefficient predicted via neural networks. With different combinations of coefficients, we can easily express different shapes. For simplicity, we use the bicubic function, which is widely used and strong enough to express common shapes, to design our network:

$$w = \phi(u, v) = a_1 + a_2 u + a_3 u^2 + ... + a_{16} u^3 v^3, \quad (2)$$

Rotation Function After generating upsampled points on the local surface, we need to rotate them from their local coordinate systems into the global coordinate system. To achieve this, we model the rotation function as follows:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = Rot(u, v, w) = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \quad (3)$$

where $[r_1, ..., r_9]$ are the elements of a rotation matrix R predicted by neural networks. To ensure that the rotation matrix follows the correct principle of $R^T * R == R * R^T == I$, we use the 6d representation proposed in (Zhou et al. 2019), which shows a good continuous property and can be decoded into a 3×3 matrix.

Network Design

However, designing a special network that can predict accurate surface parameters and seamlessly integrate them into the upsampling procedure is challenging. For convenience, we refer to the input points as parent points and the upsampled points as child points. Our idea is that each parent point will split and generate multiple child points that lie on the local surface and cover the entire surface as much as possible. To achieve this, we design the parametric surface constrained upsampler network that contains 3 major parts: (i) Spatial Feature Extractor, (ii) Surface Parameter Estimation, and (iii) Child Points Generator, aiming at extracting the local geometric shape from unordered parent points, predicting the explicit surface parameters around each parent point, and generating child points on parametric surfaces, respectively. Each of them will be described in detail in the following subsections. Figure 3 shows the general structure of the proposed upsampling network, and note that the detailed network structure is provided in the Supplementary. The proposed upsampler module requires three necessary inputs: parent coordinates $P_i \in \mathbf{R}^{N \times 3}$, parent features $F_i \in \mathbf{R}^{N \times C_1}$, and the global shape code $G \in \mathbf{R}^{1 \times C_2}$ that encodes the global shape of the input point cloud. It aims to upsample them m times and generates child points with coordinates $P_{i+1} \in \mathbf{R}^{mN \times 3}$ and features $F_{i+1} \in \mathbf{R}^{mN \times C_1}$.

i. Spatial Feature Extractor Making the network aware of the local geometric information around each parent point is a key step in predicting accurate shapes. To achieve this, we design a Spatial Feature Extractor (SFE) that can aggregate the positions and features of the parent's K nearby points to extract the local spatial information. Especially, the SFE extracts the local spatial features $SF_i \in \mathbf{R}^{N \times C_1}$ from input parent features F_i, parent positions P_i, and global shape code G, which is defined as

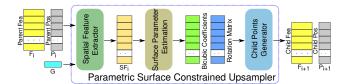$$SF_i = \texttt{Aggregate}_K(P_i, \texttt{NN}_s(F_i, G)), \quad (4)$$

Figure 3: The general structure of the parametric surface constrained upsampler. Note that due to the page limitation, the detailed network structure is in the Supplementary.
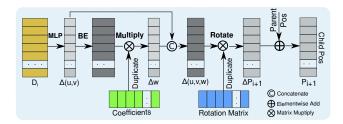

Figure 4: The network architecture for generating child points on surface via predicted surface parameters.

where Aggregate_K is the point transformer introduced in (Zhao et al. 2021) to aggregate the context of K nearest points, and NN_s is the neural network used to combine the parent features F_i with global shape code G.

ii. Surface Parameter Estimation Then, to accurately represent the underlying surface around each parent point, we propose to estimate the explicit parameters of the surface function ϕ and the rotation function Rot as mentioned before. One simple way is to directly predict their parameters from local spatial features SF_i. However, SF_i only contains the local shape information, and the global smoothness of these local shapes is not guaranteed. Thus, we incorporate the global shape code G to smooth them. Especially, these parameters can be predicted via:

$$a = \text{NN}_a(G, SF_i), \tag{5}$$

$$r = \text{NN}_r(G, SF_i), \tag{6}$$

where NN_a and NN_r are neural networks and a and r are predicted coefficients of the bicubic function and the rotation matrix.

iii. Generating Child Points on Surface Finally, our objective is to generate the child features F_{i+1} and the child positions P_{i+1} on the predicted parametric surface. Unlike previous methods (Xiang et al. 2021; Qian et al. 2021; Yu et al. 2018), where they reconstruct the 3D child coordinates directly from the child features through MLPs, we design a network that smoothly integrates the predicted surface parameters into child point generation. Especially, we first predict the displacement of the child $(\Delta u, \Delta v)$ in the projection plane and then lift them into 3D spaces using predicted parametric functions.

To implement this, we first generate the relative displacement of the child features $D_i \in R^{mN \times C_1}$ *w.r.t* to their parents' through the 1D deconvolution layer, which can easily generate different numbers of child features by setting

different kernel sizes and strides. The displacement feature D_i is used to predict the coordinate displacement of the child $(\Delta u, \Delta v)$ using an MLP. Specifically, $(\Delta u, \Delta v) = \text{MLP}(D_i)$. Next, based on Equation (2), we can calculate their embedded values $[1, \Delta u, \Delta u^2, ..., \Delta u^3 \Delta v^3]$ via a Bicubic Embedding (BE) block and multiply them with predicted bicubic coefficients to generate the coordinate displacement Δw. We then transit them into X-Y-Z coordinate system via the predicted rotation matrix and get the child displacements $\Delta P_{i+1} = (\Delta x, \Delta y, \Delta z)$, which will be added with their parent positions P_i to obtain the final position of the child points P_{i+1}. Figure 4 shows the corresponding network architecture to generate the child position in the parametric surface. After obtaining the position of the child points, we feed D_i into another MLP layer and add the output with their parent feature to get the child features F_{i+1}.

Loss Function

To train our network, we first use Chamfer Distance as a loss function for each upsampling block, which measures the point-wise distance between the predicted point cloud and ground truth. However, we notice that given a small area of parametric surface there exist multiple projection planes with different surface functions and rotation functions. Not all of them can describe this small surface correctly and efficiently. For example, in Figure 2, if the projection plane is perpendicular to the surface (parallel to the surface normal), it is difficult to find a good surface parametric function.

Ideally, we aim to select a projection plane that is perpendicular to the normal of each surface/parent point. As our network does not have ground truth normal information, we borrow the idea from the unsupervised principal component analysis algorithm (Jolliffe, Ian 2014) to select a better projection plane, where it aims to find a projection plane that can maximize the covariance matrix of $(\Delta u, \Delta v)$, *a.k.a*, minimize the covariance matrix of Δw given 3D points in a small region. Thus, inspired by this we add a constraint to our network by introducing the displacement loss as follows:

$$\mathcal{L}_d = ||\Delta w||_2^2. \tag{7}$$

In summary, our final loss function is defined as $\mathcal{L} = \mathcal{L}_{CD} + \lambda \mathcal{L}_d$, where λ is a hyperparameter that balances the weight of the chamfer loss and the displacement loss.

Experiments

To validate the effectiveness of the proposed upsampler, we first evaluate it on the PU1K dataset and conduct ablation studies to verify the effectiveness of our network. We also apply our method to the real collected LiDAR point cloud KITTI dataset (Geiger et al. 2013). Then, we further present the capability of the proposed method on a more challenging point cloud completion task on the ShapeNet-PCN dataset.

Point Cloud Upsampling

PU1K: The PU1K dataset is first introduced in PU-GCN (Qian et al. 2021) for point cloud upsampling. It consists of 1,147 3D models, which are collected from PU-GAN (Li et al. 2019) and ShapeNet dataset (Chang et al. 2015) to
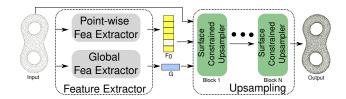
Figure 5: Point cloud upsampling with surface constrained upsampler. We can stack multiple upsampling blocks to achieve a higher upsampling ratio.

Method	CD ($\times 10^{-4}$)	P2F ($\times 10^{-3}$)	Uniformity $\times (10^{-3})$			
			0.4%	0.6%	0.8%	1.0%
EAR	1.449	3.314	1.82	3.68	6.51	9.92
PU-Net	1.751	4.847	2.07	4.24	7.54	11.78
3PU	1.461	3.559	1.99	4.12	7.23	11.04
PU-GCN	1.151	2.504	1.95	3.97	6.83	10.63
Ours	**0.886**	**1.091**	**1.40**	**2.85**	**5.06**	**7.95**

Table 1: Quantitative upsampling results compared to previous SOTA algorithms. The uniformity score is estimated in the local area of different percentages of radii.

Components	CD ($\times 10^{-4}$)
w/o Spatial Feature	1.293
w/o Parametric Function	0.967
w/o Displacement Loss	0.893
Full	**0.886**

Table 2: Ablation studies of the Parametric Surface Constrained Upsampler on the PU1K dataset.

cover various shape complexities and diversity. The input point cloud is sparse but complete with 2,048 points, and the ground truth point cloud is 4 times denser with 8,192 points. We follow the same train/test splitting strategy in PU-GCN with 1,020 training samples and 127 testing samples. Note that unlike previous patch-based methods (Qian et al. 2021; Ye et al. 2022; Yu et al. 2018; Qiu, Anwar, and Barnes 2021), our training data are entire point clouds generated from ground truth meshes by poisson disk sampling (Bridson 2007). For testing, we directly use the test data given by PU-GCN for a fair comparison.

Network Structure: Figure 5 shows the network architecture for the point cloud upsampling task. To generate the required inputs for the proposed upsampler, we use a feature extractor to capture both point-wise features and global features from sparse points. Especially, it consists of two parts: a point-wise feature extractor, which is an MLP that maps input points into the feature space, and a global feature extractor, which consists of a PointNet++ backbone (Qi et al. 2017b) with point transformers (Zhao et al. 2021) to incorporate both the local and global shape contexts. The outputs of the feature extractor block are the point-wise features F_0 and the global shape code G. Note that the design of feature extractor is not the major contribution of this paper and we can exploit any other suitable networks. We then feed these outputs along with the original position of the points P_0 into stacks of upsampling blocks to generate denser point clouds. To upsample the point cloud 4 times, we arbitrarily set two upsampler blocks with upscale ratios of 1 and 4, respectively. Note that other combinations of upscale ratios and the number of upsampler blocks are also feasible.

Evaluation Metrics: We use three widely adopted metrics in previous work to evaluate our performance: Chamfer Distance (CD), Point-to-Surface Distance (P2F) *w.r.t* ground truth meshes, and Uniformity Score (Li et al. 2019). For these metrics, a smaller value means better performance.

Training Detail: To train this network, we use 2 Tesla V100 GPUs. We set the batch size to 16 and the total epoch number to 150. Besides, we use Adam as the optimization function with a learning rate of 0.0005 at the beginning, and we decrease the learning rate by a factor of 0.5 every 50 epochs.

Experiment Results: Table 1 shows the quantitative upsampling results on the PU1K dataset. We find that our algorithm achieves the best performance over all its counterparts with large improvements. In particular, compared to the previous SOTA algorithm, PU-GCN, the proposed algorithm

reduces the average CD from 1.151×10^{-4} to 0.886×10^{-4}. Besides, the average P2F also reduces more than half from 2.504×10^{-3} to 1.091×10^{-3}, which statically proves that our generated child points preserve better surface shapes and locate closer to the ground truth surfaces. What's more, the proposed method also obtains better uniformity scores than previous SOTA algorithms. Next, to intuitively show the performance, we visually check the upsampling outputs of our method and compare them with the outputs of other algorithms. Figure 6 shows the visual results of different algorithms. We see that our proposed method can produce point clouds with much better shape quality and fewer off-the-surface points. Apparently, both the quantitative and visual results prove the superiority of the proposed network.

Ablation Study

We then perform ablation studies to figure out which part of the proposed upsampling network contributes the most to its performance. Table 2 summarizes all experiment results.

Spatial Feature: Intuitively, the local spatial information is crucial to predict accurate local surfaces. Thus, we first examine its importance by removing the point transformer in the Spatial Feature Extractor, which is designed to aggregate features of K nearest neighbors for each point. In Table 2, we see that after removing the point transformer the performance drops to 1.239×10^{-4} with a huge gap, which justifies our intuition.

Parametric Function: As we explicitly model the parametric surface via the bicubic function and the rotation function, one might ask about the effectiveness of this explicit representation compared to folding-based MLPs (Yuan et al. 2018; Liu et al. 2020; Yang et al. 2018; Wen et al. 2020). To this end, we substitute the explicit surface function with an MLP-based folding layer that takes the displacement of the child points $(\Delta u, \Delta v)$ as inputs and outputs its global displacement $(\Delta x, \Delta y, \Delta z)$. In Table 2, we see that after using MLPs to model surface functions, performance decreases
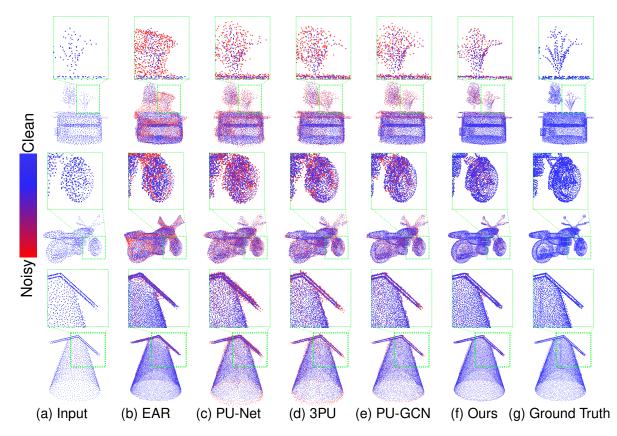
Figure 6: Visualization of upsampling results with different algorithms (EAR, PU-Net, 3PU, PUGCN, and Ours). We see that our method produces the best results, generating smooth borders and preserving fine-grained local details.

Noise	0%	0.5%	1%	1.5%
CD ($\times 10^{-4}$)	0.886	1.053	1.280	1.599

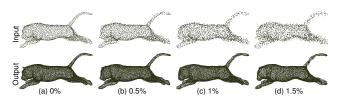Table 3: Performance under different noise perturbations.

Figure 7: Visualization of upsampling results with different levels of noise. Our method still can preserve good underlying shapes with noisy inputs.

and CD increases from 0.886×10^{-4} to 0.967×10^{-4} with a gap of 0.081×10^{-4}. This gap illustrates the superiority of parametric surface functions in representing better underlying surfaces compared to MLPs.

Displacement Loss: Since we introduce an additional displacement loss to select a better projection plane, we then illustrate its effect on training the proposed upsampler by removing this loss. We see that without displacement loss, the performance decreases slightly from 0.886×10^{-4} to 0.893×10^{-4}. This slight performance drop fits our intuition because there exist multiple choices for the projection planes, and it contributes the least to the performance of the proposed upsampler.

Robustness to Noise Another concern might be the performance of the proposed method with noisy inputs. Therefore, we add a small perturbation to each input point with a Gaussian distribution to synthesize noise, retrain and test the robustness of our method at different noise levels. Table 3 and Figure 7 show the quantitative and visual results at different levels of perturbations. Intuitively with noise points,

it becomes more difficult to infer the accurate surfaces. We see that under a small perturbation such as 0.5%, our method still achieves a promising result. Even with a 1.5% perturbation, the upsampled points still tend to preserve a smooth shape with little distortion.

Result on Real World Data Finally, we further show the performance of the proposed method on the real collected point cloud in the KITTI dataset (Geiger et al. 2013). Figure 8 shows an example of the upsampling results. Due to the hardware limitation of the LiDAR sensor, the collected point cloud is naturally sparse and non-uniformly distributed, making the upsampling more challenging. Our method can generate dense point clouds with better distributions.
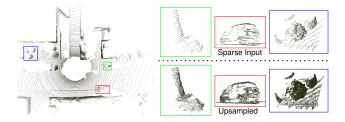
Figure 8: 4 times upsampling result on a real-world point cloud from KITTI dataset.
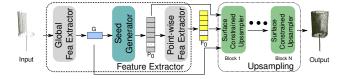

Figure 9: Network architecture for point cloud completion.

Point Cloud Completion Task

Then, we test the proposed upsampler on the point cloud completion task using the ShapeNet-PCN dataset.

ShapeNet-PCN: The ShapeNet-PCN dataset is introduced by (Yuan et al. 2018), which is derived from ShapeNet (Chang et al. 2015). It contains pairs of partial and complete point clouds from 30,974 models of 8 categories in total: airplane, cabinet, car, chair, lamp, sofa, table, and watercraft. The complete point clouds are created by sampling 16,384 points uniformly from the original meshes, and the partial point clouds are generated by back-projecting 2.5D depth images into 3D. For each ground truth, 8 partial point clouds are generated from 8 randomly distributed viewpoints. For fairness, we follow the same train/test splitting strategy in (Yuan et al. 2018; Xie et al. 2020; Xiang et al. 2021) with 29,774 training samples and 1,200 testing samples, and resample each incomplete cloud to 2,048 points.

Network Structure: As the point cloud completion task is more challenging, where the input is a sparse and incomplete point cloud and the output is a dense and complete point cloud, we use a new network to generate complete and denser point clouds. Figure 9 shows the detailed network architecture for the point cloud completion task. Because the input is incomplete, generating a coarse but complete point cloud is crucial for subsequent upsampling steps. Inspired by the previous SOTA algorithm SnowFlakeNet (Xiang et al. 2021), we adopted the seed generator used in their network to generate a sparse but complete point cloud, then feed them into three consecutive surface constrained upsamplers with upscale ratios of 1, 4, and 8, respectively, to generate high resolution point clouds.

Evaluation Metrics: For a fair comparison with previous methods, we use two commonly used metrics: L1 Chamfer Distance (L1-CD) and Earth Mover's Distance (EMD). Similarly, the smaller the metric, the better the performance.

Training Detail: We use 4 Tesla V100 GPUs with a batch size of 32 and a total epoch number of 500. Similar to SnowFlakeNet, we use Adam as the optimization function

Methods	L1-CD ($\times 10^{-3}$)	EMD ($\times 10^{-3}$)
PCN	9.64	87.14
GR-Net	8.83	**55.26**
PMP	8.73	109.67
SnowFlake	7.19	69.13
Ours	**7.04**	66.57

Table 4: Quantitative completion results compared to previous SOTA algorithms on the ShapeNet-PCN dataset.
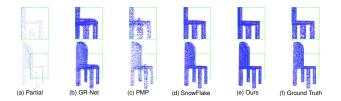

Figure 10: Visualization of completion results with different algorithms (GR-Net, PMP, SnowFlake-Net, and Ours).

with warm-up settings, where it first takes 200 steps to warm up the learning rate from 0 to 0.0005, and then the learning rate decays by a factor of 0.5 for every 50 epochs.

Experiment Results Table 4 shows the quantitative completion results on the ShapeNet-PCN dataset. We notice that our method still achieves the best performance in terms of L1-CD. As we use the same backbone and upscale settings as SnowflakeNet, which is the previous SOTA algorithm, the improvement over SnowflakeNet can directly prove the effectiveness of our proposed upsampling blocks. Compared to SnowflakeNet, we see that our network reduces the average L1-CD from 7.19×10^{-3} to 7.04×10^{-3} and the average EMD from 69.13×10^{-3} to 66.57×10^{-3}. Figure 10 shows one completion result. Note that more completion results can be found in the Supplementary. Still, we see that our method produces a much better shape quality and fewer outlier points.

Note that our upsampler is designed based on the assumption that the input points are well-distributed. But for the completion task, this assumption is not met. Even under this challenging condition, the proposed upsampler still works and the generated points are still well-constrained to the underlying surface.

Conclusions

In this paper, we propose a novel parametric surface constrained upsampler for point clouds. By introducing explicit parametric surface functions into the network design, we can obtain better shape representation ability compared to MLPs with point-wise loss and generate point clouds with smoother shapes and fewer outliers. In addition, the proposed upsampler can also be used in point completion tasks. The experiment results on both point cloud upsampling and completion tasks prove the effectiveness of our method.

Acknowledgements

References

Alexa, M.; Behr, J.; Cohen-Or, D.; Fleishman, S.; Levin, D.; and Silva, C. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1).

Bridson, R. 2007. Fast Poisson Disk Sampling in Arbitrary Dimensions. In *ACM SIGGRAPH 2007 Sketches*.

Cai, P.; and Sur, S. 2022. DeepPCD: Enabling AutoCompletion of Indoor Point Clouds with Deep Learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(2).

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; and Yu, F. 2015. ShapeNet: An Information-Rich 3D Model Repository. arXiv:1512.03012.

do Carmo, M. 1976. *Differential geometry of curves and surfaces.* Prentice Hall. ISBN 978-0-13-212589-5.

Engelmann, F.; Kontogianni, T.; Hermans, A.; and Leibe, B. 2017. Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds. In *IEEE International Conference on Computer Vision (ICCV) Workshops*.

Fan, H.; Su, H.; and Guibas, L. J. 2017. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 32(11): 1231–1237.

Huang, H.; Wu, S.; Gong, M.; Cohen-Or, D.; Ascher, U.; and Zhang, H. R. 2013. Edge-Aware Point Set Resampling. *ACM Transactions on Graphics*, 32(1).

Huang, Z.; Yu, Y.; Xu, J.; Ni, F.; and Le, X. 2020. PF-Net: Point Fractal Network for 3D Point Cloud Completion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jolliffe, Ian. 2014. *Principal Component Analysis*. John Wiley & Sons, Ltd.

Landrieu, L.; and Simonovsky, M. 2018. Large-Scale Point Cloud Semantic Segmentation With Superpoint Graphs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, R.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2019. PU-GAN: a Point Cloud Upsampling Adversarial Network. In *IEEE International Conference on Computer Vision (ICCV)*.

Li, R.; Li, X.; Heng, P.-A.; and Fu, C.-W. 2020. PointAugment: An Auto-Augmentation Framework for Point Cloud Classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, Y.; Ma, L.; Zhong, Z.; Liu, F.; Chapman, M. A.; Cao, D.; and Li, J. 2021. Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8): 3412–3432.

Lipman, Y.; Cohen-Or, D.; Levin, D.; and Tal-Ezer, H. 2007. Parameterization-Free Projection for Geometry Reconstruction. *ACM Transactions on Graphics*, 26(3).

Liu, M.; Sheng, L.; Yang, S.; Shao, J.; and Hu, S.-M. 2020. Morphing and sampling network for dense point cloud completion. In *AAAI conference on artificial intelligence*.

Liu, X.; Liu, X.; Liu, Y.-S.; and Han, Z. 2022. SPU-Net: Self-Supervised Point Cloud Upsampling by Coarse-to-Fine Reconstruction with Self-Projection Optimization. *IEEE Transactions on Image Processing*, 31: 4213–4226.

Long, C.; Zhang, W.; Li, R.; Wang, H.; Dong, Z.; and Yang, B. 2021. PC2-PU: Patch Correlation and Position Correction for Effective Point Cloud Upsampling. arXiv:2109.09337.

Luo, S.; and Hu, W. 2020. Differentiable Manifold Reconstruction for Point Cloud Denoising. In *ACM International Conference on Multimedia*.

Pan, L.; Chen, X.; Cai, Z.; Zhang, J.; Zhao, H.; Yi, S.; and Liu, Z. 2021. Variational Relational Point Completion Network. arXiv:2104.10154.

Preiner, R.; Mattausch, O.; Arikan, M.; Pajarola, R.; and Wimmer, M. 2014. Continuous Projection for Fast L1 Reconstruction. *ACM Transactions on Graphics*, 33(4).

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*.

Qian, G.; Abualshour, A.; Li, G.; Thabet, A.; and Ghanem, B. 2021. PU-GCN: Point Cloud Upsampling Using Graph Convolutional Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Qian, Y.; Hou, J.; Kwong, S.; and He, Y. 2020. PUGeo-Net: A Geometry-Centric Network for 3D Point Cloud Upsampling. In *European Conference on Computer Vision (ECCV)*.

Qiu, S.; Anwar, S.; and Barnes, N. 2021. PU-Transformer: Point Cloud Upsampling Transformer. arXiv:2111.12242.

Rusu, R. B.; Marton, Z. C.; Blodow, N.; Dolha, M.; and Beetz, M. 2008. Towards 3D point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11): 927–941.

Wang, Y.; Wu, S.; Huang, H.; Cohen-Or, D.; and Sorkine-Hornung, O. 2019. Patch-Based Progressive 3D Point Set Upsampling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wen, X.; Li, T.; Han, Z.; and Liu, Y.-S. 2020. Point Cloud Completion by Skip-Attention Network With Hierarchical Folding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wen, X.; Xiang, P.; Han, Z.; Cao, Y.-P.; Wan, P.; Zheng, W.; and Liu, Y.-S. 2021. PMP-Net: Point cloud completion by learning multi-step point moving paths. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Williams, F.; Schneider, T.; Silva, C.; Zorin, D.; Bruna, J.; and Panozzo, D. 2019. Deep Geometric Prior for Surface Reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wu, H.; Zhang, J.; and Huang, K. 2019. Point Cloud Super Resolution with Adversarial Residual Graph Networks. arXiv:1908.02111.

Wu, S.; Huang, H.; Gong, M.; Zwicker, M.; and Cohen-Or, D. 2015. Deep Points Consolidation. *ACM Transactions on Graphics*, 34(6).

Xiang, P.; Wen, X.; Liu, Y.-S.; Cao, Y.-P.; Wan, P.; Zheng, W.; and Han, Z. 2021. SnowflakeNet: Point Cloud Completion by Snowflake Point Deconvolution with Skip-Transformer. In *IEEE International Conference on Computer Vision (ICCV)*.

Xie, H.; Yao, H.; Zhou, S.; Mao, J.; Zhang, S.; and Sun, W. 2020. GRNet: Gridding Residual Network for Dense Point Cloud Completion. In *European Conference on Computer Vision (ECCV)*.

Yang, Y.; Feng, C.; Shen, Y.; and Tian, D. 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ye, S.; Chen, D.; Han, S.; Wan, Z.; and Liao, J. 2022. Meta-PU: An Arbitrary-Scale Upsampling Network for Point Cloud. *IEEE Transactions on Visualization and Computer Graphics*, 28(9): 3206–3218.

Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2018. PU-Net: Point Cloud Upsampling Network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yuan, W.; Khot, T.; Held, D.; Mertz, C.; and Hebert, M. 2018. PCN: Point Completion Network. In *International Conference on 3D Vision (3DV)*.

Zeng, Y.; Hu, Y.; Liu, S.; Ye, J.; Han, Y.; Li, X.; and Sun, N. 2018. RT3D: Real-Time 3-D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving. *IEEE Robotics and Automation Letters*, 3(4): 3434–3440.

Zhang, F.; Fang, J.; Wah, B.; and Torr, P. 2020. Deep FusionNet for Point Cloud Semantic Segmentation. In *European Conference on Computer Vision (ECCV)*.

Zhang, W.; Yan, Q.; and Xiao, C. 2020. Detail Preserved Point Cloud Completion via Separated Feature Aggregation. In *European Conference on Computer Vision (ECCV)*.

Zhao, H.; Jiang, L.; Jia, J.; Torr, P. H.; and Koltun, V. 2021. Point transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, Y.; Barnes, C.; Jingwan, L.; Jimei, Y.; and Hao, L. 2019. On the Continuity of Rotation Representations in Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.