SPU-Net: Self-Supervised Point Cloud Upsampling by Coarse-to-Fine Reconstruction With Self-Projection Optimization

Xinhai Liu, Xinchen Liu, Yu-Shen Liu, *Member, IEEE*, and Zhizhong Han

***Abstract*— The task of point cloud upsampling aims to acquire dense and uniform point sets from sparse and irregular point sets. Although significant progress has been made with deep learning models, state-of-the-art methods require ground-truth dense point sets as the supervision, which makes them limited to be trained under synthetic paired training data and not suitable to be under real-scanned sparse data. However, it is expensive and tedious to obtain large numbers of paired sparse-dense point sets as supervision from real-scanned sparse data. To address this problem, we propose a self-supervised point cloud upsampling network, named SPU-Net, to capture the inherent upsampling patterns of points lying on the underlying object surface. Specifically, we propose a coarse-to-fine reconstruction framework, which contains two main components: point feature extraction and point feature expansion, respectively. In the point feature extraction, we integrate the self-attention module with the graph convolution network (GCN) to capture context information inside and among local regions simultaneously. In the point feature expansion, we introduce a hierarchically learnable folding strategy to generate upsampled point sets with learnable 2D grids. Moreover, to further optimize the noisy points in the generated point sets, we propose a novel self-projection optimization associated with uniform and reconstruction terms as a joint loss to facilitate the self-supervised point cloud upsampling. We conduct various experiments on both synthetic and real-scanned datasets, and the results demonstrate that we achieve comparable performances to state-of-the-art supervised methods.**

***Index Terms*— Point cloud, upsampling, self-supervised, coarse-to-fine, 2D grids, self-projection.**

I. Introduction

POINT cloud, as one of the most concise 3D representations, has drawn increasing research attention due to its convenient access from various popular depth sensors, such as LiDARs and RGB-D cameras. However, raw point clouds obtained from devices are usually sparse, noisy, and non-uniform, which leads to a massive challenge for deep neural networks to deal with such irregular data directly. Given a sparse, noisy, and non-uniform point cloud, the task of upsampling aims to generate a dense and uniform point set as a trusted representation of the underlying object surface. Therefore, point cloud upsampling, as an amended operation, is meaningful for many downstream applications like rendering, analysis, reconstruction and other general processing.

Current point cloud upsampling methods with deep learning, such as PU-Net [1], MPU [2], PU-GAN [3], PUGeo-Net [4] and Dis-PU [5], have achieved outperforming results on some synthetic datasets such as ShapeNet [6] and Vision-Air repository [7]. However, these methods usually require ground-truth dense point sets as the supervision to train the neural network. And the supervision is usually constructed by sampling on synthetic CAD models from publicly available datasets [1], [3], which is unavailable for the real-scanned data. Due to the absence of paired ground-truth dense point sets, the aforementioned methods cannot be trained under the real-scanned datasets such as ScanNet [8] and KITTI [9]. In addition, when data distributions from synthetic object data do not match those from real scans, upsampling networks trained on synthetic object data do not generalize well to real (sparse) scans. For example, the supervised methods trained on synthetic object data, such as PU-Net [1] and PU-GAN [3], easily changed the origin topological structure of underlying object surface on the real-scanned data or scene point clouds. Therefore, it is promising to propose a self-supervised point cloud upsampling method, which does not require dense point sets as the supervision and can keep the original data distributions.

Recently, some unsupervised image super-resolution methods [10], [11] have been proposed and achieved outperforming performances in generating high-resolution images. However, due to the irregular and unordered nature of point clouds, it is non-trivial to directly apply these image super-resolution methods to the unsupervised point cloud upsampling. Specifically, there are two challenges in the unsupervised point cloud upsampling with deep learning models. (1) *How to establish practical self-supervised information without the supervision of dense point sets?* Previous methods, such as PU-GAN [3] and L2G-AE [12], first generate a dense point set with deep networks and then downsample the dense point set back into a

Manuscript received June 28, 2021; revised February 12, 2022; accepted May 26, 2022. Date of publication June 13, 2022; date of current version June 23, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFF0304100, in part by the National Natural Science Foundation of China under Grant 62072268, and in part by the Tsinghua-Kuaishou Institute of Future Media Data. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ioannis (Yiannis) Kompatsiaris. *(Corresponding author: Yu-Shen Liu.)*

Xinhai Liu, Xinchen Liu, and Yu-Shen Liu are with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: lxh17@mails.tsinghua.edu.cn; lxc19@mails.tsinghua.edu.cn; liuyushen@tsinghua.edu.cn).

Zhizhong Han is with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: h312h@wayne.edu).

Digital Object Identifier 10.1109/TIP.2022.3182266

sparse point set, where some supervision is usually applied to the sparse point sets. However, there is no direct supervision for the dense point sets in the unsupervised upsampling task, which makes it difficult to capture the inherent upsampling patterns. To resolve this issue, we propose a coarse-to-fine reconstruction framework to formulate the self-supervised point cloud upsampling. Specifically, we first downsample the input patch into some coarse patches and then capture the inherent upsampling patterns by reconstructing the input patch itself from the coarse patch. Next, a dense patch can be obtained by aggregating multiple fine patches, which follow the distribution of the input patch. (2) *The point clouds upsampled by deep networks should be a faithful representation of the underlying object surface.* Due to the irregular nature of the point cloud and network bias in the generation of dense point clouds from sparse ones, it is inevitable to bring some noisy points around the underlying surface when generating the upsampled dense point set, especially for unsupervised upsampling methods. Therefore, some specific loss functions are needed to constrain the spatial distribution of generated points, including uniformity and flatness. To resolve these challenges, we propose a novel self-projection optimization to constrain the noisy points around the underlying surface to the surface itself, and associate it with uniform and reconstruction terms as a joint loss to facilitate the generation of upsampled points.

To address the above challenges, we propose the coarse-to-fine reconstruction strategy to explore the upsampling patterns with only sparse point sets. In our approach, we introduce two key components to support the coarse-to-fine point upsampling, named point feature extraction and point feature expansion, respectively. In the point feature extraction module, we integrate self-attention with the graph convolutional network (GCN) to fully capture the spatial context of points within local regions. Furthermore, in the point feature expansion, a hierarchical folding operation with learnable grids is proposed to expand the point features gradually. And to refine the final distribution of generated dense points, a joint loss function is designed to constrain multiple geometry attributes, including uniformity, noise, and overall shape. In general, our contributions are summarised as follows.

- We propose a self-supervised point cloud upsampling network (SPU-Net) which can be trained without the supervision of 3D ground-truth dense point clouds. SPU-Net repeatedly upsamples from downsampled patches, which is not restricted by the paired training data and can preserve the original data distribution.
- We propose a coarse-to-fine reconstruction framework to capture the inherent upsampling patterns inside local patches, which introduces a novel self-supervision way to learn to upsample point clouds.
- To constrain the distribution of generated points without the ground-truth dense point sets, we introduce a novel self-projection optimization, which interactively projects the generated points onto the underlying object surface along the projection direction.

To evaluate the performances of SPU-Net, we adopt four widely used metrics to compare with the state-of-the-art methods under a variety of synthetic and real-scanned datasets. Experimental results show that our self-supervised method achieves good performance, comparable to the supervised methods in both qualitative and quantitative comparisons.

II. Related Work

A. Traditional Point Cloud Upsampling Methods

Traditional methods have tried various optimization strategies to generate the upsampled point clouds without using deep learning models. For example, Alexa *et al.* [13] upsampled the point set by computing the Voronoi diagram and adding points at the vertices of this diagram. Afterward, the locally optimal projection (LOP) operator [14], [15] has been proved to be effective for point resampling and surface reconstruction based on L_1 median, especially for point sets with noise and outliers. Furthermore, Huang *et al.* [16] introduced a progressive strategy for edge-aware point set resampling. To fill large holes and complete missing regions, Dpoints [17] developed a new point representation. In general, all these methods are not in a data-driven manner, which heavily relies on shape priors, such as normal estimation and smooth surface assumption.

B. Deep Learning Based Point Cloud Upsampling Methods

In recent years, deep neural networks have achieved outperforming performances in various point cloud processing tasks, including shape classification [18]–[21], object detection [22], [23], semantic scene segmentation [24]–[26], point cloud reconstruction [12], [27], [28] and point cloud completion [29]–[33]. In the field of point cloud upsampling, Yu *et al.* [1] as a pioneer, first proposed a deep neural network PU-Net to upsample point set, which works on patches by learning multi-level per-point features and expanding the point set via multi-branch convolutions. Later, they designed another edge-aware point cloud upsampling network named EC-Net [34] to achieve point expansion by minimizing the point-to-edge distances. Wang *et al.* [2] presented a multi-step progressive upsampling network to maintain the patch details further. Li *et al.* [3] proposed a GAN-based framework to generate high-quality upsampled point sets. Recently, Qian *et al.* [4] incorporated discrete differential geometry to guide the generation of points in the point cloud upsampling. Li *et al.* [5] proposed a two-step point upsampling strategy, including dense point cloud generating and point spatial refining. Existing methods can already generate high-quality upsampled point clouds under synthetic datasets with the supervision of dense point sets. However, these supervised methods require the ground-truth dense point clouds as the supervision information and are not suitable for the real-scanned data. Liu *et al.* [12] proposed a deep neural network named L2G-AE in representation learning, which can be applied for unsupervised point cloud upsampling by reconstructing overlapped local regions. However, L2G-AE concentrated on the capturing of global shape information via local-to-global reconstruction, which limits the network in capturing inherent upsampling patterns and generating high-quality upsampled point sets. To fully explore the spatial patterns inside sparse point sets, we present SPU-Net

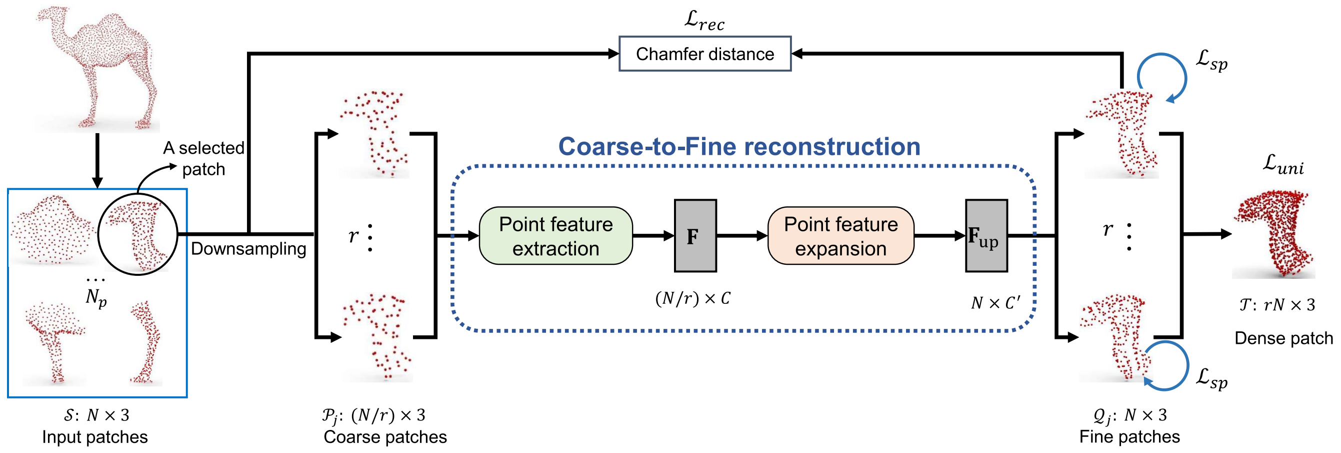


Fig. 1. The architecture of SPU-Net. Given an input patch with N points, we first downsample the input patch into r coarse patches $\mathcal{P}_j$, each with N/r points, where r is the upsampling rate. And then, we generate a fine patch $\mathcal{Q}_j$ for each coarse patch $\mathcal{P}_j$ in a coarse-to-fine reconstruction framework, which consists of point feature extraction and expansion. Finally, we obtain the dense target patch $\mathcal{T}$ by aggregating all the fine patches. In addition, C and C' are the number of feature channels that are 480 and 128, respectively, in the implementation; $\mathcal{L}_{rec}$, $\mathcal{L}_{uni}$ and $\mathcal{L}_{sp}$ denote the reconstruction, uniform, and self-projection terms, respectively.

to upsample from downsampled patches in a coarse-to-fine reconstruction framework, enabling us to generate high-quality upsampled point clouds in a self-supervised manner.

C. Unsupervied Point Cloud Analysis Methods

Recently, several methods have investigated unsupervised or self-supervised learning strategies for point cloud analysis [35]–[37]. The auto-encoder, such as FoldingNet [38] and L2G-AE [12], is a widely used framework for unsupervised point cloud learning, which adopts the input point cloud itself as the reconstruction target. During the self-reconstruction process, the corresponding point features are obtained, which can be applied in various applications, such as shape classification [38], semantic segmentation [39], [40], and shape generation [39]. Based on the encoder-decoder architecture, some works conducted self-supervised information as the optimization target, such as coordinate transformation [41], deformation reconstruction [42], and part partition [43]. In addition, the generative adversarial network [44] was also applied to distinguish the generated point set or the real point set. In this work, we adopt the generalized encoder-decoder framework to build our self-supervised upsampling network based on local patches. In order to capture the inherent upsampling patterns, the key issue is how to reproduce the upsampling process without requiring the supervised dense point sets. To resolve this issue, we propose a coarse-to-fine reconstruction framework to reproduce point cloud upsampling inside local patches. Specifically, we first downsample the input patch into several subsets, and then the coarse-to-fine reconstruction framework is applied to upsample the subsets. By repeating the downsampling and upsampling steps, our method can generate dense upsampled point sets.

III. The SPU-Net Method

A. Overview

Given a 3D point cloud, we take the same patch-based approach as PU-Net [1] and PU-GAN [3]. We first build N_p local patches according to the geodesic distance for the 3D point cloud, as shown in Figure 1. For each local patch $\mathcal{S} = \{\boldsymbol{s}_i\}_{i=1}^{N}$ with N points, our goal is to output a dense and uniform point set $\mathcal{T} = \{\boldsymbol{t}_i\}_{i=1}^{rN}$ with rN points, while keeping the original data distribution, where $\boldsymbol{s}_i$, $\boldsymbol{t}_i$ are the coordinates of 3D points and r is the upsampling rate. Without the supervision of ground-truth dense point sets, we propose a coarse-to-fine reconstruction framework to reproduce the upsampling process inside each local patch. By upsampling the sparse patch to obtain the fine patch, we are able to capture the inherent upsampling patterns for generating dense patch $\mathcal{T}$ that is uniformly distributed on the underlying object surface.

Figure 1 illustrates the architecture of our SPU-Net. For an input patch $\mathcal{S}$, we first downsample $\mathcal{S}$ into r different coarse patches $\{\mathcal{P}_1, \ldots, \mathcal{P}_j, \ldots, \mathcal{P}_r\}$, each with N/r points (Section III-B). We then introduce the coarse-to-fine framework to explore the inherent upsampling patterns inside local patches (Section III-C), which contains two main components: point feature extraction and point feature expansion. Lastly, we present the patch-based training strategy with a joint loss function formed by reconstruction, uniform, self-projection terms (Section III-D).

B. Point Set Downsampling

Without the supervision of dense point sets, we have to construct some self-supervision to support deep networks to capture the inherent upsampling patterns. To take advantage of the input patch without supervision, we propose a coarse-to-fine reconstruction framework to generate the upsampled dense patch. Specifically, as shown in Figure 1, we first downsample the input patch $\mathcal{S}$ into some coarse patches $\mathcal{P}_j$ to formulate the self-supervision with the input patch itself. Since the goal of point cloud upsampling is to generate a dense and uniform point set from a sparse input, a relatively uniform initial input is beneficial for this task. And the downsampling method largely determines the ability of deep networks to capture the surface distribution of 3D

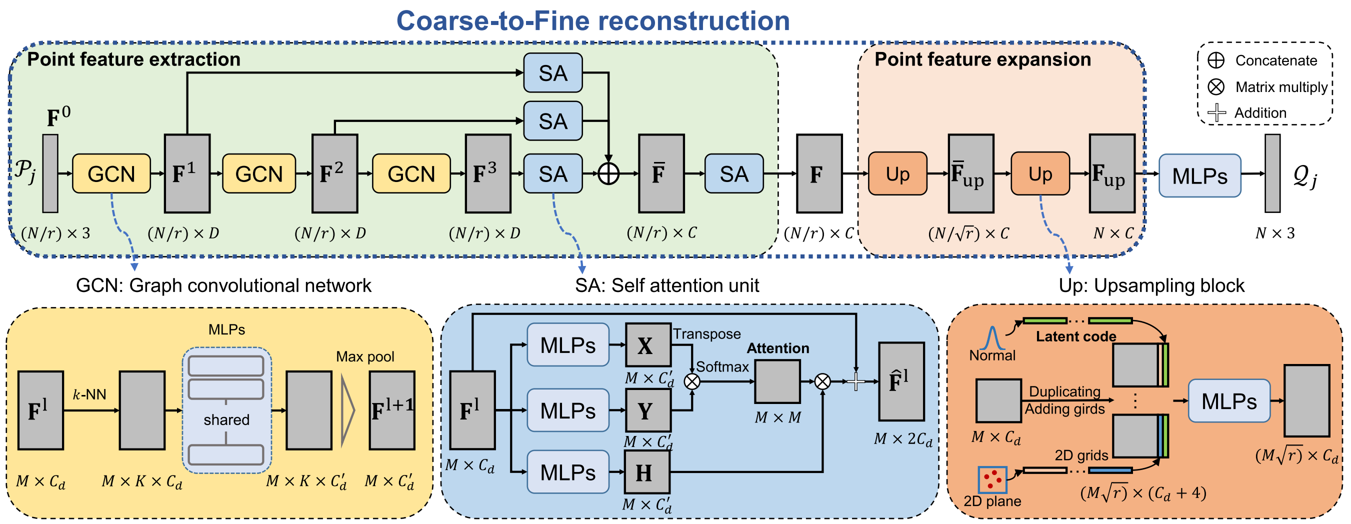


Fig. 2. The coarse-to-fine reconstruction framework. Given a coarse patch $\mathcal{P}_j$ with N/r points, we aim to generate the corresponding fine patch $\mathcal{Q}_j$ with N points. In this framework, there are two main components: point feature extraction and point feature expansion. Here, D and C are the numbers of feature channels that are 64 and 480, respectively, in the implementation.

shapes, which should cover the distribution of the entire input patch as uniformly as possible. Here, the farthest point sampling (FPS) algorithm as a uniform downsampling strategy is adopted in our method, which can generate more uniform coarse patches than other sampling methods such as random sampling and voxel-grid sampling. Specifically, we repeat the process of picking out N/r points from the input local patch $\mathcal{S}$ for r times with the FPS algorithm. Finally, we denote the downsampled coarse patches as $\{\mathcal{P}_1, \ldots, \mathcal{P}_j, \ldots, \mathcal{P}_r\}$. From the downsampled coarse patches, we can reveal the inherent upsampling process by reconstructing the input patch itself, which makes it possible to infer more dense patches.

C. Coarse-to-Fine Reconstruction

In the coarse-to-fine reconstruction framework, there are two main components: point feature extraction and point feature expansion. In the point feature extraction, we integrate self-attention with the graph convolution network (GCN) to simultaneously capture the spatial context of points both inside and among local regions. In the point feature expansion, we propose a hierarchical learnable folding strategy to facilitate the feature propagation from sparse to dense in the feature space.

1) Point Feature Extraction: To capture the context information from discrete points, it is important to extract the spatial correlation of points inside local regions. The graph convolutional network (GCN) [18], [19], [45], [46] has been widely applied to capture the context information inside local regions in existing methods. However, these methods often ignore capturing the correlation among local regions. To simultaneously extract the context information both inside and among local regions, we propose a point feature extraction module, which integrates self-attention units with GCNs, as shown in Figure 2.

Given a coarse patch $\mathcal{P}_j$ with the size of $N/r \times 3$ as input, three GCNs are first employed to capture the local contexts inside local regions by building a local graph around each point $\boldsymbol{p}_i^j$. We introduce a hierarchical feature extraction strategy with multiple semantic levels. Suppose the input feature map of a GCN at level $l \in \{0, 1, 2, 3\}$ is $\boldsymbol{F}^l = \{\boldsymbol{f}_i^l\}_{i=1}^M$ with the size of $M \times C_d$, where $\boldsymbol{f}_i^l$ is the i-th point feature in $\boldsymbol{F}^l$. In particular, the feature map $\boldsymbol{F}^0$ at level 0 is the raw points from $\mathcal{P}_j$. To calculate the point feature $\boldsymbol{f}_i^{l+1}$, we first dynamically build a local region $\mathcal{N}_i^l$ with K neighbors around each point feature $\boldsymbol{f}_i^l$ with the k-NN algorithm. And then, we formulate the propagation of point feature $\boldsymbol{f}_i^l$ $(l \in \{0, 1, 2\})$ as

$$\boldsymbol{f}_i^{l+1} = \underset{\boldsymbol{f}_j^l \in \mathcal{N}_i^l}{max} \{\sigma(\boldsymbol{h_\theta}(\boldsymbol{f}_j^l - \boldsymbol{f}_i^l))\}. \tag{1}$$

Here, $(\boldsymbol{f}_j^l - \boldsymbol{f}_i^l)$ can be regarded as the edge from point $\boldsymbol{f}_j^l$ to center point $\boldsymbol{f}_i^l$, $\boldsymbol{h_\theta}$ indicates the learnable parameters in multi-layer-perceptrons (MLPs), σ is a non-linear layer, such as ReLU [47], and max is a max-pool operation. The max-pool layer is applied to aggregate point features in the local region $\mathcal{N}_i^l$.

Following previous works [3], [12], we integrate self-attention units to capture the correlation between local regions. As shown in Figure 2, the self-attention unit cooperates with the GCN to capture the detailed spatial context information inside local patches. Suppose that the input feature map is $\boldsymbol{F}^l$ with the size of $M \times C_d$. Three MLPs are used to embed $\boldsymbol{F}^l$ into different feature spaces, $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{H}$, respectively. In particular, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are applied to calculate the attention values with simple matrix multiplication, and the updated feature map $\hat{\boldsymbol{F}}^l$ is calculated as

$$\hat{\boldsymbol{F}}^l = \boldsymbol{F}^l + softmax(\boldsymbol{Y}\boldsymbol{X}^\top)\boldsymbol{H}. \tag{2}$$

After multiple self-attention units, the correlation among local regions is captured at different semantic levels. We then aggregate the multi-level point features by concatenation as

$$\bar{\boldsymbol{F}} = MLP(\boldsymbol{F}^1 \oplus \hat{\boldsymbol{F}}^1 \oplus \hat{\boldsymbol{F}}^2 \oplus \hat{\boldsymbol{F}}^3). \tag{3}$$

To further explore the correlation in the aggregated feature $\bar{\boldsymbol{F}}$, we add another self-attention unit to obtain the final point feature $\boldsymbol{F}$.

2) Point Feature Expansion: The target of point feature expansion is to construct the mapping from current points to more points, which is widely used in some point-wise applications, such as semantic segmentation [45], point cloud reconstruction [44] and point cloud upsampling [3]. In general, current point feature expansion methods can be roughly categorized into the interpolating-based method [45], folding-based method [38] and reshaping-based method [44]. Existing point interpolating-based methods often use the point interpolation relationship to guide the expansion of point features. However, in many scenarios, the interpolation relationship between point sets is usually unknown. In addition, reshaping-based methods usually first expand the feature dimensions with deep networks, such as MLPs or fully-connected (FC) layers, and then generate the target point features via a simple reshaping operation. In recent years, the folding-based method has been developed, which first duplicates point features and then concatenates 2D grids to guide point feature expansion.

Compared with other feature expansion methods, the folding-based method is more flexible and has achieved satisfactory performances in various applications [29], [38]. However, previous fixed 2D grids are not adaptive to various feature distributions. To resolve this problem, we propose novel learnable 2D grids as the latent code to cooperate with fixed 2D grids in guiding the point feature expansion. As shown in Figure 2, given an input feature map with the size of $M \times C_d$, we first duplicate the point features and then concatenate two kinds of grids. In particular, the latent code is initialized from a standard normal distribution and can be optimized in the network training process. Moreover, in order to smooth the point feature expansion process, we introduce the hierarchical folding strategy. By using two upsampling blocks, we hierarchically obtain the upsampled point features $\bar{\boldsymbol{F}}_{up}$ and $\boldsymbol{F}_{up}$ with an upsampling rate $\sqrt{r}$. With subsequent MLPs, the point features $\boldsymbol{F}_{up}$ are applied to reconstruct the fine patches.

D. Loss Function

1) Joint Loss: In our SPU-Net, we optimize the upsampled point set with a joint loss $\mathcal{L}_{joint}$, consisting of reconstruction term $\mathcal{L}_{rec}$, uniform term $\mathcal{L}_{uni}$ and self-projection term $\mathcal{L}_{sp}$. Overall, we train our SPU-Net by minimizing the joint loss function in an end-to-end manner as

$$\mathcal{L}_{joint} = \alpha\mathcal{L}_{rec} + \beta\mathcal{L}_{uni} + \gamma\mathcal{L}_{sp}, \tag{4}$$

where α, β and γ are hyper-parameters in training.

2) Reconstruction Term: For an input local patch $\mathcal{S}$, we first downsample the input patch into some coarse patches, and then we repeat upsampling from each one of the coarse patches to explore the inherent upsampling patterns in a self-supervised manner. Assume that $\mathcal{Q}_j = \{\boldsymbol{q}_i^j\}_{i=1}^N$ is the upsampled fine patch from the j-th coarse patch $\mathcal{P}_j$. Thus, we formulate the reconstruction term using chamfer distance (CD) for $\mathcal{Q}_j$ as

$$\mathcal{L}_{rec} = d_{CD}(\mathcal{S}, \mathcal{Q}_j) = \sum_{i=1}^{N} \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{s}_i \in \mathcal{S}} \min_{\boldsymbol{q}_i^j \in \mathcal{Q}_j} ||\boldsymbol{s}_i - \boldsymbol{q}_i^j||_2 + \sum_{i=1}^{N} \frac{1}{|\mathcal{Q}^j|} \sum_{\boldsymbol{q}_i^j \in \mathcal{Q}_j} \min_{\boldsymbol{s}_i \in \mathcal{S}} ||\boldsymbol{s}_i - \boldsymbol{q}_i^j||_2. \tag{5}$$

3) Uniform Term: The reconstruction term encourages the upsampled fine patch $\mathcal{Q}_j$ to fit the input patch $\mathcal{S}$. However, the target dense patch $\mathcal{T} = \{\mathcal{Q}_1, \ldots, \mathcal{Q}_j, \ldots, \mathcal{Q}_r\}$ should be uniformly distributed on the underlying object surface. Similar to [1], [3], we introduce a uniform term to improve the uniformity of the final target patch $\mathcal{T}$ with size of $rN \times 3$. Specifically, we first use the farthest sampling to pick M seed points in $\mathcal{T}$ and apply the ball query of radius r_d to build a local region (denoted as $\mathcal{T}_j, j = 1, \ldots, M$) in $\mathcal{T}$ around each seed. Thus, the number of points roughly relies on the small disk of area πr_d^2 on the underlying surface. In addition, the patch $\mathcal{T}$ is normalized in a unit sphere with an area of $\pi 1^2$. So, the expected number of points $\hat{n}$ in $\mathcal{T}_j$ is rNr_d^2. Suppose the local region $\mathcal{T}_j$ satisfies the regular hexagonal distribution and the distance $d_{j,k}$ is for k-th point in $\mathcal{T}_j$. So, the expected point-to-neighbor distance $\hat{d}$ should be roughly $\sqrt{2\pi r_d^2/(|\mathcal{T}_j|\sqrt{3})}$. To measure the deviation of $|\mathcal{T}_j|$ from $\hat{n}$ and $d_{j,k}$ from $\hat{d}$, we calculate the uniform term with chi-square as

$$\mathcal{L}_{uni}(\mathcal{T}_j) = \sum_{j=1}^{M} U_{number}(\mathcal{T}_j) \cdot U_{distance}(\mathcal{T}_j) = \sum_{j=1}^{M} \frac{(|\mathcal{T}_j| - \hat{n})^2}{\hat{n}} \cdot \sum_{k=1}^{|\mathcal{T}_j|} \frac{(d_{j,k} - \hat{d})^2}{\hat{d}}. \tag{6}$$

4) Self-Projection Term: The reconstruction term and uniform term optimize the geometric shape and surface distribution of the generated upsampled point sets, respectively. However, due to the irregular nature of the point cloud and network bias in the generation of dense point clouds from sparse ones, it is inevitable to bring some noisy points in the generated upsampled point sets. To resolve this problem, we propose a novel self-projection loss function to constrain the generated points to lie roughly on the underlying object surface. Inspired by the traditional directed projection (DP) algorithm [48], each point in the generated set can be projected to the underlying object surface by directed projection. Differently, our SPU-Net is a learning-based method, which constrains the generated point by loss functions. Therefore, our self-projection loss term aims to characterize the local distribution of points and penalize the noise distribution, where the projection process of points is implicit. Considering the distribution differences between input sparse patches and

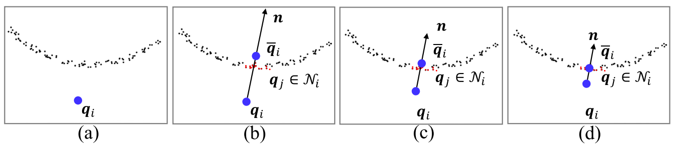


Fig. 3. The illustration of the optimization effect of the self-projection loss term $\mathcal{L}_{sp}$. During the training, the noisy point $\boldsymbol{q}_i$ can be gradually optimized to the underlying object surface by from (a) to (d) minimizing the self-projection term $\mathcal{L}_{sp}$, where $\mathcal{L}_{sp}$ is calculated by measuring the local distribution of noisy point $\boldsymbol{q}_i$, neighbor point $\boldsymbol{q}_j$ and local center $\bar{\boldsymbol{q}}_i$ in the local region $\mathcal{N}_i$.

generated dense patches, we formulate the function inside local regions in a self-projection manner. Suppose the generated fine patch is $\mathcal{Q} = \{\boldsymbol{q}_i\}_{i=1}^N$ with a size of $N \times 3$. We first build a local region $\mathcal{N}_i$ around the i-th point of $\mathcal{Q}$ with the k-nearest-neighbors algorithm. Then, we calculate the center point $\bar{\boldsymbol{q}}_i = \frac{1}{|\mathcal{N}_i|}\sum_{\boldsymbol{q}_j \in \mathcal{N}_i}(\boldsymbol{q}_j)$ of the local region $\mathcal{N}_i$ to be the projection target of the current point $\boldsymbol{q}_j$. Therefore, the self-projection optimization is formulated with chi-square as

$$\mathcal{L}_{sp} = \frac{1}{|\mathcal{Q}| \cdot |\mathcal{N}_j|} \sum_{i=1}^{|\mathcal{Q}|} \sum_{\boldsymbol{q}_j \in \mathcal{N}_i} \frac{|(\boldsymbol{q}_i - \boldsymbol{q}_j)^2 - (\bar{\boldsymbol{q}}_i - \boldsymbol{q}_j)^2|}{1 + (\boldsymbol{q}_i - \boldsymbol{q}_j)^2}. \quad (7)$$

Here, $\frac{1}{1+(\boldsymbol{q}_i - \boldsymbol{q}_j)^2}$ is the weight of each point in the local region $\mathcal{N}_j$. As shown in Figure 3, $(\boldsymbol{q}_i - \boldsymbol{q}_j)^2$ and $(\bar{\boldsymbol{q}}_i - \boldsymbol{q}_j)^2$ indicate the real distribution and the target distribution of the local region, respectively.

IV. Experiments

In this section, we first investigate how some key hyper-parameters affect the point cloud upsampling performance of our SPU-Net. Then, we evaluate SPU-Net by comparing it with state-of-the-art point cloud upsampling methods qualitatively and quantitatively. In addition, we also show some upsampling visualizations under KITTI [9], and ScanNet [8], which illustrates the advantage of our SPU-Net trained without supervision. And we engage in an ablation study of network components in SPU-Net to evaluate the effectiveness of different components. Finally, we conduct a computational complexity analysis of our approach, where the SPU-Net is comparable with other compared methods.

A. Datasets and Network Configurations

For a fair comparison, we do both quantitative and qualitative comparisons with state-of-the-art methods under the dataset from PU-GAN [3], which contains 147 3D models that are formed with 120 objects in the training dataset and 27 objects in the test set. And we follow the same patch-based training strategy as in PU-Net [1], and PU-GAN [3]. By default, we crop 24 patches for each training model and set the number of patch points $N = 256$, the upsampling rate $r = 4$. Moreover, as for the uniform term, the number of seed points M is 50, and we cropped the same set of $\mathcal{T}_j$ with radius $r_d = \sqrt{p}$ for each $p \in \{0.4\%, 0.6\%, 0.8\%, 1.0\%, 1.2\%\}$ as in [3]. We train the network for 200 epochs with the Adam algorithm [49]. Moreover, we set the learning rate as 0.0001, and we gradually reduce both rates by a decay rate of 0.7 per 50k iterations until 10^{-6}. The batch size is 24, and α, β, and γ are empirically set as 100, 10, and 0.01 in Eq. (4), respectively. We implemented our network with TensorFlow and trained it on Nvidia 2,080 Ti GPU.

To evaluate the performance of point cloud upsampling, we employ the same four evaluation metrics as PU-GAN [3], including uniformity (UNI), point-to-surface (P2F), Chamfer distance (CD), and Hausdorff distance (HD). For quantitative evaluation, we use Poisson disk sampling to sample 2,048 points as training data for each 3D object and 8,192 points as the ground truth upsampled point cloud in the testing phase.

TABLE I
The Effects of the Point Downsampling Strategies

Metric	FPS	RS	MS	VS
CD (10^{-3})	**0.38**	0.52	0.55	0.66
HD (10^{-3})	**2.24**	4.19	4.27	7.20
P2F (10^{-3})	**5.87**	7.57	7.22	9.79
UNI (10^{-3})	**8.94**	14.89	9.9	14.60

TABLE II
The Effects of the Number of Region Points K

Metric	$K = 5$	10	15	20
CD (10^{-3})	0.45	**0.38**	0.39	0.41
HD (10^{-3})	3.03	**2.24**	2.51	2.48
P2F (10^{-3})	5.91	**5.87**	6.04	6.12
UNI (10^{-3})	12.47	**8.94**	9.65	9.90

B. Parameters

All the experiments in parameter comparisons are evaluated under the dataset from PU-GAN, where Chamfer distance (CD), Hausdorff distance (HD), point-to-surface (P2F), and uniformity (UNI, $p = 1.2\%$) are adopted as evaluation metrics. For each point cloud with 2,048 points, we sample $N_p = 24$ seed points with FPS and build local patches according to geodesic distance. We initialize the network hyper-parameters, as depicted in the network configurations. Given an input patch, we first downsample this patch into some coarse patches, which should cover the input patch with a uniform distribution in the downsampling process. Thus, the sampling methods influence the neural network to capture inherent upsampling patterns. In Table I, we report the results of three different sampling strategies, including farthest point sampling (FPS), random sampling (RS), mixing of these two sampling strategies (MS) and voxel-grid sampling (VS). The results show that FPS can generate relatively uniform distributions to cover input patches and facilitate point cloud upsampling. The qualitative upsampling results with different downsampling strategies are also illustrated in Figure 4, where the FPS can promote better upsampling results than other strategies.

Then, we investigate the impact of the number of local points K in GCNs of the point feature extraction. Specifically, we range the number of local region points K from 5 to 20. Table II illustrates the results of different K. The best

TABLE III
QUANTITATIVE COMPARISONS WITH THE STATE-OF-THE-ARTS

Methods	Supervised?	Uniformity for different p (10^{-3})					P2F	CD	HD
		0.4%	0.6%	0.8%	1.0%	1.2%	(10^{-3})	(10^{-3})	(10^{-3})
PU-Net [1]	Yes	29.74	31.33	33.86	36.94	40.43	6.84	0.72	8.94
MPU [2]	Yes	7.51	7.41	8.35	9.62	11.13	3.96	0.49	6.11
PU-GAN [3]	Yes	3.38	3.49	3.44	3.91	4.64	2.33	0.28	4.64
Dis-PU [5]	Yes	**2.47**	**1.93**	**2.21**	**2.75**	**3.48**	**2.01**	**0.22**	**2.83**
EAR [50]	No	16.84	20.27	23.98	26.15	29.18	5.82	0.52	7.37
L2G-AE [12]	No	24.61	34.61	44.86	55.31	64.94	39.37	6.31	63.23
Ours (Train2Test)	No	**4.53**	**4.82**	**5.68**	**6.69**	**7.95**	5.97	0.38	2.24
Ours (All2Test)	No	4.71	5.02	5.91	7.03	8.50	**5.79**	**0.37**	2.55
Ours (Test2Test)	No	4.82	5.14	5.86	6.88	8.13	6.85	0.41	**2.18**

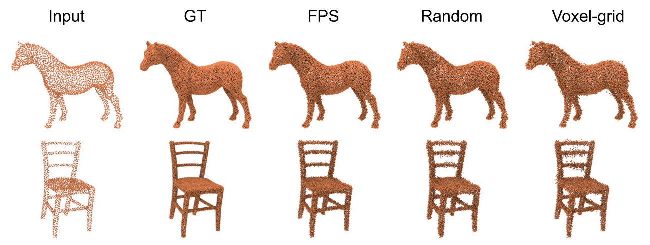


Fig. 4. The visualization of upsampling results in different downsampling strategies under the PU-GAN dataset.

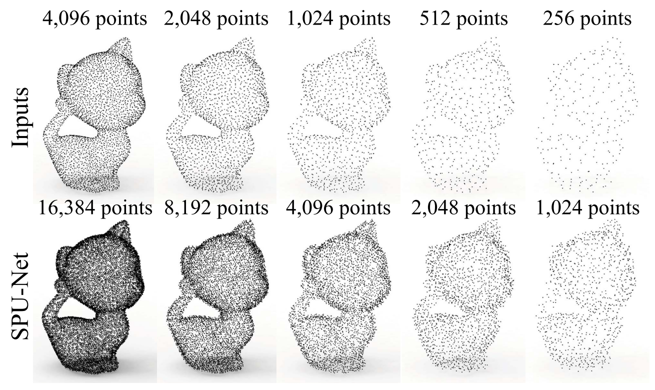


Fig. 5. The upsampling results with varying input sizes from 256 points to 4,096 points.

TABLE IV
THE EFFECTS OF THE NUMBER OF PATCHES AND THE NUMBER OF POINTS INSIDE EACH PATCH

Metric	$N = 128$	256	384	512	$N_p = 12$	18	30	36
CD (10^{-3})	0.45	0.38	0.44	0.41	0.59	0.43	0.40	0.39
HD (10^{-3})	3.26	2.24	2.76	2.50	4.53	2.77	2.37	2.38
P2F (10^{-3})	5.84	5.87	6.55	7.04	6.32	6.23	6.31	6.25
UNI (10^{-3})	16.89	8.94	9.31	8.76	22.00	11.70	8.82	8.28

TABLE V
THE EFFECTS OF THE NUMBER OF LOCAL POINTS k_g IN CALCULATING GEODESIC DISTANCE

Metric	$k_p = 3$	4	5	6
CD (10^{-3})	0.39	0.39	0.38	0.47
HD (10^{-3})	2.25	2.30	2.24	3.18
P2F (10^{-3})	6.10	6.09	5.87	5.93
UNI (10^{-3})	9.11	9.30	8.94	13.64

performance achieves at $K = 10$, which can effectively capture the local region context inside local regions.

Moreover, we also evaluate the robustness of our SPU-Net under different sparsity input point clouds. Figure 5 shows the upsampling point sets with varying sizes of input points from 256 to 4,096. Our method is stable even for input with only 256 points.

Subsequently, we explore the number of sampled points N in each input patch, which influences the distribution of local patches in point clouds. In the experiment, we keep the settings of our network as depicted in the network configuration section and modify the number of patch points N from 128 to 512. The results are shown in Table IV, where the upsampling metrics on the benchmark have a tendency to rise first and then fall. In general, the network reaches the best performance when $N = 256$. Then, we keep the point number $N = 256$ and investigate the effect of the number of patches N_p for each 3D shape. We changed the number of patches N_p from 12 to 36. From the results shown in Table IV, the three metrics slightly fluctuate when N_p goes from 18 to 36, which shows that these patch number settings can cover the entire input point cloud well. On the contrary, the patch number $N_p = 12$ can not cover all the information of point clouds.

Therefore, we employ the number of patches $N_p = 24$ and the number of patch points $N = 256$ as the setting of our network in the following experiments. Then, as shown in Table V, we show the effect of the number of local points k_g in calculating geodesic distance. We ranged the local point k_g from 3 to 6. The above results suggest that $k_g = 5$ is more suitable for our network and achieves the best performances.

Finally, to investigate the ratio of loss functions, we range the ratio α, β and γ of loss functions L_{rec}, L_{uni} and L_{pro} as shown in Table VI. The initialized loss ratios are $\alpha = 100$, $\beta = 10$ and $\gamma = 0.01$. From the experimental results, there is a trade-off between the loss function ratio and the evaluation metric. Therefore, we choose the initialization loss function ratio as the final loss setting.

C. Results on Synthetic Dataset

We compare our SPU-Net with several state-of-the-art upsampling methods both quantitatively and qualitatively, including EAR [50], PU-Net [1], MPU [2], PU-GAN [3],

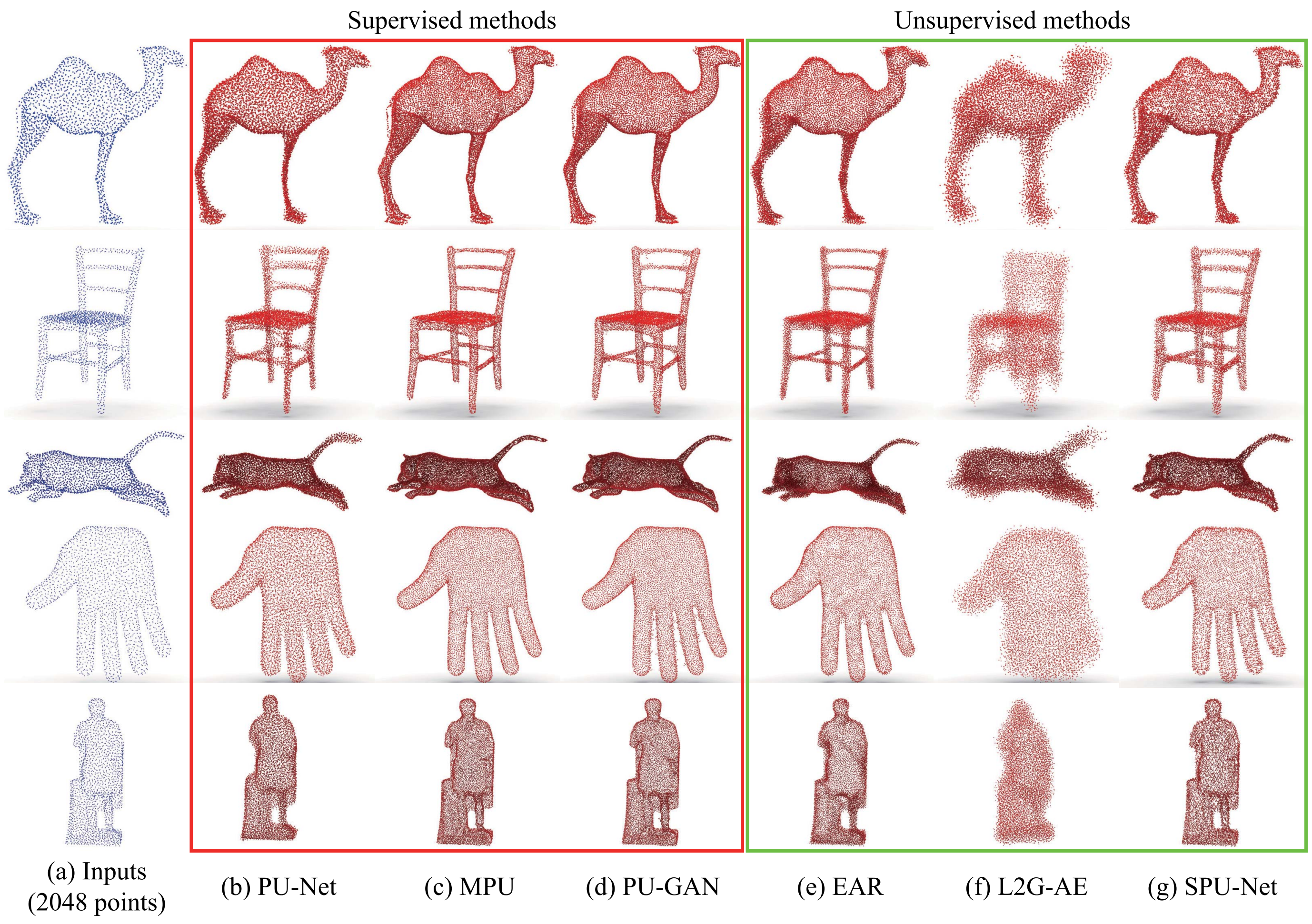


Fig. 6. The visualization results of upsampling from 2,048 points to 8,192 points under the dataset provided by [3]. With only a sparse point cloud as input, our SPU-Net can also achieve comparable results as supervised methods, where the neural network and loss function simultaneously act on the generated point clouds.

TABLE VI
THE EFFECTS OF THE LOSS FUNCTION RATIO α, β AND γ

Metric	$\alpha = 50$	100	200	$\beta = 5$	20	$\gamma = 0.005$	0.02
CD (10^{-3})	0.49	0.38	0.39	0.39	0.40	0.37	0.38
HD (10^{-3})	3.36	2.24	2.64	2.36	2.26	2.32	2.35
P2F(10^{-3})	9.16	5.87	5.55	5.43	6.55	6.04	6.13
UNI (10^{-3})	13.81	8.94	9.46	9.53	9.21	8.12	10.83

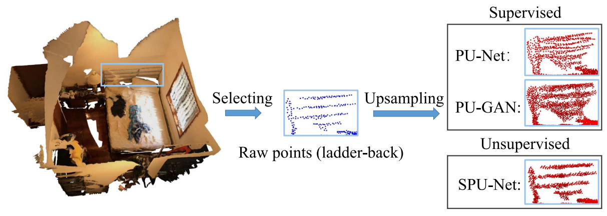


Fig. 7. The upsampling results of a real-scanned object (bed) from the ScanNet dataset [8]. The supervised methods (e.g., PU-Net [1] and PU-GAN [3]) destroy the surface distribution of raw points (the ladder-back of the bed). In contrast, our SPU-Net keeps the origin point distribution.

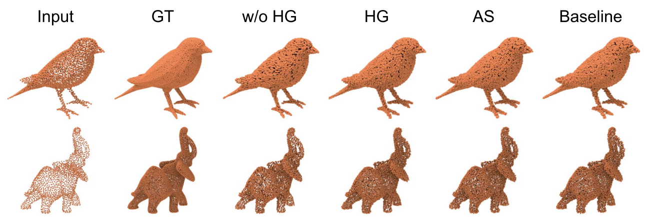


Fig. 8. The visualization of different settings in the folding operation.

Dis-PU [5] and L2G-AE [12]. For EAR, we use its demo code and generate the best results by fine-tuning the associated parameters. Note that, for the recent work PUGeoNet [4], we cannot report the comparison results due to the unavailable code and trained model so far. In addition, as for recent Dis-PU [5], we report the quantitative results tested with public code and trained model. In Table III, our SPU-Net achieves comparable results with existing supervised upsampling networks, such as PU-Net [1], and outperforms another unsupervised method L2G-AE [12]. All evaluation metrics are the same as the ones employed in PU-GAN, where the uniformity is evaluated with varying scales p. And we report multiple testing results of our SPU-Net under different training datasets, including only the training dataset (Train2Test), only the test dataset (Test2Test), and the entire dataset (All2Test). In addition, we also show the qualitative comparisons in

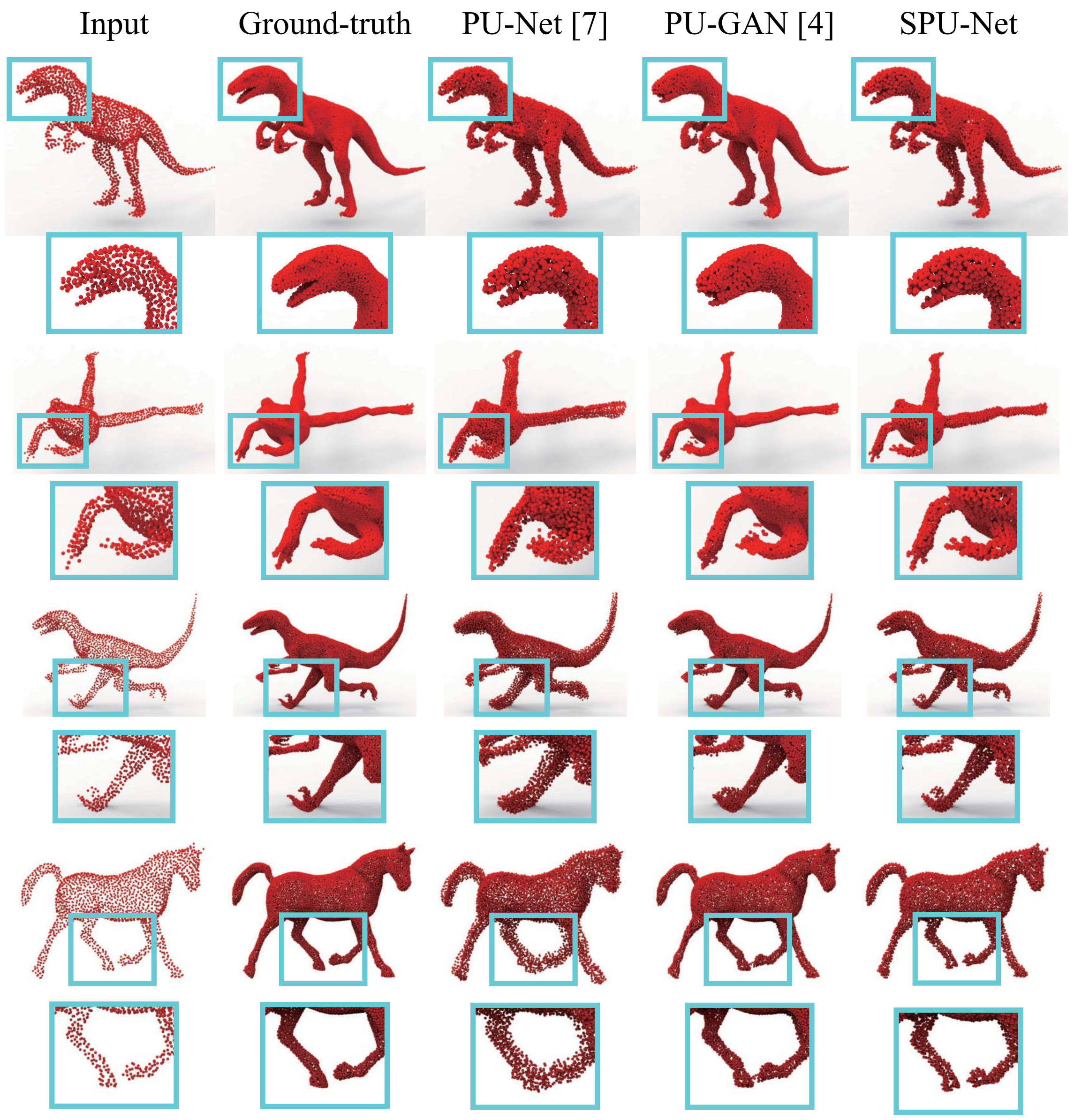


Fig. 9. The upsampling results under the ISDB dataset [51]. For each input sparse point cloud, we generate a 8,192 dense point set from 2,408 input points.

Figure 6, which shows that our SPU-Net can generate upsampled point sets with more details.

To further evaluate our SPU-Net, we also visualize some upsampling results from ISDB [51]. ISDB contains about 104 articulated models of animals and humans, some of that have slight differences between models, which is challenging for the point cloud upsampling task. For each shape, we first sample 2,048 points from the 3D mesh with Poisson Disk Sampling [52]. We aim to generate a dense point set with 8,192 points for each sparse input set with 2,048 points. Here, we also evaluate some supervised methods with the trained models under the dataset from PU-GAN, including PU-Net [1] and PU-GAN [3]. For our SPU-Net, we directly train our network under the ISDB dataset. From the results in Figure 9, we find that our SPU-Net can well preserve the detailed information of the input sparse point clouds.

D. Results on Real-Scanned Dataset

To evaluate the performance of SPU-Net, we directly train the network under the real-scanned datasets, including KITTI [9] and ScanNet [8]. As for preparing training data, we first apply FPS to sample a certain number of seed points and then use KNN to build local patches around these points. Different from supervised methods, our method can be directly trained under real-scanned data, and SPU-Net can preserve the details of raw data as shown in Figure 7. Figures 11 and 12 show our upsampling results on real-scanned point sets. From the results, our SPU-Net can generate dense and uniform upsampled point sets from sparse ones. In particular, all results under the real-scanned data are directly trained on the raw points without ground-truth dense point sets. Therefore, our SPU-Net can expand the training data from synthetic data to the real-scanned data.

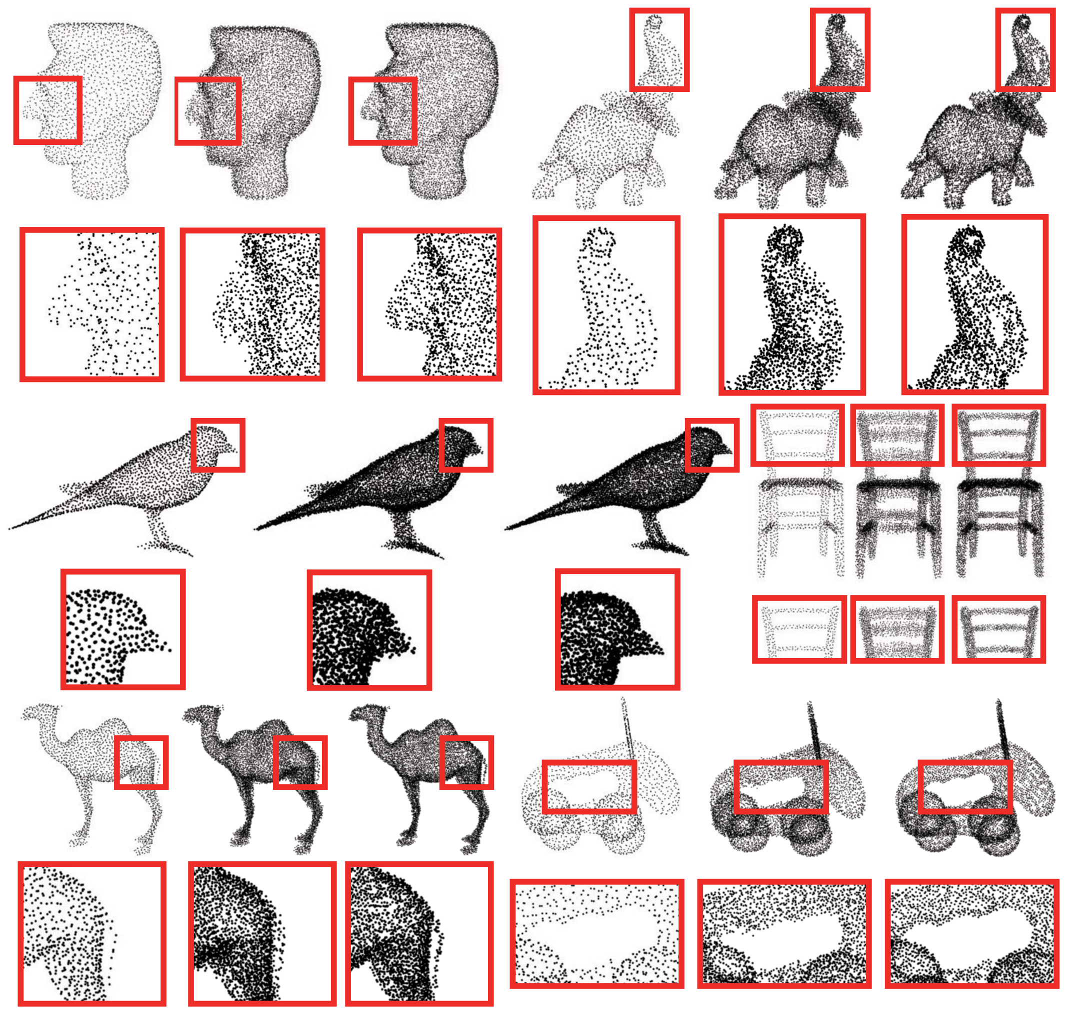


Fig. 10. The effect of the self-projection term. There are six test 3D point clouds from the dataset of PU-GAN [3]. For each point cloud, we show the sparse input point cloud (left), the upsampled point cloud without the self-projection term (middle), and the upsampled point cloud with the self-projection term (right). The red boxes show the details brought about by the self-projection term.

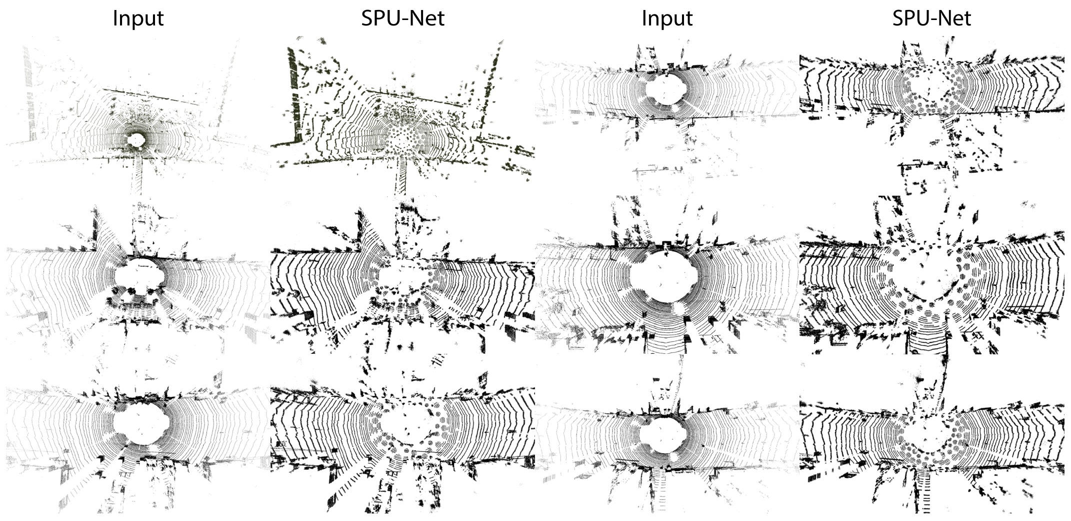


Fig. 11. The upsampling results (×4) under KITTI dataset [9]. We divide the real-scanned scenes into small patches with 256 points for each in training.

E. Ablation Study

To evaluate the components in SPU-Net, including the coarse-to-fine framework (i.e., removing the self-attention unit) and loss function terms, we remove each of them and generate upsampling results for testing models. Specifically, we remove the self-attention unit (No self-attention, NSA),

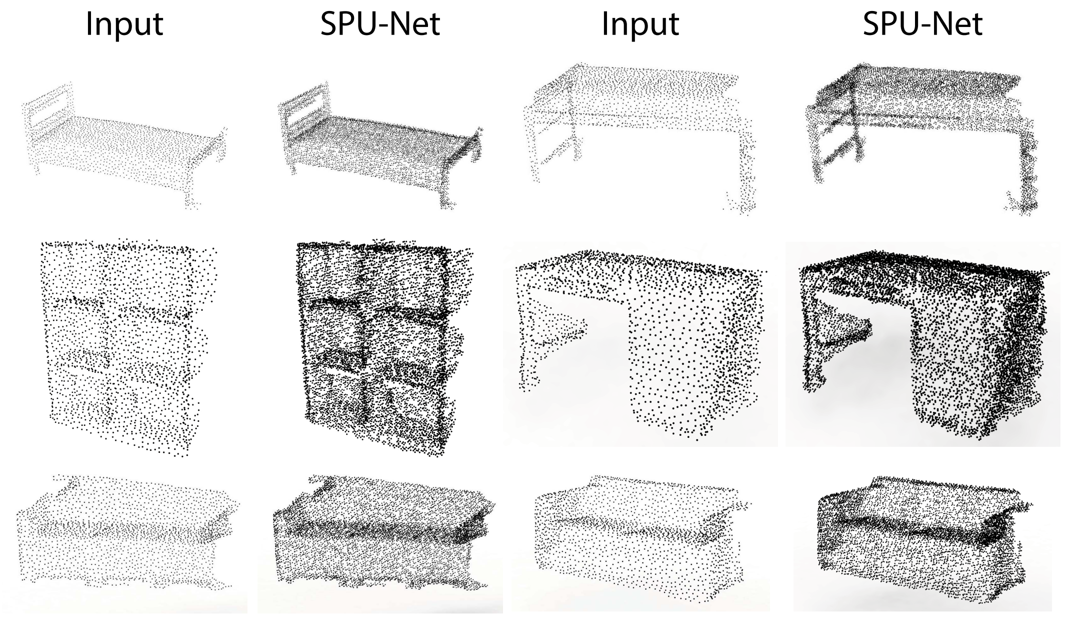


Fig. 12. The upsampling results (×4) under ScanNet dataset [8]. We pick out objects from indoor scenes and divide them into small pathches with 256 points for each for training.

TABLE VII
THE EFFECT OF SOME KEY COMPONENTS IN THE SPU-NET

Metric	Baseline	NSA	NHG	NLG	AS	NRT	NUT	NST	GP
CD (10^{-3})	**0.35**	0.44	0.68	0.36	0.42	4.68	0.38	0.65	0.41
HD (10^{-3})	**2.20**	2.99	5.11	2.36	2.86	34.58	2.55	6.24	2.53
P2F (10^{-3})	5.24	6.45	**3.95**	5.47	5.17	40.44	4.49	9.01	5.93
UNI (10^{-3})	**9.65**	12.23	30.48	9.97	11.15	802.64	13.49	19.38	10.35

the learnable grids (No learnable grid, NLG), the hierarchical grids (No hierarchical grid, NHG), the reconstruction term (No reconstruction term, NRT), the uniform loss (No uniform term, NUT) and the self-projection term (No self-projection term, NST), respectively. All above ablation studies are compared with the full network pipeline (Baseline) and we also show the results of adding supervision at the middle folding layer (AS). Specifically, we apply a single-direction CD loss to constrain the folding process in the AS setting. In addition, we also replace the whole coarse-to-fine framework with the generator of PU-GAN as a new baseline (Generator of PU-GAN, GP) to show the effectiveness of our coarse-to-fine framework. The results in Table VII suggest that all key components play an important role in improving the performance of our SPU-Net, where the network components and the loss function work together to capture the inherent upsampling patterns. In particular, we show the visualization results in Figure 10, which demonstrate the effectiveness of the self-projection loss function in optimizing noisy points to the underlying object surface itself. To intuitively show the impact of the folding settings, some qualitative results are displayed in Figure 8, including without hierarchical grids (w/o HG), only hierarchical grids (HG), adding supervision in the middle folding layer (AS) and our base model (Baseline). The results show that the hierarchical grids are important for constraining the point distribution of point upsampling. And some middle supervision might be helpful for optimizing the final distribution of points.

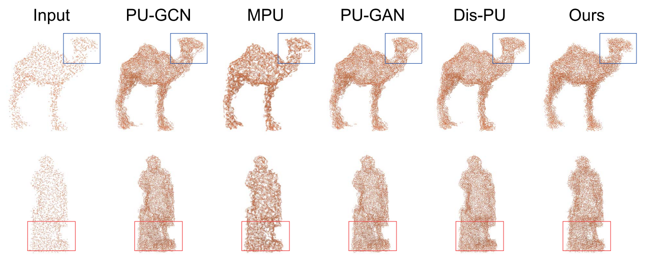


Fig. 13. The comparison of different methods under PU-GAN data with noise level of 2%. The colored boxes represent some local details of the point cloud upsampling.

1) Noise Effect: The upsampling results under different levels of noise are revealed in Figure 14. In our SPU-Net, the proposed network learns to infer the dense points in a self-supervised manner, which is a challenging task. In order to constrain the distribution of upsampled points, we introduce a joint loss function, including reconstruction term, uniform term and self-projection term. Benefitted from the self-projection term, our SPU-Net is robust to generate dense points under different noise levels of 0.1%, 0.5%, 1% and 2%.

As shown in Figure 13, we also visually compare the results with different methods under PU-GAN data with noise level 2%. Benefit from the joint loss function and coarse-to-fine upsampling strategy, our SPU-Net can explore upsampling patterns inside input sparse patches. From the results, our SPU-Net can keep the details of the input sparse point clouds, which demonstrates our good generalization ability to noisy point clouds.

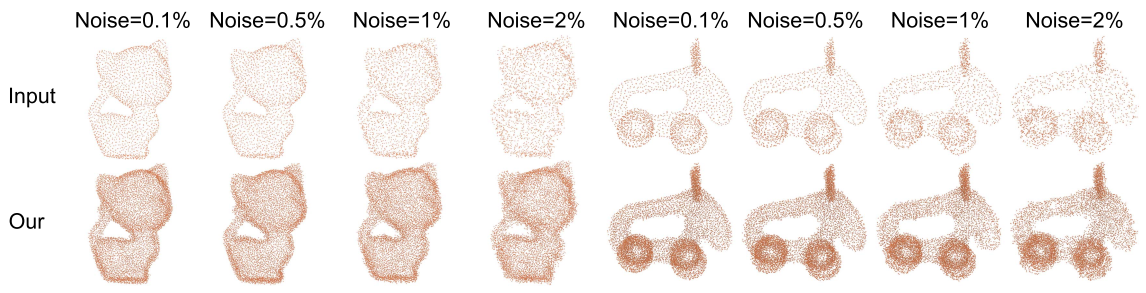


Fig. 14. The result of our SPU-Net under different noise levels of 0.1%, 0.5%, 1%, and 2%. Our SPU-Net is relatively robust to the input.

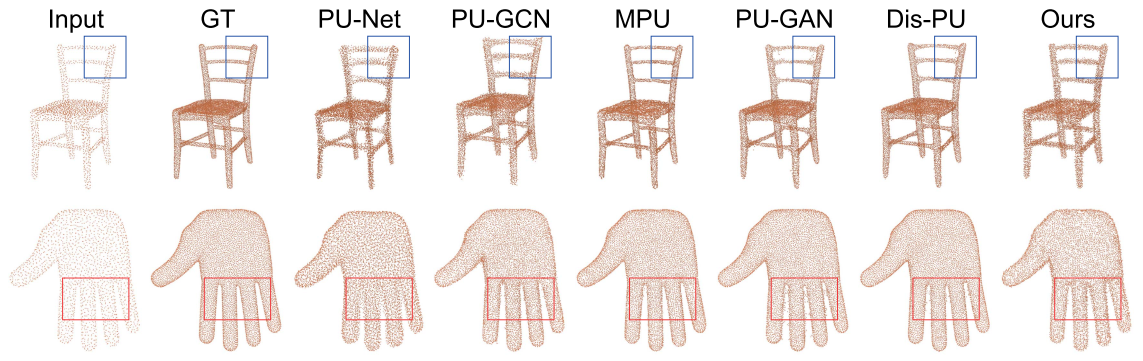


Fig. 15. Some comparisons of different upsampling methods under PU-GAN dataset.

TABLE VIII
THE ANALYSIS OF COMPUTATIONAL COMPLEXITY

Methods	GFLOPs	Training parameters (Mb)	Forward time (ms)
PU-Net [1]	15.03	0.81	135.88
MPU [2]	27.52	**0.30**	368.81
PU-GAN [3]	5.67	0.68	146.16
L2G-AE [12]	15.43	100.18	185.10
Ours (SPU-Net)	**1.86**	0.68	**130.92**

2) Local Distribution Comparisons: We compare our SPU-Net with existing supervised point cloud upsampling methods under the PU-GAN dataset in Figure 15. There are some situations that existing supervised methods cannot solve very well. In some local regions, such as the ladder-back of the chair and the fingers of the hand, our SPU-Net can obtain a relatively clean distribution in the coarse-to-fine point upsampling.

F. Computational Complexity

In this section, we do a computational complexity analysis of our SPU-Net by comparing it with some state-of-the-art upsampling methods. In Table VIII, we adopt the GFLOPs, training parameters, and forward time as the evaluation metrics. The compared methods include PU-Net [1], MPU [2], PU-GAN [3] and L2G-AE [12]. We reproduce their results using their released code, where the statistical analysis is conducted using the built-in functions in TensorFlow. Before the evaluation, we initialize the batch size of all methods to 1. The comparison in Table VIII demonstrates that our method is efficient in terms of computational performance.

V. CONCLUSION

In this paper, we propose a novel self-supervised point cloud upsampling method to generate dense and uniform point set from sparse inputs without the supervision of ground-truth dense point clouds. Our coarse-to-fine reconstruction framework effectively facilitates point cloud upsampling by point feature extraction and point feature expansion. In addition, our self-projection optimization successfully projects noisy points onto the underlying object surface itself, which greatly improves the quality of point cloud upsampling in an unsupervised manner. Our experimental results demonstrate that our method can achieve good performance on both synthetic and real-scanned datasets, even comparable results to the state-of-the-art supervised method.

ACKNOWLEDGMENT

The code is available at https://github.com/liuxinhai/SPU-Net

REFERENCES

[1] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, “PU-Net: Point cloud upsampling network,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2790–2799.
[2] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, “Patch-based progressive 3D point set upsampling,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5958–5967.
[3] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, “PU-GAN: A point cloud upsampling adversarial network,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7203–7212.
[4] Y. Qian, J. Hou, S. Kwong, and Y. He, “PUGeo-Net: A geometry-centric network for 3D point cloud upsampling,” in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 752–769.

[5] R. Li, X. Li, P.-A. Heng, and C.-W. Fu, "Point cloud upsampling via disentangled refinement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 344–353.

[6] A. X. Chang *et al.*, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.

[7] *VisionAir*. Accessed: Jun. 28, 2021. [Online]. Available: http://www.infra-visionair.eu/

[8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5828–5839.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[10] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 701–710.

[11] T. R. Shaham, T. Dekel, and T. Michaeli, "SinGAN: Learning a generative model from a single natural image," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4570–4580.

[12] X. Liu, Z. Han, X. Wen, Y.-S. Liu, and M. Zwicker, "L2G auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention," in *Proc. ACM Int. Conf. Multimedia*, 2019, pp. 989–997.

[13] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Trans. Vis. Comput. Graphics*, vol. 9, no. 1, pp. 3–15, Feb. 2003.

[14] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Trans. Graph.*, vol. 26, no. 3, p. 22, Jul. 2007.

[15] H. Hui, L. Dan, Z. Hao, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–7, 2009.

[16] H. Huang, S. Wu, M. Gong, D. Cohen-Or, and H. Zhang, "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 1–12, 2013.

[17] S. Wu, H. Huang, M. Gong, M. Zwicker, and D. Cohen-Or, "Deep points consolidation," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 1–13, 2015.

[18] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8895–8904.

[19] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8778–8785.

[20] X. Wen, Z. Han, X. Liu, and Y.-S. Liu, "Point2SpatialCapsule: Aggregating features and spatial relationships of local regions on point clouds using spatial-aware capsules," *IEEE Trans. Image Process.*, vol. 29, pp. 8855–8869, 2020.

[21] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Fine-grained 3D shape classification with hierarchical part-view attention," *IEEE Trans. Image Process.*, vol. 30, pp. 1744–1758, 2021.

[22] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, "ImVoteNet: Boosting 3D object detection in point clouds with image votes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4404–4413.

[23] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3D object detection from point cloud with part-aware and part-aggregation network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2647–2664, Aug. 2021.

[24] Q. Hu *et al.*, "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11108–11117.

[25] H. Shi, G. Lin, H. Wang, T.-Y. Hung, and Z. Wang, "SpSequenceNet: Semantic segmentation network on 4D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4574–4583.

[26] X. Wen, Z. Han, G. Youk, and Y.-S. Liu, "CF-SIS: Semantic-instance segmentation of 3D point clouds by context fusion with self-attention," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 1661–1669.

[27] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, "Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces," in *Proc. Int. Conf. Mach. Learn.*, 2021, vol. 139, pp. 7246–7257.

[28] Z. Han, B. Ma, Y.-S. Liu, and M. Zwicker, "Reconstructing 3D shapes from multiple sketches using direct shape optimization," *IEEE Trans. Image Process.*, vol. 29, pp. 8721–8734, 2020.

[29] X. Wen, T. Li, Z. Han, and Y.-S. Liu, "Point cloud completion by skip-attention network with hierarchical folding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1939–1948.

[30] X. Wang, M. H. Ang, and G. H. Lee, "Cascaded refinement network for point cloud completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 790–799.

[31] X. Wen *et al.*, "PMP-Net: Point cloud completion by learning multi-step point moving paths," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7443–7452.

[32] X. Wen, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu, "Cycle4Completion: Unpaired point cloud completion using cycle transformation with missing region coding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13080–13089.

[33] P. Xiang *et al.*, "SnowflakeNet: Point cloud completion by snowflake point deconvolution with skip-transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5499–5509.

[34] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "EC-Net: An edge-aware point set consolidation network," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 386–402.

[35] J. Sauder and B. Sievers, "Self-supervised deep learning on point clouds by reconstructing space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 12962–12972.

[36] C. Sharma and M. Kaul, "Self-supervised few-shot learning on point clouds," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7212–7221.

[37] B. Eckart, W. Yuan, C. Liu, and J. Kautz, "Self-supervised learning on 3D point clouds by learning discrete generative models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8248–8257.

[38] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 206–215.

[39] Z. Han, X. Wang, Y.-S. Liu, and M. Zwicker, "Multi-angle point cloud-VAE: Unsupervised feature learning for 3D point clouds from multiple angles by joint self-reconstruction and half-to-half prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 10441–10450.

[40] K. Hassani and M. Haley, "Unsupervised multi-task feature learning on point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8160–8171.

[41] X. Gao, W. Hu, and G.-J. Qi, "GraphTER: Unsupervised learning of graph transformation equivariant representations via auto-encoding node-wise transformations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7163–7172.

[42] I. Achituve, H. Maron, and G. Chechik, "Self-supervised learning for domain adaptation on point clouds," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 123–133.

[43] L. Zhang and Z. Zhu, "Unsupervised feature learning for point cloud understanding by contrasting and clustering using graph convolutional neural networks," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2019, pp. 395–404.

[44] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 40–49.

[45] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[46] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Nov. 2019.

[47] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

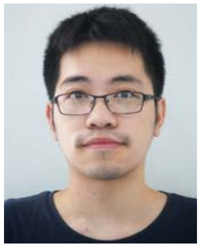
[48] Y.-S. Liu *et al.*, "Automatic least-squares projection of points onto point clouds with applications in reverse engineering," *Comput.-Aided Des.*, vol. 38, no. 12, pp. 1251–1263, Dec. 2006.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–13.

[50] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 1–12, Jan. 2013.

[51] R. Gal, A. Shamir, and D. Cohen-Or, "Pose-oblivious shape signature," *IEEE Trans. Vis. Comput. Graphics*, vol. 13, no. 2, pp. 261–271, Mar. 2007.

[52] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 6, pp. 914–924, Jun. 2012.



Xinhai Liu received the B.S. degree in computer science and technology from the Huazhong University of Science and Technology, China, in 2017. He is currently pursuing the Ph.D. degree with the School of Software, Tsinghua University. His research interests include deep learning, 3D shape analysis, and pattern recognition.

Xinchen Liu received the B.S. degree in software engineering from the Hefei University of Technology, China, in 2019. He is currently pursuing the master's degree with the School of Software, Tsinghua University. His research interests include deep learning, point cloud completion, and upsampling.

Yu-Shen Liu (Member, IEEE) received the B.S. degree in mathematics from Jilin University, China, in 2000, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2006. From 2006 to 2009, he was a Postdoctoral Researcher with Purdue University. He is currently an Associate Professor with the School of Software, Tsinghua University. His research interests include shape analysis, pattern recognition, machine learning, and semantic search.

Zhizhong Han received the Ph.D. degree from Northwestern Polytechnical University, China, in 2017. He was a Postdoctoral Researcher with the Department of Computer Science, University of Maryland, College Park, USA. Currently, he is an Assistant Professor of computer science with Wayne State University, USA. His research interests include 3D computer vision, digital geometry processing, and artificial intelligence.