SSPU-Net: Self-Supervised Point Cloud Upsampling via Differentiable Rendering

Yifan Zhao†
Nanjing University of Science and Technology, China
meidike@outlook.com

Le Hui†
Nanjing University of Science and Technology, China
le.hui@njust.edu.cn

Jin Xie‡
Nanjing University of Science and Technology, China
csjxie@njust.edu.cn

ABSTRACT
Point clouds obtained from 3D sensors are usually sparse. Existing methods mainly focus on upsampling sparse point clouds in a supervised manner by using dense ground truth point clouds. In this paper, we propose a self-supervised point cloud upsampling network (SSPU-Net) to generate dense point clouds without using ground truth. To achieve this, we exploit the consistency between the input sparse point cloud and generated dense point cloud for the shapes and rendered images. Specifically, we first propose a neighbor expansion unit (NEU) to upsample the sparse point clouds, where the local geometric structures of the sparse point clouds are exploited to learn weights for point interpolation. Then, we develop a differentiable point cloud rendering unit (DRU) as an end-to-end module in our network to render the point cloud into multi-view images. Finally, we formulate a shape-consistent loss and an image-consistent loss to train the network so that the shapes of the sparse and dense point clouds are as consistent as possible. Extensive results on the CAD and scanned datasets demonstrate that our method can achieve impressive results in a self-supervised manner. Code is available at https://github.com/fpthink/SSPU-Net.

KEYWORDS
Point Cloud Upsampling, Differentiable Rendering, Deep Neural Networks

1 INTRODUCTION
In recent years, as an important 3D data representation, point clouds have received more attention and has been gradually applied to many fields such as autonomous driving [6, 10, 30, 33, 36, 48, 59], robotics [5, 14, 18, 19, 29, 40, 60] and virtual reality [2, 4, 43, 52]. Generally, the raw point clouds obtained by 3D sensors are usually sparse. Sparse point clouds cannot characterize local geometric structures of 3D objects well, which may lead to the poor performance of the downstream point cloud processing tasks such as 3D object segmentation, detection and classification. Therefore, it is necessary to upsample sparse point clouds to generate dense and complete point clouds that can facilitate the subsequent point cloud processing tasks.

In the past decades, some optimization based upsampling methods [15, 50] were proposed to upsample point clouds, which usually rely on geometric priors, such as normal estimation and smooth surfaces. Nonetheless, the complex geometric structures of 3D objects usually limit performance of these methods. Recently, a lot of deep learning based supervised methods [23, 39, 54–57] were proposed to tackle the point cloud upsampling task. As a pioneering work, point cloud upsampling network (PU-Net) [57] proposed a network paradigm for point cloud upsampling, including three stages of feature extraction, feature augmentation, and coordinate restoration. To improve the quality of upsampled point clouds, multi-step progressive upsampling (MPU) network [55] used a cascaded patch-based upsampling network to progressively upsample point clouds on different levels of point clouds in an end-to-end manner. Lately, based on generative adversarial network (GAN), point cloud upsampling adversarial network (PU-GAN) [23] learned the distribution of points from the latent space and upsampled points on the surface patches of the object. These methods can achieve impressive point cloud upsampling results in a supervised manner by using dense ground truth point clouds. Nonetheless, it is difficult to acquire dense ground truth point clouds in the real scenarios. Therefore, it is desirable to develop self-supervised/unsupervised point cloud upsampling methods without using ground truth.

In this paper, we propose a self-supervised point cloud upsampling network (SSPU-Net), which can be trained in an end-to-end manner and does not require ground truth. To enable self-supervised learning, we encourage the input sparse point cloud and the generated dense point cloud to have similar geometric structures by imposing consistency constraints on the 3D shapes and rendered images of the sparse and dense point clouds. Specifically, we first propose a neighbor expansion unit (NEU) to upsample the sparse point cloud to the dense point cloud, where we exploit the local geometric structures of each point to adaptively learn weights to interpolate the features of new points. Besides, we also adopt the 2D grid mechanism in [23] to generate a unique 2D vector for each point feature in the NEU. After stacking two NEUs, we use a multi-layer perceptron (MLP) to generate the 3D coordinates of the dense point cloud. Then, we develop a differentiable rendering unit (DRU) to render the input sparse point cloud and the generated dense point cloud into multi-view images. In order to make the sparse and dense point clouds have similar geometric structures, we formulate a shape-consistent loss on point clouds and an image-consistent loss on rendered images to train the point cloud upsampling network. Experimental results on the CAD and scanned datasets demonstrate that our SSPU-Net can achieve impressive results in a self-supervised manner. In comparison to the supervised point cloud upsampling methods, our upsampling results are even better than PU-Net [57]. In addition, we also conduct experiments on the downstream point cloud classification and point cloud based place recognition tasks. The experimental results further show that the

†Equal contribution.
‡Corresponding author.
Figure 1: Overview of the self-supervised point cloud upsampling network (SSPU-Net). The network inputs the sparse point cloud $S$ containing $N$ points, and outputs the dense point cloud $D$ after being expanded by the neighbor expansion unit (NEU). Note that $r$ is the upsampling rate. Consequently, we adopt a differentiable rendering unit (DRU) to generate multi-view rendering images from the obtained dense point cloud. To train our network in a self-supervised manner, we propose a shape-consistent loss $L_{sc}$ and an image-consistent loss $L_{ic}$ to encourage the shapes of the sparse and dense point clouds to be consistent.

Upsampled point clouds by our self-supervised method can improve the performance of these tasks.

The main contributions of the paper are as follows:

1. We propose a neighbor extended unit (NEU) to upsample the sparse point clouds by adaptively learning weights from local geometric structures of the point clouds. We also develop a differentiable rendering unit (DRU) to render the point clouds into multi-view images, which can be nested in the network for end-to-end training.

2. We formulate a shape-consistent loss and an image-consistent loss to make the sparse and dense point clouds consistent so that the point cloud upsampling network can be trained in a self-supervised manner.

3. Our method can achieve impressive results in a self-supervised manner, and outperforms some supervised point cloud upsampling methods.

2 RELATED WORK

Optimization based point cloud upsampling. Generally, optimization based methods use geometric priors, such as normal estimation and smooth surfaces, to upsample the sparse point clouds to the dense point cloud. Alexa et al. [1] interpolated new points at the vertex of the Voronoi diagram of the local tangent plane of the surfaces to implement point cloud upsampling. In order to get rid of local plane fitting, Lipman et al. [26] proposed a local optimization mapping operator (LOP) to approximate the surface, in which we can resample the point cloud to obtain the dense point cloud. Likewise, Huang et al. [15] developed an edge-aware point set resampling method, called EAR, where the LOP is used to push points away from the edges. Recently, Wu et al. proposed a consolidation method [50] based on deep point representation to fill in the points in the missing areas. To implement point cloud upsampling, it uses the meso-skeleton of the 3D object to resample the points on the surface. Although these methods can achieve good upsampling results, they heavily rely on handcrafted geometric priors of the point cloud. For complex geometric structures of 3D objects, the performance of these methods is limited by the poor geometric priors.

Learning based point cloud upsampling. With the success of deep learning on point clouds, numerous efforts have been made for point cloud processing, including classification [35, 37, 38, 49], semantic segmentation [7, 8, 25, 41, 61], completion [58], generation [17, 42, 53], based on PointNet++ [38], Yu et al. [57] proposed a point cloud upsampling network (PU-Net), which is the first deep learning based point cloud upsampling method. It first learns multi-level features of each point and then executes feature expansion to expand point features. Finally, the expanded point feature is used to generate the 3D coordinates of the dense point cloud. Based on PU-Net, edge-aware point set consolidation network (EC-Net) [56] learns edge-aware features to better capture the local geometric structures of the point cloud. Although EC-Net can achieve high performance than PU-Net, it additionally requires the triangular surface annotation of point clouds. Lately, Wang et al. [55] proposed a patch-based progressive upsampling network (MPU), which can capture detailed structures at different levels of point clouds through a cascaded upsampling network. Li et al. [23] proposed point cloud upsampling adversarial network (PU-GAN), which uses the generative adversarial network to improve the quality of the upsampled point cloud with the adversarial training strategy. Although the above methods have achieved impressive results, they all require ground truth for supervised training.

Differentiable rendering. In order to obtain 2D representation of the corresponding 3D objects, many existing methods utilize rendering techniques, such as 3D surface reconstruction [12, 44–46], material inference [9, 27], and 3D reconstruction tasks [13, 22, 62]. Loper et al. [32] proposed an approximate differentiable renderer (DR), which explicitly learns the relationship between...
changes in model parameters and image observations. Latter, Kato et al. [20] developed a hand-crafted function that can approximate the backward gradient of rasterization to achieve differentiable rendering. Likewise, Li et al. proposed a method [24] to realize the differentiability of the secondary rendering effect, which uses a differentiable Monte Carlo ray tracer. Lately, Liu et al. [28] proposed a differentiable mesh rendering method called Soft Rasterizer, which can directly render colorized meshes using differentiable functions. In this paper, we use the differentiable rendering method to render point clouds into multi-view images.

3 METHOD

Assuming that a sparse point cloud \( S = \{ s_i \}_{i=1}^{N} \) has \( N \) points, where \( s_i \) is the 3D coordinates, the goal of our self-supervised method is to generate a dense point cloud \( D = \{ d_i \}_{i=1}^{N} \) \((r \) is the upsampling rate\), where \( d_i \) is the upsampled 3D coordinates and \( rN \) is the number of the dense point cloud. As shown in Figure 1, we illustrate the framework of our self-supervised point cloud upsampling network (SSPU-Net). The framework mainly consists of two modules: a neighbor expansion unit (NEU) and a differentiable rendering unit (DRU). In our SSPU-Net framework, we use the sparse point cloud as the input for feature extraction. Following MPU [55], we use the dense feature extraction unit to extract the local features of the sparse point clouds. Then, we feed the obtained point features into two stacked NEUs to adaptively learn weights from the local geometric structures of point clouds for point interpolation. To obtain the upsampled 3D point cloud, we use a multi-layer perceptron (MLP) to generate the 3D coordinates of the dense point cloud. After obtaining the upsampled point cloud, in the rendering stage, we use the DRU to render the input sparse point cloud and the generated dense point cloud into multi-view images. To train our self-supervised method, we formulate a shape-consistent loss and an image-consistent loss to encourage the shapes of the sparse and dense point clouds to be consistent. Since the DRU is differentiable, the network can be trained in an end-to-end manner.

3.1 Neighbor Expansion Unit

Given a learned feature map \( X = [x_1, x_2, \ldots, x_N] \) of the sparse point cloud \( S \) and the expansion factor \( r \), the neighbor expansion unit (NEU) aims to learn a dense feature map \( Y = [y_1, y_2, \ldots, y_rN] \). Existing methods such as PU-Net [57], MPU [55] and PU-GAN [23] mainly execute feature expansion by duplicating point features without considering the local geometric structures of the point cloud. Since ground truth point clouds are used to train the network, they can achieve good upsampling results. However, in the self-supervised upsampling task, the feature expansion operations in these methods cannot generate high-quality point features due to the unavailability of ground truth point cloud. To this end, we propose the NEU to adaptively learn weights from the local geometric structures of the point cloud for point interpolation.

As illustrated in Figure 2, we depict the detailed structure of the NEU. Given the \( i \)-th point feature \( x_i \in \mathbb{R}^c \), where \( c \) is the feature dimension, we first construct the local neighborhood \( N_i \) with the \( r \)-nearest points in the spatial space to construct the feature map \( A \in \mathbb{R}^{r\times c} \), where the integer \( r \) is the upsampling rate. At the same time, we repeat the feature vector \( x_i \) for \( r \) times to generate the feature map \( B \in \mathbb{R}^{r\times c} \). By concatenating feature maps \( A \) and \( B \), we can obtain the new feature map \( F = [A; B] \), where \([::]\) represents the channel-wise concatenation and \( F \in \mathbb{R}^{r\times 2c} \). In this way, we construct the local neighborhood per point. After that, we utilize the multi-layer perceptron (MLP) on the feature map \( F \) to adaptively learn the weight \( W \in \mathbb{R}^{r\times 2} \) for point interpolation. As shown in Figure 2, we split the weight \( W \in \mathbb{R}^{r\times 2} \) into two parts of the weight \( \alpha \in \mathbb{R}' \) and the weight \( \beta \in \mathbb{R}' \). \( \alpha \) and \( \beta \) represent the weights of

\[
\begin{align*}
A: r \times c \\
A': r \times c \\
k-NN \\
x_i: 1 \times c \\
\text{k-NN} \\
\text{tile} \\
B: r \times c \\
B': r \times c \\
\text{MLP} \\
\text{MLP} \\
\text{MLP} \\
\text{Matrix mul.} \\
\text{Matrix add.} \\
\text{Softmax} \\
\text{Concatenation} \\
\text{Element-wise mul.} \\
\text{Matrix add.} \\
\text{Element-wise mul.} \\
\text{Output} \\
\end{align*}
\]

Figure 2: Illustration of the neighbor expansion unit (NEU). Given the point feature \( x_i \in \mathbb{R} \), we first use the \( k \)-nearest neighbor to construct the local feature map \( A \in \mathbb{R}^{r\times c} \), and repeat the point feature \( x_i \) for \( r \) times to generate the feature map \( B \in \mathbb{R}^{r\times c} \). Then, we adaptively learn the weights \( \alpha \in \mathbb{R}' \) and \( \beta \in \mathbb{R}' \) from the concatenated feature map \( F \in \mathbb{R}^{r\times 2c} \). Consequently, we use \( \alpha \) and \( \beta \) to weight the feature maps \( A \) and \( B \) to obtain a new feature map. Finally, we use the 2D grid mechanism and the self-attention mechanism to generate the final feature map.
the adjacent points and center points, respectively. Then, the point interpolation is formulated as:

\[ h_j = \gamma(\alpha_j) \cdot x_i + \gamma(\beta_j) \cdot x_j \]  

(1)

where \( h_j \in \mathbb{R}^C \) is the interpolated point feature by weighting the point feature \( x_i \) and \( j \)-th point feature \( x_j \), where \( j \) enumerates the index of points in \( N_i \). Besides, \( \gamma(\cdot) \) is the sigmoid activation function. As a result, for the point feature \( x_i \), we can generate \( r \) new point features. Thus, we upsample the point cloud by \( h \) times. Following [23], we also use a 2D grid mechanism and a self-attention mechanism for better feature fusion.

In the experiment, we stack two NEUs to upsample the point cloud. After the first NEU, we first resize the feature size from \( rN \times C \) to \( N \times rC \) and then apply MLP to generate a new feature map with the size \( N \times C \). Consequently, we subtract the new feature map with the original feature map \( X \) and then feed it into the second NEU. Finally, we use the MLP network to generate the 3D coordinates of the upsampled point cloud.

3.2 Differentiable Rendering Unit

After generating the upsampled point cloud, we develop a differentiable rendering unit (DRU) to render the sparse and dense point clouds into multi-view images for computing the subsequent image-consistent loss. As shown in Figure 3, we illustrate the differentiable rendering unit. Specifically, we first place the virtual cameras at different viewpoints around the input point cloud to reconstruct the projection surfaces of the point cloud. Then we utilize the soft mesh rendering method [28] to render multi-view images from the projection surfaces.

To obtain the projection surfaces of the point cloud, we present an approximate approach to construct a local tangent triangular plane for each point. As shown in Figure 4 (a), given the point \( p_i \in \mathbb{R}^3 \) and the camera center \( o \in \mathbb{R}^3 \), we can obtain the projection vector \( x = p_i - o \) by subtracting the point \( p_i \) with the camera center \( o \). Then, we compute the vectors \( v_1, v_2, \) and \( v_3 \) for constructing the local tangent triangular plane. For a unit vector \( s = [1, 0, 0] \), we can compute the vector \( v_1 \) by:

\[ v_1 = x \times s \]  

(2)

In order to compute the vectors \( v_2 \) and \( v_3 \), we first calculate an auxiliary vector \( \hat{o} \) perpendicular to vectors \( s \) and \( v_1 \) by:

\[ \hat{o} = s \times v_1 \]  

(3)

After that, we can calculate the vector \( v_2 \) by solving the following equations:

\[
\begin{align*}
\langle v_2 \times v_1 \rangle \cdot \hat{o} &= 0 \\
\hat{o} \cdot v_2 &= \|v_2\| \|\hat{o}\| \cos 30° \\
\|v_2 \times \hat{o}\| &= \|v_2\| \|\hat{o}\| \sin 30° \\
\|v_1 \times v_2\| &= \|v_1\| \|v_2\| \cos 120°
\end{align*}
\]

(4)

Note that we can also compute the vector \( v_3 \) by replacing the vector \( v_2 \) in Equation (4) with the vector \( v_3 \). Finally, the angles between the vectors \( v_1, v_2, \) and \( v_3 \) are 120 degrees. Besides, we normalize the vectors \( v_1, v_2, \) and \( v_3 \) to unit vectors. As a result, we construct the tangent triangular plane for each point. Once we construct the local tangent triangular plane, we feed the constructed approximate surface to the differential mesh rendering method [28] to render the final images. Besides, Figure 4 (b) shows an example of rendering the approximate surface into an image.

3.3 Loss Functions

Shape-consistent loss. Generally, the Chamfer Distance (CD) or Earth Mover’s Distance (EMD) are used for the point cloud upsampling task. Due to the dense ground truth point clouds, they can work well in a supervised manner. Since the EMD requires that the number of points on the two input point clouds should be consistent, it cannot be applied to our self-supervised method. Compared to EMD, the CD does not require the input point clouds to have the same number of points. Nonetheless, due to the uneven distribution between the input sparse point cloud and the generated dense point cloud, the CD does not work well. Therefore, we propose a shape-consistent loss to make the shape of the dense point cloud consistent to the shape of the sparse point cloud. Specifically, we regard the sparse point cloud \( S \) as the result of downsampling the dense point cloud \( D \), so the downsampling result \( \hat{D} \) shares the common local geometric structures with \( D \). Given a downsampling function \( g(\cdot) \), we can obtain the downsampling point cloud \( \hat{D} \) from the dense point cloud \( D \). By minimizing the EMD between the input sparse point cloud \( S \) and the downsampling point cloud \( \hat{D} \), we enforce the dense point cloud to have the consistent shape with the sparse point cloud. The proposed shape-consistent loss is formulated as:

\[
L_{sc} = \min_{\psi : \hat{D} \rightarrow S} \sum_{d_i \in D} \|d_i - \psi(d_i)\|_F^2
\]  

(5)
where $\hat{D} = g(D)$. Note that there are rendering different choices for the downsampling function $g(\cdot)$, and we use the farthest point sampling (FPS) for simplicity in the experiment.

**Image-consistent loss.** After rendering the sparse and dense point clouds into multi-view images, we formulate an image-consistent loss to make the shapes of the input sparse point cloud and the generated dense point cloud consistent. Given the sparse point cloud $S$, we can obtain the rendered images $\{I_{d1}, I_{d2}, \ldots, I_{dm}\}$ at different camera poses, where $m$ is the number of the cameras. Similarly, we can obtain the rendered images $\{I_{d1}, I_{d2}, \ldots, I_{dn}\}$ of the dense point cloud $D$. As the dense point cloud has more points than the sparse point cloud, the rendered images of the dense point cloud have more triangular surfaces. Nonetheless, according to experimental results, this gap has few effects on upsampling. We encourage the shapes of the sparse and dense point clouds to be as consistent as possible, and formulate an image-consistent loss, which is written as:

$$L_{ic} = \frac{1}{m} \sum_{j=1}^{m} ||I_{sj} - I_{dj}||_{F}^{2}$$

(6)

where $I_{sj}$ and $I_{dj}$ are the rendered images of the sparse and dense point clouds, respectively.

The joint loss function to train the network is defined as:

$$L = \lambda_{sc}L_{sc} + \lambda_{ic}L_{ic} + \lambda_{hd}L_{hd} + \lambda_{un}L_{un}$$

(7)

where $L_{sc}$ and $L_{ic}$ are the proposed shape-consistent loss and the image-consistent loss, respectively. Note that $L_{hd}$ is the Hausdorff distance loss that is used to constrain the noise points and outliers, and $L_{un}$ is the uniform loss in PU-GAN [23] to make the generated point cloud as uniform as possible. For more details of the uniform loss, please refer to [23]. $\lambda_{sc}, \lambda_{ic}, \lambda_{hd}$ and $\lambda_{un}$ are hyperparameters.

4 EXPERIMENT

4.1 Dataset and Implementation Details

We collect datasets released by PU-Net [57], MPU [55], PU-GAN [23], and part of human body models in FAUST [3] to construct a large dataset, called SSPU-DataSet, which includes 227 3D models of different categories. We randomly select 191 models as the training set, and the remaining models as the test set. We use the same strategy as in MPU [55] to extract 195 patches per 3D object, so a total of 37,245 patches are obtained. Besides, we set 256 points per patch. For a fair comparison, we also conduct experiments on the dataset used in PU-GAN [23]. During testing, we use the Chamfer distance (CD), Hausdorff distance (HD), and point-to-surface (P2F) distance to evaluate the performance of different methods.

During training, we use data augmentation to prevent the network from overfitting. We use Adam [21] as the optimizer and set the learning rate to 0.001. In the experiment, we train the network for 30 epochs with a batch size of 28. In NEU, we set $k=4$ to construct the local neighborhoods of point clouds. The hyperparameters $\lambda_{sc}, \lambda_{ic}, \lambda_{hd}$, and $\lambda_{un}$ are set to 100, 30, 10, and 25, respectively. The whole experiments are conducted on a single 2080Ti GPU.

4.2 Results

**Quantitative results.** As shown in Table 1, we report the upsampling results of different methods on our SSPU-Dataset. Note that we have listed the results of two cases of inputting 2048 points and 4096 points, respectively. It can be seen that our SSPU-Net has achieved comparable results to those supervised methods such as MPU and PU-GAN. It is worth noting that our SSPU-Net (self-supervised) outperforms PU-Net (supervised) with all metrics. Although our method is self-supervised, our method considers the local geometric structures of the point cloud, while PU-Net does not.

We also conduct experiments on the dataset of PU-GAN [23] to demonstrate the effectiveness of our method. We show the evaluation results in Table 2. It can be found that the performance of our SSPU-Net is still superior to PU-Net, while the performance is comparable to PU-GAN and MPU. Note that the PU-GAN dataset is smaller than the collected SSPU-DataSet. Generally, self-supervised methods require a larger dataset to obtain better performance. The performance on those two datasets further demonstrates that our method can achieve good upsampling results in a self-supervised manner.

**Visual results.** In addition to the quantitative results, in Figure 5, we also provide the visualization results of the upsampling results and the surface reconstruction results. The first column (a) shows the input sparse point cloud (4069 points) and the corresponding surface reconstruction result, and the last column (c) shows the corresponding ground truth (16384 points). The columns (b), (c), (d) show the results of EAR, PU-GAN, and SSPU-Net (ours), respectively. It can be seen that our SSPU-Net can recover more details than other methods. For example, our method can more
Figure 5: Comparison results of point cloud upsampling (×4) and surface reconstruction results with different methods (b-d) from the input point clouds (a).

accurately recover the legs of the horse and arms of the statue. On the one hand, our method considers the local geometric structures of the point cloud through two NEUs. As claim in [15], EAR is limited by the parameter radius, which measures the size of the neighborhood. As PU-GAN does not consider the local geometric structures of the point cloud, it cannot accurately recover the detailed local structures. On the other hand, although our method is self-supervised, the shape-consistent loss and image-consistent loss provide sufficient supervision signals to train the network. In addition, the good upsampling results also lead to the accurate surface reconstruction results. The visualization and reconstruction results further demonstrate the effectiveness of our method.

4.3 Ablation Study

Neighbor expansion unit. To study the effect of the proposed NUE, we replace it with the corresponding structure (up-down-up expansion unit) in PU-GAN [23] to evaluate the performance on the SSPU-dataset. We report the results in Table 3. Note that "w/o NEU" in Table 3 denotes the corresponding results. Although the improvement of the NEU module is not so large in comparison to the consistency loss, the improvement of the NEU is remarkable. For example, with the NEU, the upsampling performance is improved by 1% and 2% in terms of the HD and P2F-mean metrics. The NEU exploits the local geometric structure of the sparse point cloud to adaptively interpolate neighboring points so that the local structures of point clouds can be preserved in the upsampling process. The up-down-up expansion unit uses feature duplication to interpolate points, so it cannot capture the local geometric structures of the point cloud.

Table 3: The evaluation results (×4) of different components of SSPU-Net on the SSPU-Dataset. Note that CD, HD, and P2F are multiplied by 10^3.

|                        | CD   | HD   | P2F-mean | P2F-std |
|------------------------|------|------|----------|---------|
| w/o NEU                | 0.35 | 4.38 | 6.63     | 7.70    |
| w/o shape-consistent loss | 2.31 | 27.74| 19.73    | 14.85   |
| w/o image-consistent loss | 1.11 | 23.98| 14.83    | 13.95   |
| full pipeline          | 0.29 | 3.10 | 4.29     | 7.29    |
Shape/Image-consistent loss. As our method is self-supervised, the shape-consistent loss and the image-consistent loss are regarded as the supervision signals to train our network. In Table 3, we report the evaluation results of discarding the shape-consistent loss or the image-consistent loss. As can be seen from the table, without the shape-consistent loss or the image-consistent loss, the performance will decrease significantly. The shape-consistent loss encourages the sparse and dense point clouds to have similar 3D shapes overall, while the image-consistent loss encourages the sparse and dense point cloud to have similar local geometric structures through multi-view images.

Different number of input points. To verify the effectiveness of SSPU-Net on different numbers of input points, we show the visualization results in Figure 6. Note that we only train the network on 2048 points, and directly test the trained model on different numbers of input points, including 512, 2048, and 4096 points. It can be seen that our method can produce good upsampling results in the cases of different numbers of input points. The experiments show that our method is robust to different densities of input point clouds.

Generalization ability. Furthermore, we also directly test the pre-trained model (trained on the SSPU-Dataset) on the outdoor KITTI dataset [11] to demonstrate the generalization ability of the proposed method. As shown in Figure 7, we visualize the upsampling results of the car and cyclist categories on the KITTI dataset. The upsampling results show that our SSPU-Net can still preserve the geometric details of the real scanned scenes. Thus, the upsampling results further demonstrate that our method has good generalization ability.

### 4.4 Classification

In this subsection, we perform the downstream classification task to demonstrate that the upsampling results can further improve the classification performance. Specifically, we use PointNet++ [38] to conduct the experiments on the ModelNet40 [51] dataset. We first use the pre-trained model to generate \( \times4 \) upsampling results (1024 points → 4096 points) on the ModelNet40 dataset. Then, we train the PointNet++ on the upsampled ModelNet40 dataset with the same experimental settings as the original PointNet++.

As shown in Table 4, we report the classification results using different point cloud upsampling methods, including SSPU-Net, PU-Net, and PU-GAN. Compared with the original PointNet++, our SSPU-Net can improve the classification results with the gain of 1.6% on the mean class accuracy. In addition, our method is superior to PU-Net and achieves good results, which are comparable to PU-GAN. The classification results demonstrate that our method can improve the performance of the downstream classification task.

| Method                        | mAcc | OA  |
|-------------------------------|------|-----|
| PointNet++                    | 86.2 | 90.1|
| PointNet++ (\( \times4 \) by PU-Net*) | 85.4 | 89.8|
| PointNet++ (\( \times4 \) by PU-GAN*) | 87.9 | 90.2|
| PointNet++ (\( \times4 \) by SSPU-Net*) | 87.8 | 90.6|

### 4.5 Retrieval

Here, we also conduct the experiments on the point cloud based place recognition task [16, 31, 47] to demonstrate that our method can still improve the downstream retrieval task. In the experiment, we use the classical PointNetVLAD [47] to perform the experiments on the Oxford RobotCar dataset [34]. Specifically, we first use the pre-trained model to generate \( \times2 \) upsampling results (4096 points → 8192 points) of the Oxford RobotCar dataset. Then, we use the upsampled data to train the PointNetVLAD and evaluate the retrieval results.

In Table 5, we list the retrieval results using different upsampling methods, including SSPU-Net, PU-Net, and PU-GAN. From this table, it can be seen that using upsampled data with SSPU-Net can effectively improve the performance of PointNetVLAD (dubbed as
Although our SSPU-Net is self-supervised, we can also yield comparable results to PU-GAN. In addition, since our SSPU-Net considers the local geometric structures of the point cloud, it can obtain better retrieval results than PU-Net.

Table 5: The retrieval results of the average recall (%) at top 1% (@1%) and at top 1 (@1) on the Oxford RobotCar dataset.

|          | Ave recall @1% | Ave recall @1 |
|----------|----------------|---------------|
| PN_VLAD  | 81.0           | 62.7          |
| PN_VLAD (x2 by PU-Net) | 81.4 | 62.9 |
| PN_VLAD (x2 by PU-GAN)  | 82.1 | 63.4 |
| PN_VLAD (x2 by SSPU-Net) | 82.0 | 63.5 |

5 CONCLUSION
In this paper, we proposed SSPU-Net, an end-to-end self-supervised point cloud upsampling network. The proposed neighbor expansion unit (NEU) can adaptively learn weights from the local geometric structures of the point cloud for point interpolation. The proposed differentiable rendering unit (DRU) can render the input point cloud into multi-view images, and also support end-to-end training. In order to train the network in a self-supervised manner, we proposed a shape-consistent loss and an image-consistent loss. The shape-consistent loss encourages the input sparse point cloud and the generated dense point cloud to have similar 3D shapes, and the image-consistent loss encourages the rendered images of the input and generated point clouds to have similar local geometric structures. Experimental results on the point cloud upsampling task and downstream classification and retrieval tasks demonstrate the effectiveness of our proposed self-supervised point cloud upsampling network.

ACKNOWLEDGMENTS
This work was supported by the National Science Fund of China under Grant 61876084.

REFERENCES
[1] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. 2003. Computing and rendering point set surfaces. IEEE Transactions on Visualization and Computer Graphics 9, 1 (2003), 3–15.
[2] Thomas Blanc, Mohamed El Beheiry, Jean-Baptiste Masson, and Bassam Hajj. 2020. Genmune: a virtual reality platform for visualizing and analyzing multidimensional point cloud data. bioRxiv (2020).
[3] Elizabeth R Boroson and Nora Ayanian. 2019. 3D Keypoint Repeatability for Heterogeneous Multi-Robot SLAM. In IEEE International Conference on Robotics and Automation. 6337–6343.
[4] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T Freeman. 2018. Unsupervised training for 3D morphable model generation. In Proceedings of the AAAI Conference on Artificial Intelligence. 1140–1147.
[5] Mingmei Cheng, Le Hui, Jin Xie, and Jian Yang. 2021. SSPC-Net: Semi-supervised multi-view learning for 3D meshes with only 2D supervision. arXiv preprint arXiv:2107.09259 (2021).
[6] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T Freeman. 2018. Unsupervised training for 3D morphable model generation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 8377–8386.
[7] Paul Henderson and Vittorio Ferrari. 2018. Learning to generate and reconstruct 3D meshes with only 2D supervision. arXiv preprint arXiv:1807.09259 (2018).
[8] Gergely Horváth and Gábor Erdős. 2017. Point cloud based robot cell calibration. CIRP Annals 66, 1 (2017), 145–148.
[9] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Boubekeur. 2018. Single-image svbrdf capture with a rendering-aware deep network. ACM Transactions on Graphics 37, 4 (2018), 1–15.
[10] Renyong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Ramfazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. 2019. Adversarial sensor attack on LiDAR-based perception in autonomous driving. In ACM SIGSAC Conference on Computer and Communications Security. 2267–2281.
[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2013. Vision meets robotics: The kitti dataset. The International Journal of Robotics Research 32, 11 (2013), 1231–1237.
[12] Thomas Blanc, Mohamed El Beheiry, Jean-Baptiste Masson, and Bassam Hajj. 2020. Genmune: a virtual reality platform for visualizing and analyzing multidimensional point cloud data. bioRxiv (2020).
[13] Thomas Blanc, Mohamed El Beheiry, Jean-Baptiste Masson, and Bassam Hajj. 2020. Genmune: a virtual reality platform for visualizing and analyzing multidimensional point cloud data. bioRxiv (2020).
[14] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Boubekeur. 2018. Single-image svbrdf capture with a rendering-aware deep network. ACM Transactions on Graphics 37, 4 (2018), 1–15.
[15] Mingmei Cheng, Le Hui, Jin Xie, and Jian Yang. 2021. SSPC-Net: Semi-supervised multi-view learning for 3D meshes with only 2D supervision. arXiv preprint arXiv:2107.09259 (2021).
