RepKPU: Point Cloud Upsampling with Kernel Point Representation and Deformation

Yi Rong, Haoran Zhou, Kang Xia, Cheng Mei, Jiahao Wang, Tong Lu*
State Key Laboratory for Novel Software Technology, Nanjing University
rongyi@smail.nju.edu.cn, hrzhou98@gmail.com, mf21330092@smail.nju.edu.cn
meicheng@smail.nju.edu.cn, wangjh@smail.nju.edu.cn, lutong@nju.edu.cn

Abstract

In this work, we present RepKPU, an efficient network for point cloud upsampling. We propose to promote upsampling performance by exploiting better shape representation and point generation strategy. Inspired by KPConv [47], we propose a novel representation called RepKPoints to effectively characterize the local geometry, whose advantages over prior representations are as follows: (1) density-sensitive; (2) large receptive fields; (3) position-adaptive, which makes RepKPoints a generalized form of previous representations. Moreover, we propose a novel paradigm, namely Kernel-to-Displacement generation, for point generation, where point cloud upsampling is reformulated as the deformation of kernel points. Specifically, we propose KP-Queries, which is a set of kernel points with predefined positions and learned features, to serve as the initial state of upsampling. Using cross-attention mechanisms, we achieve interactions between RepKPoints and KP-Queries, and subsequently KP-Queries are converted to displacement features, followed by a MLP to predict the new positions of KP-Queries which serve as the generated points. Extensive experimental results demonstrate that RepKPU outperforms state-of-the-art methods on several widely-used benchmark datasets with high efficiency. Codes will be available at https://github.com/EasyRy/RepKPU.

1. Introduction

Nowadays, 3D vision is in high demand for research thanks to the rapid development of 3D acquisition technology. Point cloud is the most common description of 3D data and is widely used in real-world applications [18, 34, 37, 49]. However, real-scanned point clouds are often noisy and sparse, limiting their further applications in downstream tasks, *e.g.*, classification, reconstruction, and segmentation. To alleviate this issue, various deep-learning algorithms for

*Corresponding author.

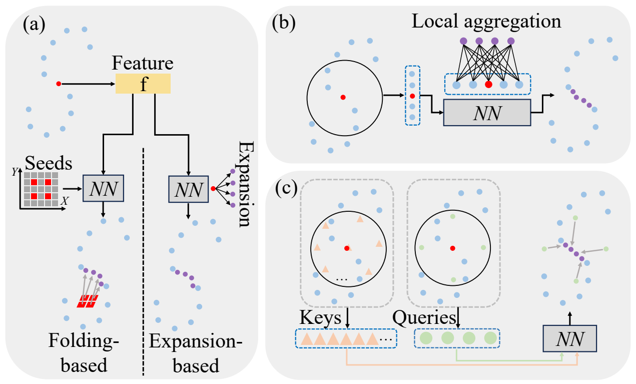

Figure 1. **Illustration of our main idea on 2D points.** Take $4\times$ upsampling task as an example: (a) Early approaches [38, 65, 68] use a feature vector (yellow) to represent the local region and generate points using folding or feature expansion operations. (b) Recent works [58, 75] adopt local aggregation operations for point upsampling, *i.e.*, new points are generated by summarizing local semantic information. (c) RepKPU encodes the local features into a set of kernel points (orange triangles) as the local shape representation, and lets this representation guide the deformation of another set of kernel points (green circles) for point generation.

point cloud upsampling have been developed [19, 38, 64, 65]. The goal of point cloud upsampling, as demonstrated in Figure 1, is to generate clean and dense point clouds from noisy and sparse inputs; therefore, upsampling models can be integrated into other approaches for related tasks, *e.g.*, point cloud completion [58, 75].

Existing upsampling methods can be principally categorized into two types: generation-based and refinement-based. The generation-based approaches [19, 38, 43, 65] are primarily divided into two steps: (1) Feature extraction: per-point geometric features are extracted in this phase; (2) Coordinate prediction: coordinates of new points are predicted base on the extracted features. However, generation-based methods suffer from outliers or shrinkage artifacts due to the difficulty in predicting 3D coordinates [12, 24]. The refinement-based approaches [4, 12, 20, 24, 27, 51] mainly consist of: (1) Coarsely generation: an upsampled coarse point cloud is generated by adopting simple interpolation or adding random noise; (2) Feature extraction: extracting features from the coarse point cloud; (3) Refinement: refining the point coordinates based on the distance function estimated from the extracted features. However, learning accurate distance functions (*e.g.*, SDFs [32] or UDFs [6]) from low-res and noisy point clouds is difficult [15, 22, 25], therefore, performance of these refinement-based approaches is still limited.

To alleviate the aforementioned problems, we propose a novel network, namely RepKPU, for point upsampling. RepKPU is primarily based on the proposed upsampling paradigm, namely *Kernel-to-Displacement*, as shown in Figure 1 (c). Our insight into promoting upsampling quality is to introduce better *shape representation* and *point generation strategy*, and in this paper, we show both can be achieved by exploiting kernel points [47]. By revisiting KPConv [47], we first summarize two key ingredients behind the success of KPConv and kernel points: (1) Abundant geometric information: different kernel points can represent distinct geometric patterns thanks to geometry-aware weight averaging and separated convolutional weights; (2) Flexible deformation: the positions of kernel points can adapt to local geometry, as revealed in deformable KPConv. Inspired by this, we reformulate point cloud upsampling as the process of kernel point deformation guided by kernel point representation.

Specifically, from the perspective of local shape representation, early works usually adopt a shape vector and formulate point upsampling as per-point splitting (see Figure 1 (a)). Some recent approaches [58, 64, 75] utilize the local patch (*i.e.*, the set of k-nearest neighbors), and subsequently generate points by aggregating patch's features (see Figure 1 (b)). Different from them, we propose kernel point representation (*i.e.*, RepKPoints) to effectively characterize the local geometry by encoding local features into RepKPoints (see Figure 1 (c)). Following the spirit of KPConv, RepKPoints is a set of kernel points centered on each input point, and each kernel point contains a position and feature to represent one certain geometric pattern. RepKPoints has the following advantages over prior local representations: (1) *Density-sensitive.* We use ball query [36] to search the set of neighbor points whose size is determined by local density. (2) *Large receptive fields.* We use a large searching radius for ball query; therefore, a large number of local points can be compressively encoded into a small number of kernel points. (3) *Position-adaptive.* The positions of RepKPoints can adapt to local geometry, like deformable KPConv, and if kernel points collapse to the center point or move to the corresponding neighbor points, RepKPoints will degrade to the feature vector or local patch.

From the perspective of point generation, common folding-based methods [61, 68] project 2D coordinates sampled from a 2D grid onto the surface with the guidance of extracted features (see Figure 1 (a)). Given upsampling rate r, feature expansion operations [38, 58, 64] split the per-point features by r times via MLPs or deconvolutions (see Figure 1 (a)). Unlike them, we devise Kernel-to-Displacement generation for upsampling. Specifically, we first extract kernel point queries (*i.e.,* KP-Queries) with predefined positions and learned features and then let KP-Queries be the queries of cross-attention transformer. Next, by utilizing RepKPoints as the keys and values of transformer, KP-Queries are converted to displacement features. Finally, the 3D coordinates are predicted via MLP from displacements features. Despite the difficulty of predicting 3D coordinates, RepKPU's elaborate designs allow it to generate high-quality point clouds with high efficiency (*e.g.*, 29% faster than the latest Grad-PU [12]). Moreover, we propose a parameter-free strategy to extend RepKPU for arbitrary-scale upsampling by exploiting existing techniques.

In conclusion, the contributions of our paper can be summarized as follows: (1) We propose RepKPoints, which is an efficient local representation and a generalized form of previous representations. (2) We devise a novel Kernel-to-Displacement paradigm for point cloud generation with KP-Queries. (3) We propose a parameter-free strategy to extend fix-scale upsampling methods to flexible-scale upsampling. (4) We propose RepKPU for point cloud upsampling, and it outperforms state-of-the-art methods on several widely-used benchmarks.

2. Related Work

Point cloud learning. The techniques for point cloud learning can be categorized into three types: multiview-based, voxel-based, and point-based. The former two types of works usually transform the irregular point clouds into regular representations by using multi-view projection [1, 11, 18, 45] or 3D voxelization [17, 28, 29, 44] and subsequently adopt 2D/3D CNNs. With the proposal of PointNet [35] and PointNet++ [36], point-based methods have became the mainstream. PointNet and PointNet++ are the pioneering point-based networks that adopt Set Abstraction operations to model relationships among points. Following their spirit, a number of SetAbstraction-based methods have been proposed [26, 39]. Some methods [21, 52, 72, 74] convert the local region to a graph, followed by graph convolution layers. Other methods [47, 56, 59, 60] define novel convolutional operators that apply directly to the 3D coordinates without quantization. Recently, attention-based methods [10, 31, 57, 66, 73] have achieved impressive success thanks to the inherent permutational invariance, where the whole point set or local region is converted to a sequence and the aggregation weights are adaptive. Inspired by pre-training strategies in the realm of 2D vision,

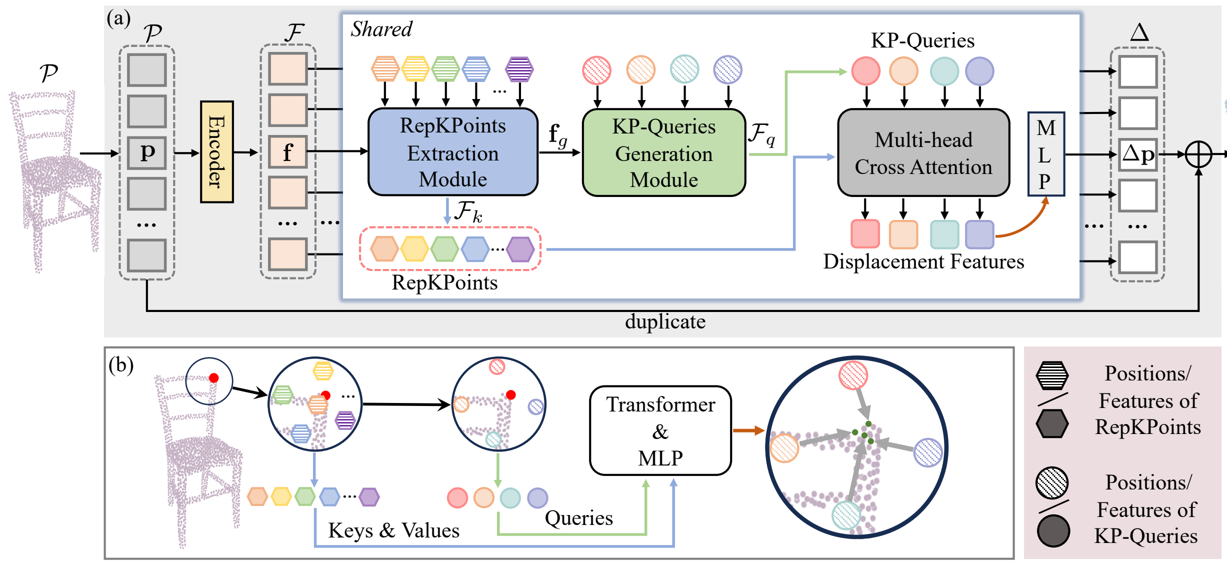

Figure 2. **Overview of RepKPU.** (a) We design *Kernel-to-Displacement* paradigm for point upsampling. Given per-point features $\mathcal{F}$ (extracted by encoder) and coordinates $\mathcal{P}$, we first feed them into RepKPoints Extraction Module (REM) to extract RepKPoints, and then adopt KP-Queries Generation Module (KGM) to generate KP-Queries. Next, we employ transformer implemented with multi-head cross-attention mechanisms [50] to convert KP-Queries to displacement features. Finally, displacement features are mapped to coordinates using MLP. (b) A visual illustration for point generation in RepKPU.

more and more pre-training methods have been proposed for point cloud [30, 53, 67, 70, 71]. Among all the aforementioned methods, we first adopt vector attention mechanisms [57, 73] to extract per-point features from sparse inputs, and then we propose Kernel-to-Displacement generation for point upsampling inspired by KPConv [47].

Learning-based point cloud upsampling/completion. Point cloud upsampling and completion are two similar tasks, and there exist many general techniques for point generation. Boosted by point-based deep networks, PCN [68] and PU-Net [65] are the first learning-based methods for point cloud completion and upsampling, respectively. Following them, progressive (or multi-stage) generation methods have sprung up [5, 14, 20, 43, 46, 54, 55, 58, 64, 75], especially in point cloud completion. Unlike common point generation, some methods try to use denoising-based paradigm for point upsampling, *i.e.*, a coarsely upsampled point cloud is generated by adding noise and then refined by a neural network [4, 12, 24, 27]. Arbitrary-scale (also called flexible-scale) upsampling [9, 12, 23, 40, 41, 62] is another trend that point clouds can be upsampled at any rate after training one time. In this paper, we show that RepKPU can be extended for arbitrary-scale upsampling with existing techniques without retraining.

By revisiting aforementioned methods, the most related upsampling methods can be categories into two types: (1) Single-point splitting (see Figure 1 (a)): points are generated by feature expansion (*e.g.*, duplicated-based deconvolutions [64, 65] or folding operations [61, 68]) with feature vectors. (2) Local aggregation generation (see Figure 1 (b)): points are generated via aggregating local geometric information, *e.g.*, SPD [58] and Upsample Transformer [75], where the local representation is the local patch. The upsampling strategy of RepKPU is different from these previous methods: we propose *adaptive* representation named RepKPoints and devise the Kernel-to-Displacement paradigm for upsampling, where the point generation is reformulated as the deformation of kernel points (see Figure 1 (c)).

3. Method

3.1. Revisiting KPConv [47] and Kernel Points

Applications of KPConv have achieved great success in many areas of point cloud [42, 47, 63], except point cloud upsampling. Kernel points are a set of points distributed in Euclidean space whose positions are denoted as $P_k \in \mathbb{R}^{N_k\times 3}$. KPConv first uses ball query [36] to search the local region of each input point with coordinate $\mathbf{p} \in \mathbb{R}^3$, and the local coordinates and corresponding features are $P_r \in \mathbb{R}^{N_r\times 3}$ and $F_r \in \mathbb{R}^{N_r\times C}$, respectively. Then, local features are projected onto kernel points via weight average:

$$\mathbf{f}_{k,i} = \sum_{j=0}^{N_r-1} \max(0, 1 - \frac{\|(\mathbf{p}_{r,j} - \mathbf{p}) - \mathbf{p}_{k,i}\|}{\sigma})\mathbf{f}_{r,j}, \quad (1)$$

where, $\mathbf{f}_{k,i} \in \mathbb{R}^C$ is the projected feature of i-th kernel point. $\mathbf{p}_{r,j} \in P_r$ and $\mathbf{p}_{k,i} \in P_k$ are position of j-th local point and i-th kernel point, respectively. Kernel points share a receptive field σ to shield the distant local points. After weight average, the local region with a large number of points (≈ 30) can be compressively encoded into a small number of kernel points (≤ 15).

Moreover, each kernel point has an independent filtering weight $\mathrm{W}_{k,i} \in \mathbb{R}^{C \times C'}$; therefore, different kernel points are sensitive to different geometric patterns. Finally, individual filtered features of all kernel points are aggregated for a local-aware feature vector $\mathbf{f}_{out} \in \mathbb{R}^{C'}$:

$$\mathbf{f}_{out} = \sum_{i=0}^{N_k-1} \mathbf{f}_{k,i} \mathrm{W}_{k,i}. \tag{2}$$

The above calculation process for KPConv can be formulated as follows:

$$\mathbf{f}_{out} = \mathrm{KPConv}(P_r, F_r, \mathbf{p}, P_k, W_k, \sigma). \tag{3}$$

Here, $W_k = \{\mathrm{W}_{k,i} | i = 0, 1, ..., N_k - 1\}$. The positions of kernel points in the rigid KPConv are fixed and uniformly located on the spherical surface (except the central one located at the origin point). In the case of deformable KPConv, the positions are learnable like [7]; therefore, kernel points can adaptively fit local geometry.

In summary, the key ingredients behind the success of KPConv and the feasibility of applying kernel points to the upsampling task are as follows: (1) Different kernel points can capture different local geometric patterns, and their set can finely represent the local geometry, naturally showing their potential to serve as shape representations. (2) Kernel points can adapt to local geometry with flexible deformations; therefore, kernel points are a competitive alternative for point generation compared with other methods.

3.2. Overview Pipeline

The overall pipeline of RepKPU is illustrated in Figure 2. $\mathcal{P} \in \mathbb{R}^{N \times 3}$ is the low-resolution point cloud, which is first fed into the encoder to extract per-point features $\mathcal{F} \in \mathbb{R}^{N \times C_e}$. Then, for each input point whose feature and coordinate are respectively $\mathbf{f} \in \mathcal{F}$ and $\mathbf{p} \in \mathcal{P}$, *RepKPoints Extraction Module* (REM) extracts kernel point representation (*i.e.*, RepKPoints) and local-aware feature $\mathbf{f}_g \in \mathbb{R}^{C_k}$ from its local region. The coordinates and features of RepKPoints are respectively denoted as $\mathcal{P}_k \in \mathbb{R}^{N_k \times 3}$ and $\mathcal{F}_k \in \mathbb{R}^{N_k \times C_k}$. And now, $\mathcal{F}$ is enhanced to $\mathcal{F}_g = \{\mathbf{f}_g\}$. Next, we employ *KP-Queries Generation Module* (KGM) to generate kernel point queries (*i.e.*, KP-Queries), whose coordinates and features are $\mathcal{P}_q \in \mathbb{R}^{N_q \times 3}$ and $\mathcal{F}_q \in \mathbb{R}^{N_q \times C_k}$, respectively. Note that $N_q = r$ and r is the upsampling rate. Finally, with the help of multi-head cross-attention mechanisms [50], KP-Queries can adaptively summarize the features of RepKPoints for displacement features $\mathcal{F}_d \in \mathbb{R}^{r \times C_d}$, followed by a MLP to predict offsets $\Delta \mathbf{p} \in \mathbb{R}^{r \times 3}$.

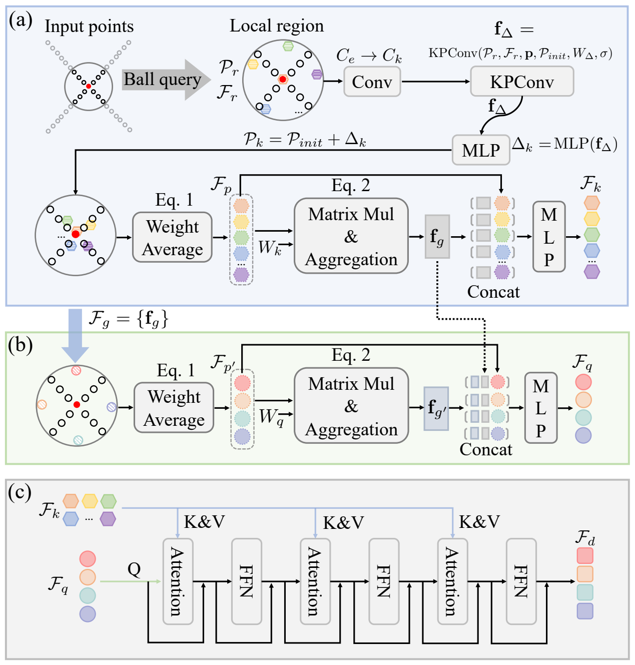

Figure 3. (a) Details of RepKPoints Extraction Module. (b) Details of KP-Queries Generation Module. (c) Details of Transformer implemented with multi-head cross-attention mechanisms.

The offsets can be seen as the new coordinates of KP-Queries relative to the center point. As all input points share the same modules, by duplicating $\mathcal{P}$ by r times and adding it with $\Delta = \{\Delta \mathbf{p}\}$, we can obtain the upsampled point cloud $\mathcal{P}_u \in \mathbb{R}^{(N \times r) \times 3}$.

Encoder. The purpose of encoder is to extract per-point features $\mathcal{F}$ from input coordinates. As our key designs are centralized in the upsampling phase, the encoder is not constrained. By comparing several alternatives as discussed in Section 4.7, we choose vector attention [73] as the fundamental operation and construct our encoder with dense connections [13]. Please refer to the supplementary material for complete details.

3.3. Kernel-to-Displacement Generation

We propose a novel and efficient paradigm for point cloud upsampling, namely Kernel-to-Displacement generation, as highlighted in Figure 2 (a).

RepKPoints Extraction Module. The details of this module are shown in Figure 3 (a). Following KPConv [47], the initial positions $\mathcal{P}_{init} \in \mathbb{R}^{N_k \times 3}$ of RepKPoints are uniformly located at a spherical surface with a radius of r_k, except the one at the origin. For each input point whose coordinate is $\mathbf{p}$ (red solid circle), we first adopt ball query to search its neighbor coordinates $\mathcal{P}_r \in \mathbb{R}^{N_r \times 3}$ and features $\mathcal{F}_r \in \mathbb{R}^{N_r \times C_e}$ under the searching radius of r_s.

Then, we use a linear layer to map the dim of $\mathcal{F}_r$ to C_k, and subsequently, we use KPConv and MLP to predict the offsets $\Delta_k \in \mathbb{R}^{N_k \times 3}$ of $\mathcal{P}_{init}$. With the adaptive positions $\mathcal{P}_k$, RepKPoints can adapt to local geometry. Consider the following two special cases: (1) RepKPoints degrades to the feature vector if all kernel points collapse to the origin; (2) RepKPoints degrades to a local patch if all kernel points correspond with local points. Therefore, RepKPoints is a generalized variant of previous representations.

Next, we project $\mathcal{F}_r$ onto RepKPoints for projected features $\mathcal{F}_p \in \mathbb{R}^{N_k \times C_k}$ according to Eq. 1. With learnable convolutional weights $W_k \in \mathbb{R}^{N_k \times C_k \times C_k}$, we aggregate projected features of kernel points to obtain $\mathbf{f}_g \in \mathbb{R}^{C_k}$ according to Eq. 2. Finally, using concatenation and MLP, we propagate $\mathbf{f}_g$ to $\mathcal{F}_p$ for features of RepKPoints, denoted as $\mathcal{F}_k = \{\mathbf{f}_{k,i} | i = 0, 1, ..., N_k - 1\}$:

$$\mathbf{f}_{k,i} = \text{MLP}(\text{cat}(\mathbf{f}_{p,i}, \mathbf{f}_g)), \mathbf{f}_{p,i} \in \mathcal{F}_p. \tag{4}$$

Following the practice of KPConv, we set $N_k = 15$, $\sigma = 0.2$, $r_s = 4\sigma$, $r_k = 1.5\sigma$, and the maximum number of neighbor points to 30. Therefore, RepKPoints has advantages as follows: (1) density-sensitive, thanks to ball query; (2) large receptive fields; (3) position-adaptive.

KP-Queries Generation Module.

As illustrated in Figure 3 (b), input features have been mapped from $\mathcal{F}$ to $\mathcal{F}_g = \{\mathbf{f}_g\}$ via RepKPoints Extraction Module. We first let the initial positions of KP-Queries be uniformly located at the spherical surface *without* the center one, and the positions are denoted as $\mathcal{P}_q$. Then the new local features $\mathcal{F}_g$ are projected onto predefined kernel points for projected features $\mathcal{F}_{p'} \in \mathbb{R}^{N_q \times C_k}$ according to Eq. 1. Next, according to Eq. 2, the local-aware feature $\mathbf{f}_{g'} \in \mathbb{R}^{C_k}$ is extracted with learnable matrix $W_q \in \mathbb{R}^{N_q \times C_k \times C_k}$. Finally, we concatenate several aforementioned feature vectors and feed the merged features into a MLP for $\mathcal{F}_q = \{\mathbf{f}_{q,i} | i = 0, 1, ..., N_q - 1\}$:

$$\mathbf{f}_{q,i} = \text{MLP}(\text{cat}(\mathbf{f}_{p',i}, \mathbf{f}_g, \mathbf{f}_{g'})), \mathbf{f}_{p',i} \in \mathcal{F}_{p'}. \tag{5}$$

Here, hyper-parameter setups (*i.e.*, σ, r_k, r_s and maximum number of neighbor points) are the same as with REM.

Cross-attention mechanisms. As shown in Figure 3 (c), we use a three-layer transformer-style architecture to convert $\mathcal{F}_q$ to $\mathcal{F}_d \in \mathbb{R}^{r \times C_d}$. Before feeding $\mathcal{F}_q$ and $\mathcal{F}_k$ into this module, we employ a linear layer to project the dim of them to C_d. The attention operations in Figure 3 (c) are the standard multi-head attention mechanisms [50], where we let $\mathcal{F}_q$ be Q, and $\mathcal{F}_k$ be K and V. We do not use any positional encodings as $\mathcal{F}_q$ and $\mathcal{F}_k$ already contain geometric information. In Figure 3 (c), FFN indicates a three-layer MLP. With cross-attention mechanisms, KP-Queries can adaptively summarize the geometric patterns of RepKPoints and be gradually converted to displacement features.

Coordinates prediction. As shown in Figure 2 (a), $\mathcal{F}_d$ is mapped to $\Delta_\mathbf{p} \in \mathbb{R}^{r \times 3}$ by a MLP with tanh as activation function. $\Delta_\mathbf{p}$ is the new local positions of KP-Queries. By duplication, upsampled point cloud $\mathcal{P}_u = \{\text{Dup}(\mathbf{p}) + \Delta_\mathbf{p}\}$ is generated, where Dup indicates duplication.

3.4. Training Loss

Given the ground truth point cloud $\mathcal{P}_{gt} \in \mathbb{R}^{N_g \times 3}$, we adopt Chamfer Distance loss ($\mathcal{L}_{cd}$) to supervise the upsampled point cloud $\mathcal{P}_u \in \mathbb{R}^{N_u \times 3}$. We also adopt the fitting and repulsive loss ($\mathcal{L}_{fit}$ and $\mathcal{L}_{rep}$) used in Deformable KPConv [47] to constrain the offsets Δ_k of RepKPoints. Please refer to our supplementary material for the details of $\mathcal{L}_{cd}$, $\mathcal{L}_{fit}$ and $\mathcal{L}_{rep}$. Therefore, the overall training loss is:

$$\mathcal{L} = \mathcal{L}_{cd} + \mathcal{L}_{fit} + \mathcal{L}_{rep}. \tag{6}$$

3.5. Extending to Arbitrary-Scale Upsampling

While our model is trained with fixed upsampling rate (*i.e.*, $4\times$), it can be easily extended to arbitrary-scale upsampling with existing techniques, and the proposed strategy is applicable for all fixed-scale methods. We name it Fixed-to-Arbitrary (F2A). Specifically, given input point cloud $\mathcal{P}_0 \in \mathbb{R}^{N_0 \times 3}$ and upsampling rate r, we first use mid-point interpolation [12] to generate intermediate point cloud $\mathcal{P}_1 \in \mathbb{R}^{N_1 \times 3}$ without learning, where $N_1 = \lceil \frac{N_0 \times r}{4} \rceil$. This intermediate point cloud is noisy and non-uniform. Then we use RepKPU to upsample and refine $\mathcal{P}_1$ for $\mathcal{P}_2 \in \mathbb{R}^{N_2 \times 3}$, where $N_0 \times r + 4 \geq N_2 = 4 \times \lceil \frac{N_0 \times r}{4} \rceil \geq N_0 \times r$. Finally, we random drop redundant points of $\mathcal{P}_2$ for $\mathcal{P}_3 \in \mathbb{R}^{(N_0 \times r) \times 3}$, and what we need is $\mathcal{P}_3$.

4. Experiments

4.1. Implementation Details

Experiment setup. All experiments of RepKPU are implemented with PyTorch [33]. We train our model on a single Nvidia 1080Ti GPU with Adam optimizer [16] and batch sizes of 32 for 100 epochs. The initial learning rate is set to $1e^{-3}$ and we decay it to $1e^{-4}$ gradually. We use the same data augmentation strategy as with previous methods [12, 20, 38]. The hyper-parameter settings of RepKPU are as follows: $C_e = 64$, $C_k = C_d = 128$, and the head number of attention mechanisms is 4. During evaluation, all models are evaluated on a single Nvidia 1080Ti GPU.

Evaluation Metrics. Following common practice, We adopt three evaluation metrics: Chamfer Distance (CD) [8], Hausdorff distance (HD) [2], and Point-to-surface Distance (P2F) [65].

4.2. Evaluation on PU-GAN Dataset

About PU-GAN dataset. PU-GAN dataset [19] contains 147 3D models collected from [64, 65]. Among them, 120 models are used for training and the rest for testing. By cropping 200 patches per training model and sampling

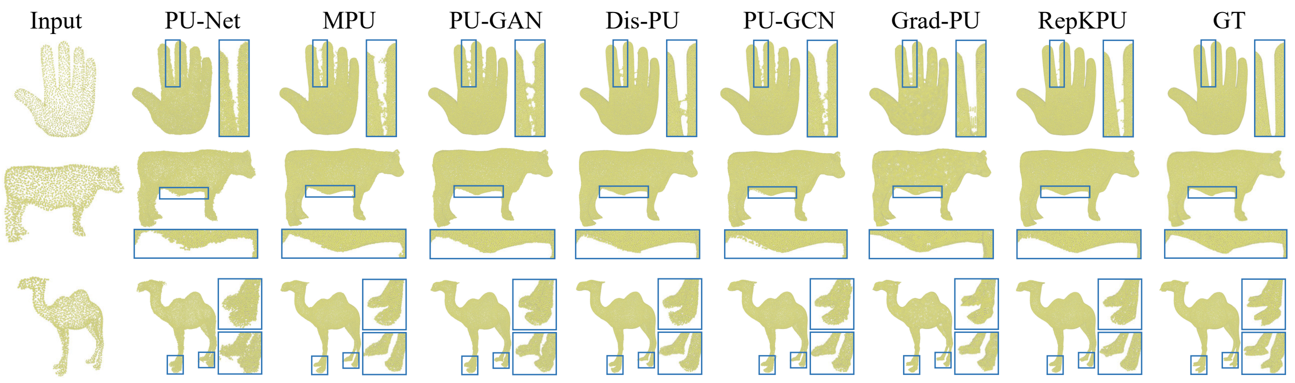

Figure 4. Visual comparison (16×) on PU-GAN dataset. Please zoom for better viewing.

Table 1. 4× and 16× quantitative results on PU-GAN dataset. We also report the inference speed in terms of seconds per point cloud (s/pc) and training memory usage. The best and second-best results are highlighted in bold and underline, respectively.

	4×				16×			
Methods	CD ↓ $\times 10^3$	HD ↓ $\times 10^3$	P2F ↓ $\times 10^3$	Speed ↓ s/pc	CD ↓ $\times 10^3$	HD ↓ $\times 10^3$	P2F ↓ $\times 10^3$	Training Memory
PU-Net [65]	0.444	3.931	4.578	0.314	0.508	6.508	5.867	7.596 G
MPU [64]	0.280	3.910	2.842	0.303	0.181	5.204	3.127	7.050 G
PU-GAN [19]	0.260	4.707	1.991	0.403	0.217	4.119	2.604	7.058 G
Dis-PU [20]	0.264	4.411	2.020	0.579	0.197	4.143	2.625	7.060 G
PU-GCN [38]	0.278	3.579	2.549	0.317	0.155	3.843	2.764	7.562 G
Grad-PU [12]	0.264	**2.623**	1.982	0.306	0.129	2.631	2.242	7.144 G
RepKPU	**0.248**	2.880	**1.906**	**0.215**	0.107	3.345	**2.068**	**5.860 G**
RepKPU w/ F2A	-	-	-	-	**0.104**	**1.975**	2.271	-

Table 2. 4× quantitative results on PU1K dataset. The best and second-best results are highlighted in bold and underline, respectively.

Methods	CD ↓ $\times 10^3$	HD ↓ $\times 10^2$	P2F ↓ $\times 10^3$
PU-Net [65]	0.665	0.693	5.747
MPU [64]	0.440	0.340	1.237
PU-GAN [19]	0.424	0.540	1.580
PU-GCN [38]	0.385	0.374	1.385
Dis-PU [20]	0.394	0.362	1.336
MAFU [41]	0.362	0.450	1.139
PU-Flow [27]	0.356	0.321	1.315
Grad-PU [12]	0.404	0.373	1.474
RepKPU	**0.327**	**0.268**	**0.938**

points from them using Poisson disk sampling [69], 24,000 training point clouds are produced in total. The resolution of an input patch is 256, and the resolution of corresponding ground truth is 1,024 (*i.e.*, 4× upsampling). Following common practice, input patches are augmented to imitate random and non-uniform inputs. However, the test set only has mesh files; therefore, we sample pairs of input and ground truth from original meshes by Poisson disk sampling implemented with Open3D [76]. Each test shape has 2,048 points, and the ground truth has 8,192 or 32,768 points, corresponding to 4× or 16× upsampling, respectively.

Evaluation setup. Following [19, 65], each test input is first cropped into several patches with 256 points, and then the upsampled patches are merged and downsampled to target size using FPS [36]. The common 16× upsampling is achieved by conducting 4× upsampling twice. We also report the results of the direct 16× upsampling results of RepKPU with our proposed F2A in Table 1. The training memory usage was evaluated with a batch size of 32.

Results. As the quantitative results reported in Table 1 show, RepKPU outperforms other methods in terms of CD and P2F, and it is second only to the latest Grad-PU in terms of HD. The visual comparisons are shown in Figure 4, from which we can find that there are holes and weird patterns in point clouds generated by Grad-PU. In contrast, our methods can generate more smooth and uniform point clouds. Significantly, RepKPU is 29% faster than the second-fastest MPU and takes 17% less GPU memory during training than MPU, which justifies the efficiency of RepKPU.

4.3. Evaluation on PU1K Dataset

About PU1K dataset. PU1K dataset [38] consists of 1,147 3D models split into 1,020 training samples and 127 testing samples. PU1K dataset is nearly 8 times larger than PU-GAN dataset; therefore, it is a more challenging benchmark dataset. Similar with [19], training set consists of 51,000 patches cropped from original models. The training input size and ground truth size are 256 and 1,024, respectively. The testing set has been provided in the form of point clouds; the resolution of the input is 2,048 and the resolution of the ground truth is 8,192.

Evaluation setup. Similarly, each complete testing point cloud is first divided into patches with 256 points, and then the result is produced via FPS. The early works [19, 38, 65] were trained with random inputs, while recent works [12, 27] favor uniform inputs. Therefore, following recent works, we use uniform inputs for training.

Table 3. Quantitative results of the robustness test. The best and second-best results are highlighted in bold and underline, respectively.

Methods	Average			Uniform inputs			Noisy inputs			Random inputs			Noisy + Random		
	CD ↓ $\times 10^3$	HD ↓ $\times 10^3$	P2F ↓ $\times 10^3$	CD ↓ $\times 10^3$	HD ↓ $\times 10^3$	P2F ↓ $\times 10^3$	CD ↓ $\times 10^3$	HD ↓ $\times 10^3$	P2F ↓ $\times 10^3$	CD ↓ $\times 10^3$	HD ↓ $\times 10^3$	P2F ↓ $\times 10^3$	CD ↓ $\times 10^3$	HD ↓ $\times 10^3$	P2F ↓ $\times 10^3$
PU-Net [65]	1.091	9.884	9.523	1.056	8.302	7.904	1.118	9.309	11.943	0.982	9.713	6.900	1.206	12.213	11.346
MPU [64]	0.839	9.630	6.731	0.699	7.042	4.847	0.827	9.253	8.266	0.822	10.064	5.079	1.006	12.159	8.732
PU-GAN [19]	0.812	9.823	7.252	0.668	7.268	3.851	0.912	9.925	10.082	**0.679**	9.446	4.332	0.990	12.651	10.744
Dis-PU [20]	0.745	8.891	5.802	0.628	6.908	3.694	0.762	9.033	7.397	0.701	8.608	3.976	0.889	11.017	8.139
PU-GCN [38]	0.827	8.382	6.478	0.690	6.012	4.462	0.821	7.845	8.082	0.804	8.655	4.851	0.992	11.016	8.517
Grad-PU [12]	0.950	9.612	**5.457**	0.640	**5.440**	3.598	0.800	**6.945**	**7.148**	1.124	12.586	**3.868**	1.233	13.480	**7.214**
RepKPU	**0.732**	**8.204**	5.556	**0.622**	5.996	**3.520**	**0.757**	7.762	7.262	0.692	**8.260**	3.878	**0.855**	**10.799**	7.567

Quantitative results. We report the quantitative results in Table 2. Whether previous methods adopt coordinate prediction [19, 38, 64] or denoising paradigm [12, 27], our method outperforms them by a large margin on all three metrics.

4.4. Robustness Test

About Dataset. Based on the testing meshes provided by PU-GAN, we construct a 4× upsampling dataset to conduct a comprehensive robustness test. Ground-truth point clouds are sampled from testing meshes via Poisson disk sampling. As shown in Table 3, the input point clouds have four patterns as follows: (1) *Uniform*: low-res inputs are sampled from meshes using Poisson disk sampling; (2) *Noisy*: inputs are added noise with level $\tau = 0.01$ after being uniformly sampled; (3) *Random*: random inputs are obtained by randomly selecting from ground-truth point clouds; (4) *Noisy + Random*: random inputs are added noise with level $\tau = 0.01$. Moreover, for each type of input pattern, we provide three different resolutions (*i.e.*, 512, 1,024, and 2,048) and report the average results of three resolutions. Therefore, this dataset contains 324 testing samples in total. Please refer to the supplementary material for more results about the robustness test.

Quantitative results. We report the quantitative results in Table 3. From the results, we can see that previous methods have defects as follows: (1) some methods fail when given certain input patterns; (2) it is hard to achieve triple wins on all three metrics. Specifically, PU-GAN achieves the best CD when given random inputs, while it fail in noisy cases; Grad-PU fails with random inputs. From the average metrics (highlighted in gray background) we can see, RepKPU outperforms other methods by a large margin. Although Dis-PU keeps up with RepKPU in terms of CD, it lags behind a lot in terms of HD. While PU-GCN achieves the second-best HD, it is not good enough in terms of CD and P2F. Therefore, RepKPU is a robust method under diverse input patterns and resolutions.

Table 4. Results of arbitrary-scale upsampling on PU-GAN dataset in terms of CD, HD and P2F.

Rates	Grad-PU [12]				RepKPU			
	CD ↓ $\times 10^3$	HD ↓ $\times 10^3$	P2F ↓ $\times 10^3$	Speed ↓ s/pc	CD ↓ $\times 10^3$	HD ↓ $\times 10^3$	P2F ↓ $\times 10^3$	Speed ↓ s/pc
5×	0.241	**2.521**	2.452	0.384	**0.221**	2.667	**1.867**	**0.290**
6×	0.227	**2.537**	2.904	0.501	**0.204**	2.595	**1.873**	**0.366**
7×	0.222	2.619	3.343	0.638	**0.170**	**2.229**	**1.775**	**0.452**
11×	0.222	3.720	4.953	1.338	**0.136**	**2.058**	**1.808**	**0.890**
15×	0.246	4.679	6.412	2.252	**0.107**	**2.162**	**2.063**	**1.483**
19×	0.278	5.852	7.701	3.395	**0.102**	**3.771**	**2.030**	**2.220**

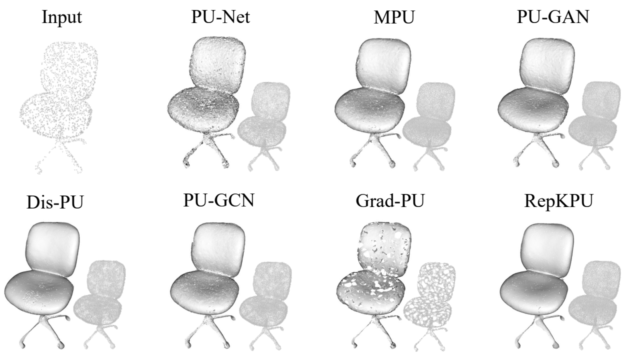

Figure 5. Visual evaluation of impacts on surface construction on real-scanned ScanObjectNN dataset.

4.5. Arbitrary-scale Upsampling

The results of arbitrary-scale upsampling are reported in Table 4. For each testing mesh in PU-GAN dataset, we first sample 2,048 points as input, then we sample 10,240 (5×), 12,288 (6×), 14,336 (7×), 22,528 (11×), 30,720 (15×), and 38,912 (19×) points as ground-truth point clouds, respectively. Note that, in the case of arbitrary-scale upsampling, inputs will be directly upsampled to target resolutions, which is different from the common 16× setup discussed in Section 4.2. The quantitative results strongly justify the effectiveness of RepKPU and F2A strategy.

4.6. Reconstruction from Real-scanned Data

We conduct experiments on ScanObjectNN dataset [48], where point clouds are real-scanned, non-uniform and

Table 5. $4\times$ upsampling results on PU1K dataset. RepKPoints: deformable kernel points or rigid ones?

Models	CD $\times 10^3 \downarrow$	HD $\times 10^2 \downarrow$	P2F $\times 10^3 \downarrow$
Rigid	0.345	0.286	1.118
Deformable	**0.327**	**0.268**	**0.938**

Table 6. $4\times$ comparative results on PU1K dataset with different upsampling approaches.

Models	Details	CD $\downarrow$ $\times 10^3$	HD $\downarrow$ $\times 10^2$	P2F $\downarrow$ $\times 10^3$
A	Baseline	0.365	0.301	1.263
B	Baseline + $\mathcal{F}_g$	0.346	0.281	1.124
C	Local patches	0.354	0.296	1.150
D	Local patches + $\mathcal{F}_g$	0.344	0.275	1.099
E	Folding queries	0.355	0.286	1.191
F	Learnable queries	0.348	0.282	1.095
G	Ours	**0.327**	**0.268**	**0.938**

noisy. Since there are no ground-truth point clouds, we show the visual comparisons in Figure 5. Given an input with 2,048 points, we first conduct $16\times$ upsampling, and then the surface is reconstructed from the upsampled point cloud using Ball-pivoting algorithm [3] implemented with Open3D. From the reconstruction results, we can clearly see that RepKPU can generate more smooth and uniform point clouds and subsequently benefit surface reconstruction.

4.7. Ablation Study

Deformable kernel points *vs.* rigid ones. Like KP-Conv [47], the positions of RepKPoints can be rigid if we drop the module that generates Δ_k (see Figure 3 (a)). The results are shown in Table 5, which indicates the importance of the deformable capability of RepKPoints.

Comparisons of several upsampling variants. As shown in Table 6, model A is our baseline which consists of the encoder and folding operation [68] based on $\mathcal{F}$. Model B adopts RepKPoints Extraction Module (REM) to extract $\mathcal{F}_g$, followed by folding operation, and the results show the superiority of the feature extraction capability of REM. From the perspective of local shape representation, we compare the local patch with our RepKPoints. Specifically, we let the local patch of each input point be the keys and values of transformer, and the neighbor size is set to 16. Model C drops REM and model D uses F_g extracted by REM. Compared with model A (or B), model C (or D) achieves limited improvements due to the redundant information in the local patch. We also discuss several designs of KP-Queries Generation Module (KGM). The folding queries of model E are generated by concatenating $\mathcal{F}_g$ with seeds sampled from 2D grid followed by a MLP for fusion. As to model F, rigid folding seeds in model E are replaced by a learnable matrix whose size is $N_q \times C_k$. All the other variants (A-F) illustrate the superiority of REM and KGM.

Table 7. Comparative results on PU1K with different feature extraction operations.

Models	Details	CD $\downarrow$ $\times 10^3$	HD $\downarrow$ $\times 10^2$	P2F $\downarrow$ $\times 10^3$
A	Graph Convolution [52]	0.334	0.279	1.220
B	Set Abstraction [36]	0.347	0.283	1.094
C	Vector Attention [73]	**0.327**	**0.268**	**0.938**

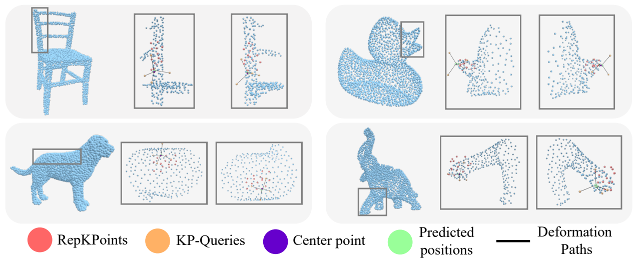

Figure 6. Visualization for Kernel-to-Displacement generation. For each 3D model, we select one patch and show each patch from two viewpoints.

Comparisons of feature extractor. As Table 7 shown, we provide comparative results about different encoder designs. After trying graph convolution [52] (GCN), Set Abstraction [36] (SA) and vector attention [73] (VA), we adopt VA as the fundamental operation of our encoder.

Visualization for Kernel-to-Displacement generation. As the point generation process shown in Figure 6, we can see that RepKPoints can fit local geometry, and KP-Queries can be properly deformed to surface with Kernel-to-Displacement generation.

5. Conclusion

We have proposed a novel network, namely RepKPU, for point cloud upsampling. There are two issues that limit the performance of existing methods: suboptimal shape representations and point generation approaches. To this end, we propose kernel point representation and Kernel-to-Displacement generation. Specifically, we first extract kernel point representation (RepKPoints) for each input point. Then, by reformulating upsampling as the deformation of kernel points, we adopt Kernel-to-Displacement to generate points. Justified by extensive experimental results, our proposed RepKPU shows superiority over prior approaches.

Acknowledgments

This work is supported by the the National Natural Science Foundation of China (Grant No. 62372223) and China Mobile Zijin Innovation Insititute (No. NR2310J7M).

References

[1] Mohamed Afham, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9902–9912, 2022. 2

[2] Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva. A benchmark for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(2):1–17, 2013. 5

[3] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. 8

[4] Haolan Chen, Shitong Luo, Wei Hu, et al. Deep point set resampling via gradient fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2913–2930, 2022. 1, 3

[5] Zhikai Chen, Fuchen Long, Zhaofan Qiu, Ting Yao, Wengang Zhou, Jiebo Luo, and Tao Mei. Anchorformer: Point cloud completion from discriminative nodes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13581–13590, 2023. 3

[6] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020. 2

[7] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 4

[8] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 5

[9] Wanquan Feng, Jin Li, Hongrui Cai, Xiaonan Luo, and Juyong Zhang. Neural points: Point cloud representation with neural fields for arbitrary upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18633–18642, 2022. 3

[10] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021. 2

[11] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1–11, 2021. 2

[12] Yun He, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. Grad-pu: Arbitrary-scale point cloud upsampling via gradient descent with learned distance functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2023. 1, 2, 3, 5, 6, 7

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 4

[14] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7662–7670, 2020. 3

[15] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. 2020 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[17] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE international conference on computer vision*, pages 863–872, 2017. 2

[18] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. 1, 2

[19] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: a point cloud upsampling adversarial network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7203–7212, 2019. 1, 5, 6, 7

[20] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Point cloud upsampling via disentangled refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 344–353, 2021. 1, 3, 5, 6, 7

[21] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Learning of 3d graph convolution networks for point cloud analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4212–4224, 2021. 2

[22] Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep implicit moving least-squares functions for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1788–1797, 2021. 2

[23] Luqing Luo, Lulu Tang, Wanyi Zhou, Shizheng Wang, and Zhi-Xin Yang. Pu-eva: An edge-vector based approximation solution for flexible-scale point cloud upsampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16208–16217, 2021. 3

[24] Shitong Luo and Wei Hu. Score-based point cloud denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4583–4592, 2021. 1, 3

[25] Baorui Ma, Yu-Shen Liu, and Zhizhong Han. Reconstructing surfaces for sparse point clouds with on-surface priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6315–6325, 2022. 2

[26] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022. 2

[27] Aihua Mao, Zihui Du, Junhui Hou, Yaqi Duan, Yong-jin Liu, and Ying He. Pu-flow: A point cloud upsampling network with normalizing flows. *IEEE Transactions on Visualization and Computer Graphics*, 2022. 1, 3, 6, 7

[28] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015. 2

[29] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. Vv-net: Voxel vae net with group convolutions for point cloud segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8500–8508, 2019. 2

[30] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision*, pages 604–621. Springer, 2022. 3

[31] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16949–16958, 2022. 2

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 2

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5

[34] François Pomerleau, Francis Colas, Roland Siegwart, et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4(1):1–104, 2015. 1

[35] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2

[36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2, 3, 6, 8

[37] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018. 1

[38] Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. Pu-gcn: Point cloud upsampling using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11683–11692, 2021. 1, 2, 5, 6, 7

[39] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35:23192–23204, 2022. 2

[40] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. Pugeo-net: A geometry-centric network for 3d point cloud upsampling. In *European conference on computer vision*, pages 752–769. Springer, 2020. 3

[41] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. Deep magnification-flexible upsampling over 3d point clouds. *IEEE Transactions on Image Processing*, 30:8354–8367, 2021. 3, 6

[42] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. Geometric transformer for fast and robust point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11143–11152, 2022. 3

[43] Shi Qiu, Saeed Anwar, and Nick Barnes. Pu-transformer: Point cloud upsampling transformer. In *Proceedings of the Asian Conference on Computer Vision*, pages 2475–2493, 2022. 1, 3

[44] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017. 2

[45] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 2

[46] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 383–392, 2019. 3

[47] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019. 1, 2, 3, 4, 5, 8

[48] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019. 7

[49] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic grasping. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2442–2447. IEEE, 2017. 1

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3, 4, 5

[51] Xiaogang Wang, Marcelo H Ang Jr, and Gim Hee Lee. Cascaded refinement network for point cloud completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 790–799, 2020. 1

[52] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019. 2, 8

[53] Ziyi Wang, Xumin Yu, Yongming Rao, Jie Zhou, and Jiwen Lu. Take-a-photo: 3d-to-2d generative pre-training of point cloud models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5640–5650, 2023. 3

[54] Xin Wen, Tianyang Li, Zhizhong Han, and Yu-Shen Liu. Point cloud completion by skip-attention network with hierarchical folding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1939–1948, 2020. 3

[55] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. Pmp-net: Point cloud completion by learning multi-step point moving paths. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7443–7452, 2021. 3

[56] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630, 2019. 2

[57] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35:33330–33342, 2022. 2, 3

[58] Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5499–5509, 2021. 1, 2, 3

[59] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3173–3182, 2021. 2

[60] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European conference on computer vision (ECCV)*, pages 87–102, 2018. 2

[61] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215, 2018. 2, 3

[62] Shuquan Ye, Dongdong Chen, Songfang Han, Ziyu Wan, and Jing Liao. Meta-pu: An arbitrary-scale upsampling network for point cloud. *IEEE transactions on visualization and computer graphics*, 28(9):3206–3218, 2021. 3

[63] Zi Jian Yew and Gim Hee Lee. Regtr: End-to-end point cloud correspondences with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6677–6686, 2022. 3

[64] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3d point set upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5958–5967, 2019. 1, 2, 3, 5, 6, 7

[65] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2790–2799, 2018. 1, 3, 5, 6, 7

[66] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12498–12507, 2021. 2

[67] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022. 3

[68] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 international conference on 3D vision (3DV)*, pages 728–737. IEEE, 2018. 1, 2, 3, 8

[69] Cem Yuksel. Sample elimination for generating poisson disk sample sets. In *Computer Graphics Forum*, pages 25–32. Wiley Online Library, 2015. 6

[70] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training. *Advances in neural information processing systems*, 35:27061–27074, 2022. 3

[71] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8552–8562, 2022. 3

[72] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5565–5573, 2019. 2

[73] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. 2, 3, 4, 8

[74] Haoran Zhou, Yidan Feng, Mingsheng Fang, Mingqiang Wei, Jing Qin, and Tong Lu. Adaptive graph convolution for point cloud analysis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4965–4974, 2021. 2

[75] Haoran Zhou, Yun Cao, Wenqing Chu, Junwei Zhu, Tong Lu, Ying Tai, and Chengjie Wang. Seedformer: Patch seeds based point cloud completion with upsample transformer. In *European conference on computer vision*, pages 416–432. Springer, 2022. 1, 2, 3

[76] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 6