PU-Net: Point Cloud Upsampling Network

Lequan Yu*[1,3], Xianzhi Li*[1], Chi-Wing Fu[1,3], Daniel Cohen-Or[2], and Pheng-Ann Heng[1,3]

[1]The Chinese University of Hong Kong, [2]Tel Aviv University, [3]SIAT, China

Introduction

Problem definition:

Point cloud upsampling: given a set of points, generate a denser set of points to describe the underlying geometry by learning the geometry of a training dataset.

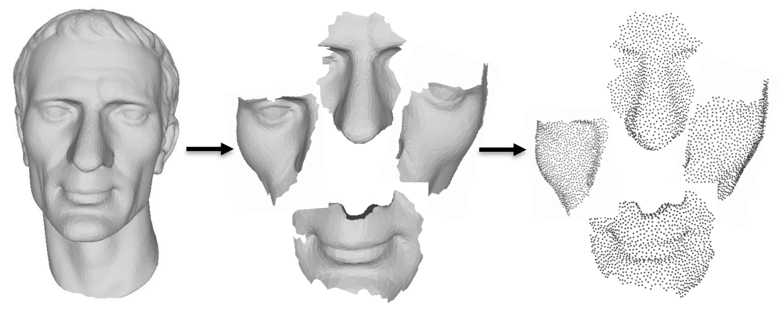
Challenges:

- Point clouds do not have any spatial order nor regular structure.
- The generated points should describe the underlying geometry.
- The generated points should be informative and should not clutter together.

Network Architecture

➢ **Patch extraction**

i. Randomly select points on the object surface.
ii. Grow a surface patch in a ring-by-ring manner.
iii. Generate points using Poisson disk sampling.



➢ **Point feature embedding**

i. Hierarchical feature learning (PointNet++).
ii. Multi-level feature aggregation.

➢ **Feature expansion**

Suppose the dimension of embedded feature f is $N \times \tilde{C}$, where N is the number of input points and $\tilde{C}$ is the feature dimension of the concatenated feature, the feature expansion operation can be represented as:

$$f' = \mathcal{RS}\left(\left[\mathcal{C}_1^2\left(\mathcal{C}_1^1(f)\right), \ldots, \mathcal{C}_r^2\left(\mathcal{C}_r^1(f)\right)\right]\right),$$

where $\mathcal{C}_i^1(\cdot)$ and $\mathcal{C}_i^2(\cdot)$ are two sets of separate 1×1 convolutions, $\mathcal{RS}(\cdot)$ is a reshape operation to convert an $N \times r\tilde{C}_2$ tensor to an $rN \times \tilde{C}_2$ tensor, and r is upsampling rate.

➢ **Coordinate reconstruction**

Reconstruct the 3D coordinates ($rN \times 3$) of output points from the expanded feature with the size of $rN \times \tilde{C}_2$ via a series of fully connected layers.

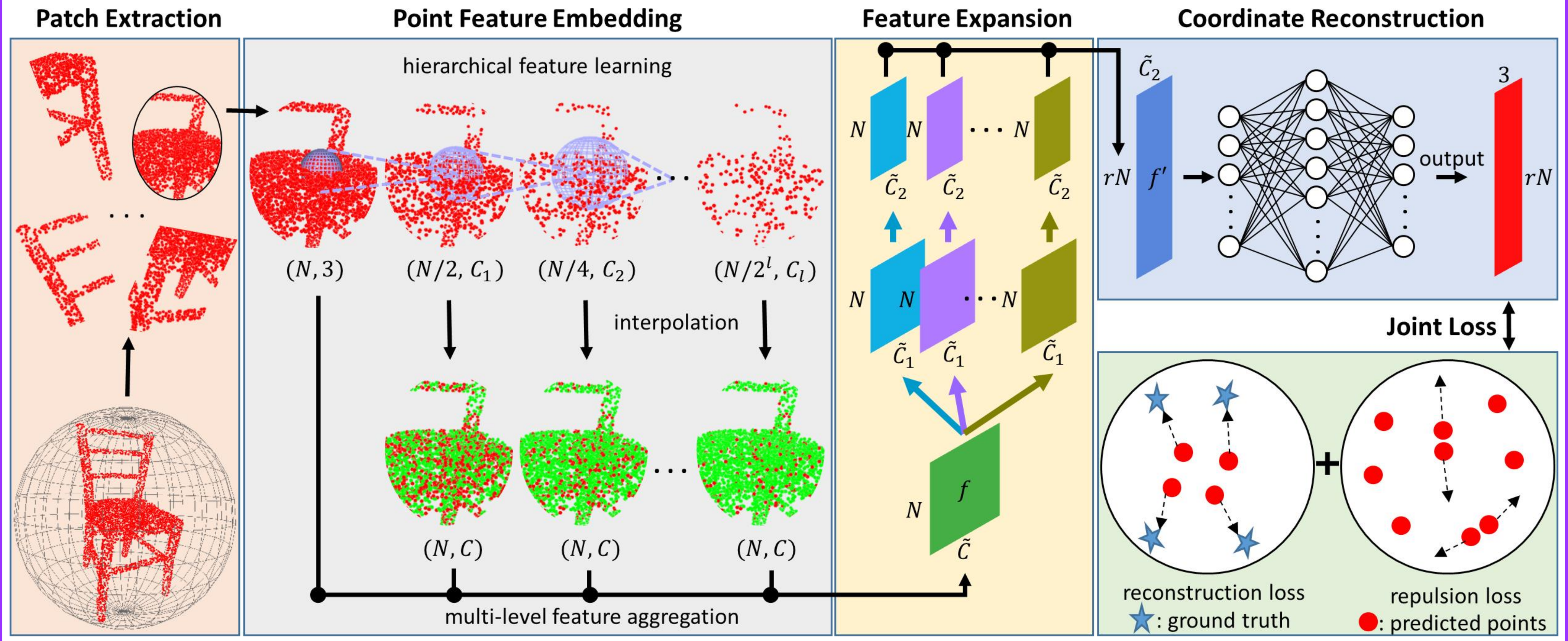
Joint Loss Function

➢ **Reconstruction loss**

To encourage the points to be located on the underlying object surfaces, we use the Earth Mover's distance as the reconstruction loss to evaluate the similarity between the predicted point cloud and the ground truth:

$$L_{rec} = d_{EMD}(S_p, S_{gt}) = \min_{\phi: S_p \to S_{gt}} \sum_{x_i \in S_p} \|x_i - \phi(x_i)\|_2,$$

where $\phi: S_p \to S_{gt}$ indicates a bijection mapping.



➢ **Repulsion loss**

To distribute the generated points more uniformly, we design the repulsion loss:

$$L_{rep} = \sum_{i=0}^{\hat{N}} \sum_{i' \in K(i)} \eta(\|x_{i'} - x_i\|) w(\|x_{i'} - x_i\|),$$

where $\hat{N} = rN$ is the number of output points, $K(i)$ is the index set of the k-nearest neighbors of point x_i, $\eta(r) = -r$ is the repulsion term, and $w(r) = e^{-r^2/h^2}$.

➢ **End-to-end training**

$$L(\boldsymbol{\theta}) = L_{rec} + \alpha L_{rep} + \beta \|\boldsymbol{\theta}\|^2$$

Experimental Results

➢ **Comparison with deep learning-based methods**

We designed some baseline methods for comparison based on the PointNet and PointNet++ architectures; see Figures 1 & 2.

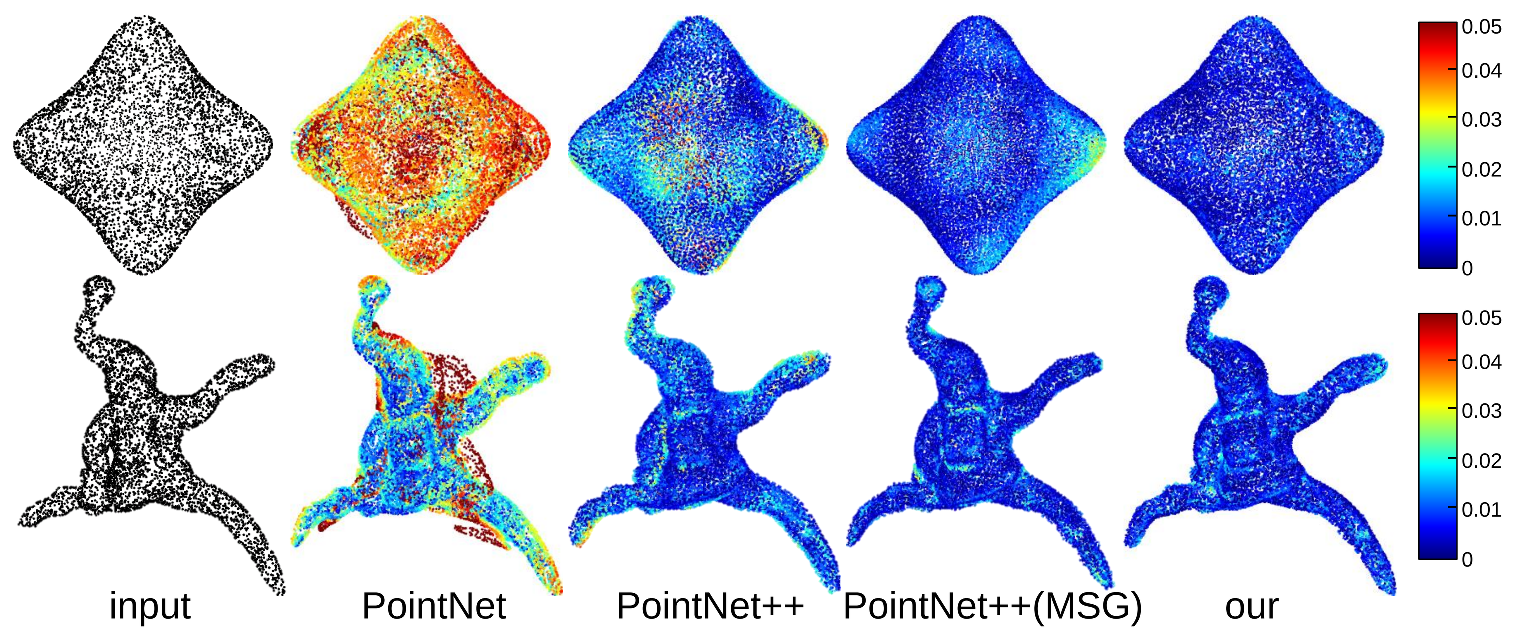


Fig.1 Comparison with other baseline methods. The colors reveal the surface distance errors.

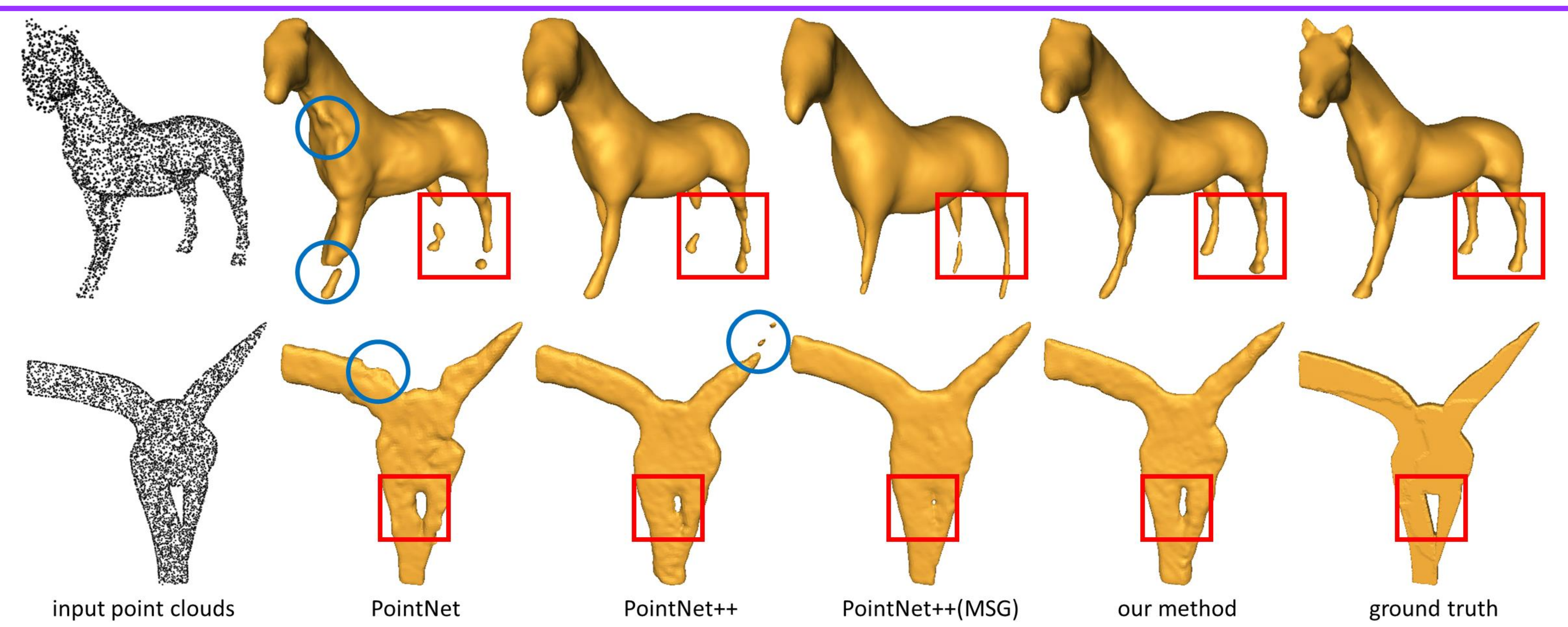


Fig.2 Surface reconstruction comparison with other baseline methods.

➢ **Results of iterative upsampling**

We design an iterative upsampling experiment, which takes the output of the previous iteration as the input to the next iteration. Figure 3 shows the results.

➢ **Results from noisy input point sets**

Figure 4 demonstrates the surface reconstruction results from the noisy point clouds, showing that our network facilitates surface reconstruction even from noisy inputs.

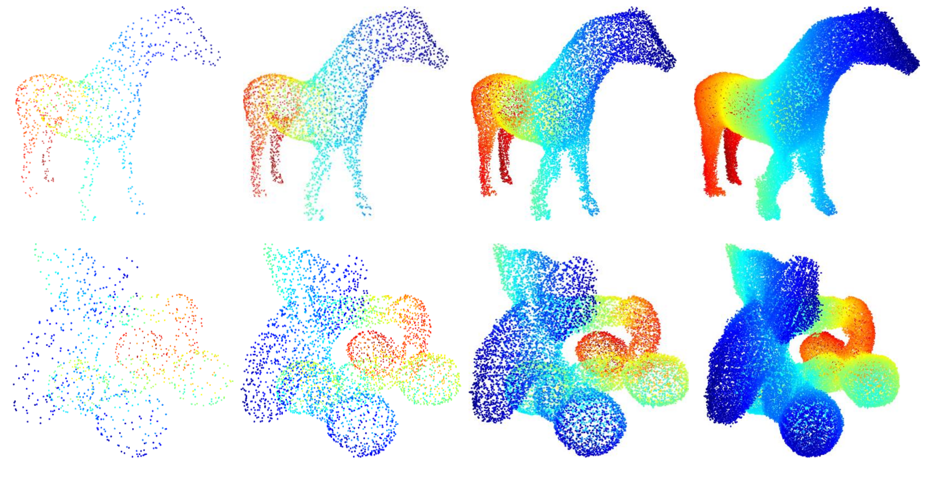


Fig.3 Results of iterative upsampling.

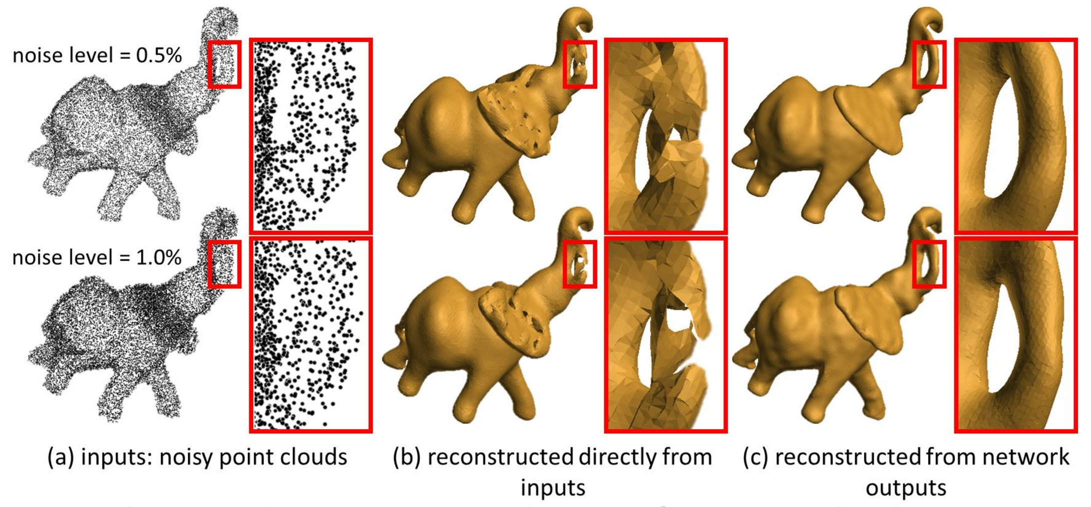


Fig.4 Surface reconstruction results from noisy inputs.

➢ **Quantitative comparison**

- Deviation: compute the mean and standard deviation of point-to-mesh distances.
- Normalized uniformity coefficient (**NUC**): put D equal-sized disks on the object surface and compute the standard deviation of the number of points inside the disks.

$$avg = \frac{1}{K \times D} \sum_{k=1}^{K} \sum_{i=1}^{D} \frac{n_i^k}{N^k \times p}$$

$$NUC = \sqrt{\frac{1}{K \times D} \sum_{k=1}^{K} \sum_{i=1}^{D} \left(\frac{n_i^k}{N^k \times p} - avg\right)^2}$$

K is the total number of objects, n_i^k is the number of points within the i-th disk of the k-th object, N^k is the total number of points on the k-th object, and p is the percentage of the disk area.

Table 1. Quantitative comparison.

Method	NUC with different p			Deviation (10^{-2})	
	0.2%	0.6%	1.0%	mean	std
Input	0.316	0.185	0.150	-	-
PointNet	0.409	0.295	0.252	2.27	3.18
PointNet++	0.215	0.160	0.143	1.01	0.83
PointNet++ (MSG)	0.208	0.152	0.137	0.78	0.61
PU-Net (Ours)	**0.174**	**0.122**	**0.112**	**0.63**	**0.53**

➢ Code is at https://github.com/yulequan/PU-Net

The work is supported in part by the National Basic Program of China, the 973 Program (Project No. 2015CB351706), the RGC of Hong Kong (Project no. CUHK 14225616), the Shenzhen Science and Technology Program (No. JCYJ20170413162617606), and the CUHK strategic recruitment fund.