PU-Net: Point Cloud Upsampling Network

Article · January 2018

CITATIONS
10

READS
251

5 authors, including:

Lequan Yu
The Chinese University of Hong Kong
44 PUBLICATIONS **1,135** CITATIONS
SEE PROFILE

Xianzhi Li
The Chinese University of Hong Kong
6 PUBLICATIONS **18** CITATIONS
SEE PROFILE

Chi-Wing Fu
Nanyang Technological University
115 PUBLICATIONS **1,416** CITATIONS
SEE PROFILE

PU-Net: Point Cloud Upsampling Network

Lequan Yu*[1] Xianzhi Li*[1] Chi-Wing Fu[1] Daniel Cohen-Or[2] Pheng-Ann Heng[1,3]
[1]The Chinese University of Hong Kong [2] Tel Aviv University
[3]Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
{lqyu,xzli,cwfu,pheng}@cse.cuhk.edu.hk dcor@mail.tau.ac.il

Abstract

Learning and analyzing 3D point cloud with deep networks is challenging due to the sparseness and irregularity of the data. In this paper, we present a data-driven point cloud upsampling technique. The key idea is to learn multi-level features per point, and then expanding them via a multi-branch convolution unit, to implicitly expand the point set in feature space. The expanded feature is then split to a multitude of features, which are then reconstructed to an upsampled point set. Our network is applied at a patch-level, with a joint loss function that encourages the upsampled points to remain on the underlying surface with a uniform distribution. We conduct various experiments using synthesis and scan data to evaluate our method and demonstrate its superiority over some baseline methods and an optimization-based method. The results show that our upsampled results have better uniformity, and sampled closer to the underlying surface.

1. Introduction

Point cloud is a fundamental 3D data representation that has drawn increasing attention due to the popularity of various depth scanning devices. Recently, pioneering works [24, 25, 15] began to explore the possibility of reasoning this 3D data representation by means of deep networks for understanding geometry and recognizing 3D structures. In these works, the deep networks directly extract features from the raw 3D point coordinates without involving traditional features (*e.g.*, normal and curvature). These works present impressive results for 3D object classification and semantic scene segmentation.

In this work we are interesting in an upsampling problem: Given a sparse set of points, generate a denser set of points to describe the underlying geometry by learning geometry of training dataset. This upsampling problem is similar in spirit to the image super-resolution problem [28, 17], however dealing with 3D points rather than a 2D grid of pixels makes pose some challenges.

First, unlike image space, which is represented with a regular grid, point clouds do not have any spatial order regular structure. Second, the generated points should describe the underlying geometry of a latent target object, meaning that they should roughly lie on the target object surface, relying a sparse set of input point only. Third, the generated points should be informative and should not clutter together near the input points. Having said that, the generated point set should be more uniform on the target object surface. Thus, simple interpolation between input points cannot produce satisfactory results.

To meet the above challenges, we present a data-driven point cloud upsampling technique. Our network is applied at a patch-level, with a joint loss function that encourages the upsampled points to remain on the underlying surface with a uniform distribution. The key idea is to learn multi-level features per point, and then expanding them via a multi-branch convolution unit, to implicitly expand the point set in feature space. The expanded feature is then split to a multitude of features, which are then reconstructed to an upsampled point set.

The presented network architecture, named PU-Net, learns geometry semantics of point-based patches from a set of 3D models, and applies the learned knowledge to upsample a given point cloud. It should be noted that unlike previous network-based techniques designed for 3D point sets [24, 25, 15], here the number of input and output points is not the same.

We formulate several metrics to quantitatively evaluate the uniformity and deviation of distance to underlying surfaces of the upsampled point set, and test our method using variety of synthetic and real-scanned data. We also evaluate and measure the performance of our method and compare it to baseline and state-of-the-art optimization-based method. Our results show that our upsampled results have better uniformity, and sampled closer to the underlying surface.

*Equal contribution.

In the following we first revised related works. Then in Section 2, our PU-Net architecture will be introduced in detail with four different components. We will further elaborate the patch-based training manner and introduce the novel joint loss function for end-to-end training in Section 3. The formulation of evaluation metrics and a variety of experiments will be presented in Section 4. Finally, Section 5 concludes the paper.

Related work. Below, we discuss two related research areas on processing point cloud data:

Optimization-based methods. An early work by Alexa *et al.* [3] upsamples a point set by interpolating points at vertices of a Voronoi diagram in the local tangent space, where the main purpose is to enhance the point set for rendering. Lipman *et al.* [20] present a locally optimal projection (LOP) operator for points resampling and surface reconstruction based on an L_1 median. The operator works well even when the input point set contains noise and outliers. Successively, Huang *et al.* [11] proposed an improved weighted LOP to address the point set density problem.

Although these works have demonstrated good results, they make a strong assumption that the underlying surface is smooth, thus restricting the method's scope. Then, Huang *et al.* [12] introduce an edge-aware point set resampling method by first resampling away from edges and then progressively approaching edges and corners. However, the quality of their results heavily relies on the accuracy of normals at given points and a careful tuning of parameters. It is worth mentioning that Wu *et al.* [29] propose a deep points representation method to fuse consolidation and completion into one coherent step. Since its main focus is on filling large holes, global smoothness is, however, not enforced, so the method is sensitive large noise. The above methods are not data-driven, thus heavily rely on priors.

Deep-learning-based methods. Points in a point cloud do not have any specific order nor follows any regular grid structure, hence only a few recent works adopt a deep learning model to directly process point clouds. Most existing works convert a 3D point cloud into some other 3D representations such as volumetric grids [23, 30, 26] and geometric graphs [4, 22] for point-based processing. Qi *et al.* [24, 25] firstly introduce a deep learning network on point clouds for classification and segmentation purposes. In particular, the PointNet++ makes use of a hierarchical feature learning architecture that can capture both local and global geometry context. In some other aspects, Klokov *et al.* [15] propose the Kd-network for unstructured point cloud recognition. The recent work of Fan *et al.* [8] is most related to us. They introduce a prediction network to regress a point cloud from a given single image. Since the input modality is an image, it can employ a convolutional neural network to extract features. Consequently, Lin *et al.* [19] present a pseudo-renderer, as an approximation of true rendering operation, to generate a 3D point cloud from a single image. Achlioptas *et al.* [2] introduce a deep auto-encoder network for processing 3D point clouds. To the best of our knowledge, there are no prior works focusing on the point cloud upsampling problem.

2. Network Architecture

We take a patch-based approach to train the network and learn geometry semantics. Given a sparse 3D point cloud with point coordinates in nonuniform distributions, our network aims to output a denser point cloud that follows the underlying surface of the target object while being uniform in distribution. Our network architecture (see Fig. 1) has four components: *patch extraction*, *point feature embedding*, *feature expansion*, and *coordinate reconstruction*. First, we extract patches of points in varying scales and distributions from a given set of prior 3D models (Sec. 2.1). Then, the *point feature embedding* component in the network maps the raw 3D coordinates to a feature space by hierarchical feature learning and multi-level feature aggregation (Sec. 2.2). After that, we expand the number of features using the *feature expansion* component (Sec. 2.3) and reconstruct the 3D coordinates of the output point cloud via a series of fully connected layers in the *coordinate reconstruction* component (Sec. 2.4).

2.1. Patch Extraction

We collect a set of 3D objects as prior information for training the network. These objects cover a rich variety of shapes, from smooth surface to shapes with sharp edges and corners. Essentially, for our network to upsample a point cloud, it should learn local geometry patterns on these objects. This motivates us to take a patch-based approach for training.

In detail, we randomly select M points on the surface of these object. From each selected point, we grow a surface patch on the object, such that any point on the patch is within a certain geodesic distance (d) from the selected point. Then, we use Poisson disk sampling to randomly generate $\hat{N}$ points on each patch as the referenced ground truth point distribution on the patch. In our upsampling task, both local and global context contribute to a smooth and uniform output. Hence, we set d with varying sizes, so that we can extract patches of points on the prior objects with varying scale and density.

2.2. Point Feature Embedding

To learn both local and global geometry context from the patches, we consider the following two feature learning strategies, whose benefits complement each other:

Hierarchical feature learning. Progressively capturing

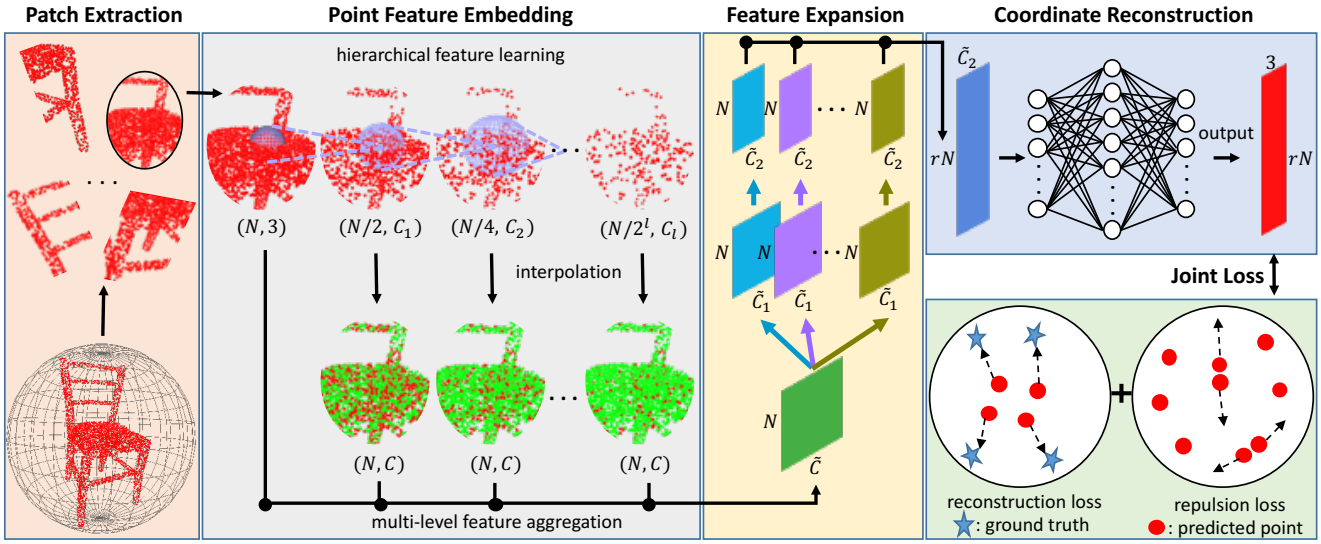

Figure 1. The architecture of PU-Net (better view in color version). The input has N points, while the output has rN points with r as the upsampling rate. C_i, $\tilde{C}$ and $\tilde{C}_i$ represent the feature channel number. We restore the different level features for the original N points with interpolation, and reducing all level features to a fixed dimension C with convolution. The red color in point feature embedding component shows the original and progressively subsampled points in hierarchical feature learning, while the green color indicates the restored features. We adopt the joint of reconstruction loss and repulsion loss for end-to-end training.

features of growing scales in a hierarchy has been proved to be an effective strategy for extracting local and global features. Hence, we adopt the recently proposed hierarchical feature learning mechanism in PointNet++ [25] as the very frontal part in our network. To adopt hierarchical feature learning for point cloud upsampling, we specifically use a relatively small grouping radius in each level, since generating new points usually involves more of the local context than the high-level recognition tasks in [25].

Multi-level feature aggregation. Lower layers in a network generally correspond to local features in smaller scales, and vice versa. For better upsampling results, we should aggregate features in different levels more optimally. Some previous works adopt skip-connections for cascaded multi-level feature aggregation [21, 27, 25]. However, we found by experiments that such top-down propagation is not efficient for aggregating features in our upsampling problem. Therefore, we propose to directly combine features from different levels and let the network learn the importance of each level [9, 32, 10].

Since the input point set on each patch (see point feature embedding in Fig. 1) is subsampled gradually in hierarchical feature extraction, we concatenate the point features from each level by first restoring the features of all original points from the downsampled point features by the interpolation method in PointNet++ [25]. Specifically, the features of an interpolated point x in level ℓ is calculated by:

$$f^{(\ell)}(x) = \frac{\sum_{i=1}^{3} w_i(x) f^{(\ell)}(x_i)}{\sum_{i=1}^{3} w_i(x)} \quad f^{(\ell)}(x) = 1 \qquad (1)$$

where $w_i(x)=1/d(x, x_i)$, which is an inverse distance weight, and x_i, x_2, x_3 are the three nearest neighbors of x in level ℓ. We then use a 1×1 convolution to reduce the interpolated feature in different level to a same dimension, say C. Finally, we concatenate the features from each level as f as the embedded point feature.

2.3. Feature Expansion

After the point feature embedding component in the network, we expand the number of features in the feature space. This is equivalent to expanding the number of points, since points and features can exchange from one another correspondingly. Suppose the dimension of f is $N \times \tilde{C}$, N is the number of input points and $\tilde{C}$ is the feature dimension of concatenated embeded feature. The *feature expansion* operation would output a feature f' with dimension as $rN \times \tilde{C}_2$, where r is the upsampling rate and $\tilde{C}_2$ is the new feature dimension. Essentially, this is similar to the feature upsampling in image-related tasks, which can be done by deconvolution [21] (also known as transposed convolution) or interpolation [7]. However, it is non-trivial to apply these operations to point cloud due to the non-regularity and unordered properties of point clouds.

We therefore propose an efficient *feature expansion* operation based on sub-pixel convolution layer [28]. This operation can be represented as:

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

where $\mathcal{C}(\cdot)$ is a convolution, and $\mathcal{RS}(\cdot)$ is a reshape operation to convert a $N \times r\tilde{C}_2$ tensor to a tensor of shape

$rN \times \tilde{C}_2$. We emphasize that the feature in the embedding space has already encapsulated the relative spatial information from neighborhood points via the efficient multi-level feature aggregation, and we do not need to explicitly consider the spatial information when doing feature expansion.

It is worth mentioning that the r feature sets generated from the first convolution $\mathcal{C}^1(\cdot)$ in each set have a high correlation, and this would cause the final reconstructed 3D points are very close to each other. Therefore, we further add another different convolution for each feature set. Since we train the network to learn the r different convolutions for the r feature sets, these new features can include different information and thus decrease the correlation among them. This *feature expansion* operation can be implemented by separated convolution on the r feature sets, as shown in Fig. 1. It can also be implemented by more computation efficient grouped convolution [16, 31, 33].

2.4. Coordinate Reconstruction

In this part, we reconstruct the 3D point coordinates of output point cloud from the expanded feature with the size of $rN \times \tilde{C}_2$. Specifically, we regress the 3D coordinates by applying a series of fully connected layers on the feature of each point, and the final output is the upsampled point coordinates $rN \times 3$.

3. End-to-End Network Training

3.1. Training Data Generation

Point cloud upsampling is actually an ill-posed problem due to the uncertainty or ambiguity of upsampled point clouds. Given a sparse input point cloud, there are many feasible output point distributions. Therefore, it is difficult or impossible to generate the "correct pairs" of input and the ground truth. To alleviate this problem, we propose to use an on-the-fly input generation scheme. Specifically, the referenced ground truth point distribution of a training patch is fixed, whereas the input points are randomly sampled with downsampling rate r from the ground truth point distribution at each training epoch. Intuitively, this scheme is equal to simulating many feasible output point distributions for a given sparse input point distribution. Additionally, this scheme can further enlarge the training dataset, allowing us to depend on a relatively small dataset for training.

3.2. Joint Loss Function

In order to train the network in an end-to-end fashion, we need to design an appropriate loss function to measure the quality of the output point cloud. As we mentioned above, we want to encourage the generated points following the underlying object surfaces while being more uniform in the distribution. Therefore, we propose a novel joint loss function by combining *reconstruction loss* and *repulsion loss* to encourage these two goals.

Reconstruction loss. To generate points on the underlying object surfaces, we propose to use Earth Mover's distance (EMD) [8] as our reconstruction loss to evaluate the similarity between the predicted point cloud $S_p \subseteq \mathbb{R}^3$ and the referenced ground truth point cloud $S_{gt} \subseteq \mathbb{R}^3$, which is denoted as follows:

$$L_{rec} = d_{EMD}(S_p, S_{gt}) = \min_{\phi: S_p \to S_{gt}} \sum_{x_i \in S_p} \|x_i - \phi(x_i)\|_2, \tag{3}$$

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

Actually, in [8], Chamfer Distance (CD) is also a candidate to evaluate the similarity between two sets. Compared with CD, EMD can better capture the shape (readers may refer to [8] for more details), so that it can encourage the output points following the underlying object surface as much as possible. Therefore, we choose to use EMD as our reconstruction loss.

Repulsion loss. Although training with reconstruction loss can generate points on the underlying object surfaces, the generated points tend to be too close to the original points. In order to make the generated points have a more uniform distribution, we add another repulsion loss into our loss function, which is represented as:

$$L_{rep} = \sum_{i=0}^{\hat{N}} \sum_{i' \in K(i)} \eta(\|x_{i'} - x_i\|) w(\|x_{i'} - x_i\|), \tag{4}$$

where $\hat{N} = rN$ is the number of output points, $K(i)$ is the set of k-nearest neighborhood of point x_i, $\|\cdot\|$ is the L2-norm, and $\eta(r) = -r$, called the *repulsion term*, is a decreasing function penalizing point x_i that gets too close to other points in $K(i)$. Since we only want to penalize x_i when it is close to its neighborhood points, we thus add two restrictions: the first one is that we only consider the points $x_{i'}$ which is in the k-nearest neighborhood of points x_i; the other is we add the fast-decaying weight function $w(r)$ into the repulsion loss. We set $w(r) = e^{-r^2/h^2}$ in our experiments following [20, 11].

Finally, we train the network to minimize the following loss function:

$$L(\boldsymbol{\theta}) = L_{rec} + \alpha L_{rep} + \beta \|\boldsymbol{\theta}\|^2, \tag{5}$$

where $\boldsymbol{\theta}$ indicates the parameters in our network, α balances the reconstruction loss and repulsion loss, and β denotes the multiplier of weight decay. For simplification, we ignore the index of each training sample.

4. Experiments

4.1. Datasets

Since there is no public benchmark dataset for point cloud upsampling, we collect a dataset from Visionair repository [1] with 60 different category models ranging from smooth non-rigid objects (*e.g.*, Bunny) to steep rigid objects (*e.g.*, Chair). We randomly select 40 objects for training, and 20 objects for testing [1]. We crop 100 patches for each training object, and we use $M = 4000$ patches to train the network in total. For testing objects, we use Monte Carlo random sampling approach to sample 5000 points on each object as input. To further demonstrate the generalization ability of our network, we directly test our well-trained network on SHREC15 [18] dataset. This dataset contains 1200 shapes from 50 categories. Since each category contains 24 similar objects with various poses, we randomly choose one model from each category for testing. As for ModelNet40 [30] and ShapeNet [5], it is a bit difficult to cut patches from those objects due to the low mesh quality (*e.g.*, holes, self-intersection). Therefore, we only use them for testing and results can be found in the supplementary material.

4.2. Implementation Details

The default point number $\hat{N}$ of each patch is 4096, and the upsampling rate r is 4. Therefore, each input data has 1024 points. To avoid overfitting, we augment the data by randomly rotating, shifting and scaling. We use 4 levels with grouping radius as 0.05, 0.1, 0.2 and 0.3 in the point feature embedding component, and the dimension C of the restored feature is 64. For other detailed network architecture parameters, please see our supplementary material. The k and h in repulsion loss are set as 5 and 0.03, respectively. The balancing weights are set as follows: α = 0.01 and β = 10^{-5}. The implementation is based on TensorFlow[2]. For optimization, we train the network using Adam [14] algorithm and minibatch size of 28. The learning rate is 0.001 and we train 120 epoch. Generally, it takes about 4.5h to train the network on a NVIDIA TITAN Xp GPU.

4.3. Evaluation Metric

To quantitatively evaluate the quality of output, we formulate two metrics to measure deviation between generated points and the ground truth mesh as well as uniformity of points distribution. For surface deviation, specifically, we find the closest point $\hat{x}_i$ on the mesh for each predicted point x_i, and calculate the distance between them. Then we use mean and standard deviation over all points as one of our metrics.

[1]The complete object list can be found in the supplementary material.

[2] https://github.com/yulequan/PU-Net

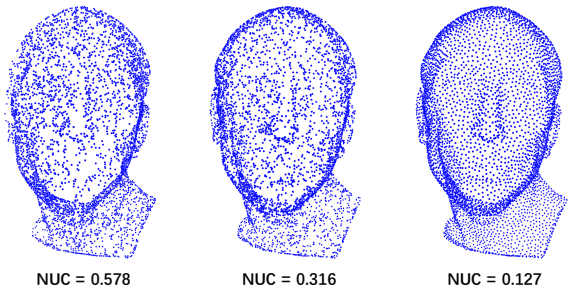

Figure 2. Some examples with different normalized uniformity coefficient. The numbers indicates the normalized uniformity coefficient with $p = 0.2\%$.

As for uniformity metric, we randomly put D equal-size disks on the object surface ($D = 9000$ in our experiments) and calculate the standard deviation of number of points in each disk. We further normalize the density of each object in order to evaluate the uniformity of all testing dataset result. Therefore, we define the *normalized uniformity coefficient* (NUC) with disk area percentage p as:

$$\begin{aligned} avg &= \frac{1}{K*D}\sum_{k=1}^{K}\sum_{i=1}^{D}\frac{n_i^k}{N^k*p}, \\ NUC &= \sqrt{\frac{1}{K*D}\sum_{k=1}^{K}\sum_{i=1}^{D}(\frac{n_i^k}{N^k*p}-avg)^2}, \end{aligned} \tag{6}$$

where n_i^k is the number of points within the i-th disk of object k, N^k is the total number of points on object k, K is the total number of test objects, and p is the percentage of disk area to the total surface area of the object. Note that we use geodesic distance not Euclidean distance to count the number of points in each disk. Fig. 2 shows some different point distributions and the corresponding NUC values. As we can see, the proposed NUC metric is in accordance with human visual comparison.

4.4. Comparisons with Other Methods

Comparison with the optimization-based method. Since Edge Aware Resampling (EAR) approach [12] is the state-of-art method for point cloud upsampling, we compare our method with this method. The results are shown in Fig. 3, where the Chair is from our collected testing dataset and the Spider is from SHREC15 dataset. We color-code the point clouds to show the deviation from ground truth mesh. There are 1024 points in the input and we do 4X upsampling. Since the EAR approach relies on initial normal input, to be fair, we calculate the normal according to ground truth mesh. We show two results of EAR method with increasing radius and the other parameters are set as default ones. As we can see, the radius parameter has a great influence to the performance in EAR method. For relatively small radius,

Table 1. Quantitative comparison on our collected dataset.

Method	NUC with different p						Deviation (10^{-2})		Time (s)
	0.2%	0.4%	0.6%	0.8%	1.0%	1.2%	mean	std	
Input	0.316	0.224	0.185	0.164	0.150	0.142	-	-	-
PointNet [24]	0.409	0.334	0.295	0.270	0.252	0.239	2.27	3.18	**0.05**
PointNet++ [25]	0.215	0.178	0.160	0.150	0.143	0.139	1.01	0.83	0.16
PointNet++(MSG) [25]	0.208	0.169	0.152	0.143	0.137	0.134	0.78	0.61	0.45
PU-Net (Ours)	**0.174**	**0.138**	**0.122**	**0.115**	**0.112**	**0.110**	**0.63**	**0.53**	0.15

Table 2. Quantitative comparison on SHREC15 dataset.

Method	NUC with different p						Deviation (10^{-2})		Time (s)
	0.2%	0.4%	0.6%	0.8%	1.0%	1.2%	mean	std	
Input	0.315	0.222	0.184	0.165	0.153	0.146	-	-	-
PointNet [24]	0.490	0.405	0.360	0.330	0.309	0.293	2.03	2.94	**0.05**
PointNet++ [25]	0.259	0.218	0.197	0.185	0.176	0.170	0.88	0.75	0.16
PointNet++(MSG) [25]	0.250	0.199	0.175	0.161	0.153	0.148	0.69	0.59	0.45
PU-Net (Ours)	**0.219**	**0.173**	**0.154**	**0.144**	**0.140**	**0.137**	**0.52**	**0.45**	0.15

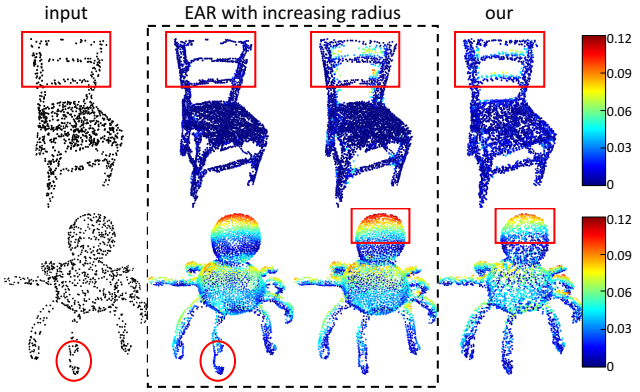

Figure 3. Comparison with EAR method. We color-code all point clouds to show the deviation from ground truth mesh. For EAR method, the radius of Chair model is 0.1 and 0.182; the radius of Spider model is 0.106 and 0.155.

the output has low surface deviation but the added points are not uniform, while more outliers will be introduced if radius is large. As a contrast, our method can better balance the deviation and uniformity without carefully parameter tuning. This comparison shows that the optimization-based EAR method has no ability to utilize object semantic information for upsampling, but this kind of information is very important for generating meaningful points.

Comparison with deep learning-based methods. As far as we know, there is no related deep learning-based approach for point cloud upsampling, and thus we design several baseline methods for comparison.

Since PointNet [24] and PointNet++ [25] are pioneers for 3D point cloud reasoning with deep learning technique, we design our baselines based on them. However, these two networks are designed for high level point cloud recognition problems rather than upsampling, therefore we adopt their semantic segmentation network architecture for point feature embedding and use one set of convolutions to do feature expansion. Note that there are two versions of PointNet++: basic PointNet++ and PointNet++ with multi-scale grouping (MSG) for handling non-uniform sampling density, therefore, we totally have three baselines and we train them only with reconstruction loss. Although we modify PointNet, PointNet++ and PointNet++(MSG) architectures to fit the upsampling problem, for convenience, we still denote them as the original architecture name.

Table 1 and Table 2 list the quantitative comparison on our collected dataset and SHREC15, respectively. The unit of deviation mean and std is 10^{-2}. Note that the NUC with small p can demonstrate local uniformity, while NUC with large p can demonstrate global uniformity. Among those baselines, PointNet produces the worst results because it cannot capture local structure information. Compared with PointNet++, the PointNet++(MSG) can slightly improve the uniformity due to the explicit multi-scale information grouping. However, this version introduces more parameters and significantly increase the training and testing time. Overall, our PU-Net achieves the best performance of both deviation and uniformity among all the other methods, especially on the local uniformity, demonstrating the effectiveness of our network.

We also show some examples for visual comparison as shown in Fig. 4. We colorize the points with surface distance deviation. As we can see, the predicted point clouds

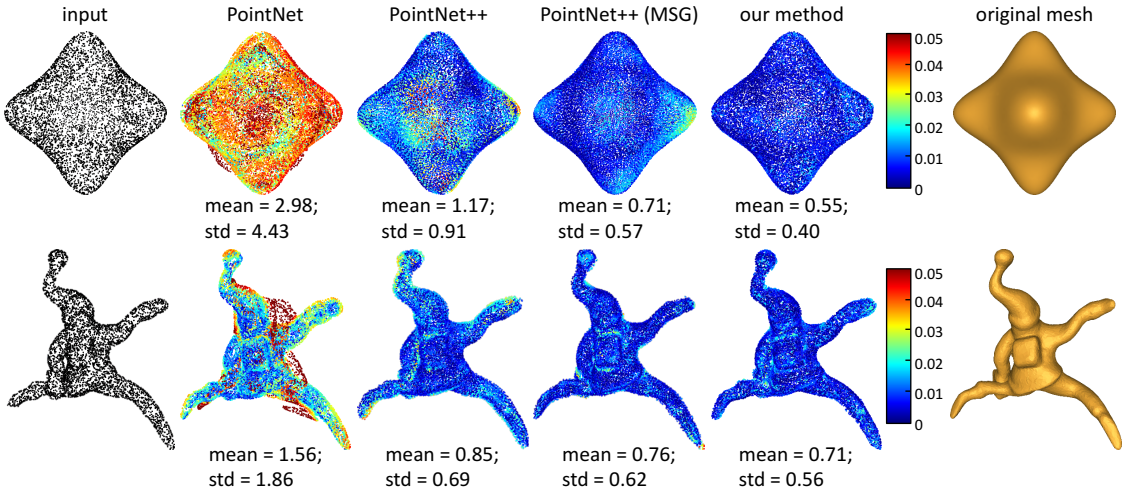

Figure 4. Visual comparison on our collected dataset (1st) and SHREC (2nd). Shapes are colored according to the surface error distance of each point, where the blue is low error and the red is high error.

Table 3. Architecture design analysis on our collect dataset.

Methods	NUC with different p				Deviation (10^{-2})	
	0.4%	0.6%	0.8%	1.0%	mean	std
interp1+EMD	0.153	0.136	0.126	0.121	0.67	0.54
interp2+EMD	0.144	0.129	0.122	0.118	0.71	0.57
CD	0.185	0.167	0.154	0.147	0.87	0.69
EMD	0.140	0.126	0.119	0.116	0.68	0.58
Ours	**0.138**	**0.122**	**0.115**	**0.112**	**0.63**	**0.53**

by our method can better describe the underlying surface with smaller surface distance error.

4.5. Architecture Design Analysis

Analysis of different feature expansion. We compare our proposed feature expansion approach with two interpolation-like approaches. The first one is similar to naive point interpolation idea (note as *interp1*). After extracting the point features, we find the nearest neighborhoods of each point. Then we combine its own features and the features from nearest points to generate the upsampled features. The second one introduces more randomness (note as *interp2*). Instead of using the features from the nearest neighbors, we use a radius based ball query approach to find the neighborhood and combine the features from these points to generate new point features. We train the two networks with reconstruction loss (also named EMD loss) and the performance on our collected data is list in the top two rows of Table 3. For fair comparison, we also train our network with EMD loss, and results are shown in the fourth row. When comparing with these two interpolation-like approaches, our proposed feature expansion can generate more uniform output with comparable surface distance deviation.

Comparison of different loss function. As we mentioned above, the EMD can better capture the shape of object than CD, which is demonstrated well by comparing the CD and EMD loss performance in Table 3. We can observe that the EMD loss largely improves the uniformity of the point cloud with low surface distance deviation when comparing with CD loss, which shows that the EMD loss can encourage the output points following the underlying object surface as much as possible. Moreover, by comparing the last two rows in Table 3, we can see that the repulsion loss can further improve the uniformity of the output.

4.6. More Results

Surface reconstruction from upsampled point sets. One of the important applications of point cloud upsampling is for surface reconstruction with a better quality. Therefore, we compare the reconstruction result of our method with those baseline methods with direct Poisson surface reconstruction approach [13] implemented in MeshLab [6]. We use the same parameters for surface reconstruction, and the result is shown in Fig. 5. We can observe that the reconstruction result of our method is the most close to the ground truth, whereas other methods are either missing some structures (*e.g.*, the leg of the horse) or overfilling the hole.

Results of iterative upsampling. In order to demonstrate the ability of our network to handle input point cloud with different number of points, we design an iterative upsampling experiment. In our experiment, we take the output of previous iteration as the input of next iteration. Fig. 6 shows the results. The initial input point cloud has 1024 points and we increase fourfold in each iteration. From the results we can see that our network can produce promising results even with different input point number. Furthermore, such iterative upsampling experiment also proves the anti-noise ability of our network to resist the accumulated error

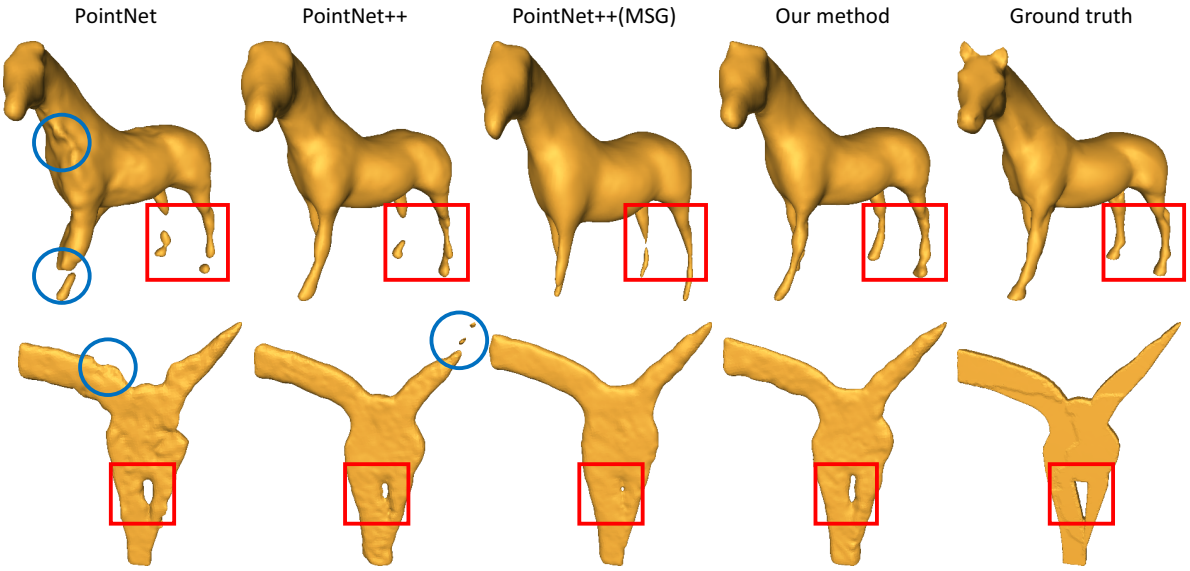

Figure 5. Results of surface reconstruction from upsampled point clouds.

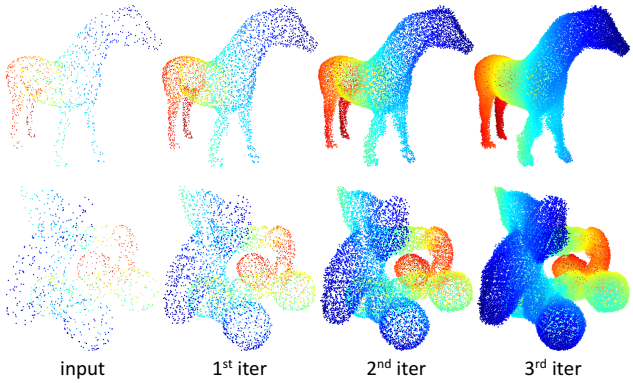

Figure 6. Results of iterative upsampling. We color-code all figures to show the depth information. Blue points are more closer to us.

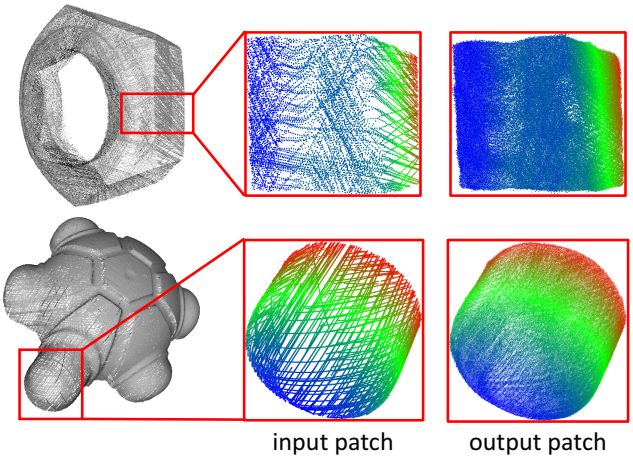

Figure 7. Results on real scanned point clouds (Screw nut & Turtle). We color-code input patches and upsampling results to show the depth information. Blue points are more closer to us.

of iterative upsampling.

Results on real scanned point clouds. To fully evaluate the ability of our network, we show some upsampling results on real scanned point clouds which are downloaded from Visionair [1] in Fig. 7. The left-most column presents the real scanned point clouds. Even though each real scanned point cloud contains millions of points, non-uniform point distribution is still obvious. For better visualization, we cut a small patch from original point cloud as shown in the middle column. We can observe that the real scanned points tend to have a line-like distribution, and our network has the ability to uniformly add points in the sparse region.

5. Conclusion

In this paper, we present a deep network for point cloud upsampling, with a objective of generating a denser and uniform set of points from a sparser set of points. Our network is trained at a patch-level using a multi-level features that aggregate both local and global geometry features. The design of our network bypasses the need for a prescribed order among the points, by operating on individual features that contains enough non-local geometry to allow a context-aware upsampling. Our experiments demonstrate the effectiveness of our method.

In the future, we would like to continue to investigate and develop more means to handle irregular and sparse data, both for regression purposes and for synthesis. One immediate step is to develop a down sampling method. Although, down sampling seems like simpler problem, there is room to devise proper losses and architecture that maximize the

preservation of information in the decimated point set. We believe that in general, the development of deep learning methods for irregular structures is a viable research direction.

References

[1] Visionair. [Online; accessed 14-November-2017]. 5, 8

[2] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Representation learning and adversarial generation of 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017. 2

[3] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Trans. Vis. & Comp. Graphics*, 9(1):3–15, 2003. 2

[4] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 2

[5] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[6] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. *Eurographics Italian Chapter Conference*, 2008. 7

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pat. Ana. & Mach. Int.*, 38(2):295–307, 2016. 3

[8] H. Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. *IEEE CVPR*, 2017. 2, 4

[9] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. *IEEE CVPR*, pages 447–456, 2015. 3

[10] Q. Hou, M.-M. Cheng, X.-W. Hu, A. Borji, Z. Tu, and P. Torr. Deeply supervised salient object detection with short connections. *arXiv preprint arXiv:1611.04849*, 2016. 3

[11] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. on Graphics (SIGGRAPH Asia)*, 28(5):176, 2009. 2, 4

[12] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang. Edge-aware point set resampling. *ACM Trans. on Graphics*, 32(9):1–12, 2013. 2, 5

[13] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. *SGP*, pages 61–70, 2006. 7

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[15] R. Klokov and V. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. *arXiv preprint arXiv:1704.01222*, 2017. 1, 2

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pages 1097–1105, 2012. 4

[17] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. 1

[18] Z. Lian, J. Zhang, S. Choi, H. ElNaghy, J. El-Sana, T. Furuya, A. Giachetti, R. A. Guler, L. Lai, C. Li, H. Li, F. A. Limberger, R. Martin, R. U. Nakanishi, A. P. Neto, L. G. Nonato, R. Ohbuchi, K. Pevzner, D. Pickup, P. Rosin, A. Sharf, L. Sun, X. Sun, S. Tari, G. Unal, and R. C. Wilson. Non-rigid 3d shape retrieval. pages 107–120, 2015. 5

[19] C.-H. Lin, C. Kong, and S. Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. *arXiv preprint arXiv:1706.07036*, 2017. 2

[20] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Trans. on Graphics (SIGGRAPH)*, 26(22):1–5, 2007. 2, 4

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE CVPR*, pages 3431–3440, 2015. 3

[22] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. *Proc. of the IEEE ICCV workshops*, pages 37–45, 2015. 2

[23] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928, 2015. 2

[24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 1, 2, 6

[25] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 1, 2, 3, 6

[26] G. Riegler, A. O. Ulusoys, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. *arXiv preprint arXiv:1611.05009*, 2016. 2

[27] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *Int. Conf. on MICCAI*, pages 234–241, 2015. 3

[28] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *IEEE CVPR*, pages 1874–1883, 2016. 1, 3

[29] S. Wu, H. Huang, M. Gong, M. Zwicker, and D. Cohen-Or. Deep points consolidation. *ACM Trans. on Graphics (SIGGRAPH Asia)*, 34(6):176, 2015. 2

[30] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D shapenets: A deep representation for volumetric shapes. *IEEE CVPR*, pages 1912–1920, 2015. 2, 5

[31] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016. 4

[32] S. Xie and Z. Tu. Holistically-nested edge detection. *IEEE ICCV*, pages 1395–1403, 2015. 3

[33] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017. 4