CAD-PU: A Curvature-Adaptive Deep Learning Solution for Point Set Upsampling

Jiehong Lin, Xian Shi, Yuan Gao, Ke Chen, Kui Jia

Abstract—Point set is arguably the most direct approximation of an object or scene surface, yet its practical acquisition often suffers from the shortcoming of being noisy, sparse, and possibly incomplete, which restricts its use for a high-quality surface recovery. Point set upsampling aims to increase its density and regularity such that a better surface recovery could be achieved. The problem is severely ill-posed and challenging, considering that the upsampling target itself is only an approximation of the underlying surface. Motivated to improve the surface approximation via point set upsampling, we identify the factors that are critical to the objective, by pairing the surface approximation error bounds of the input and output point sets. It suggests that given a fixed budget of points in the upsampling result, more points should be distributed onto the surface regions where local curvatures are relatively high. To implement the motivation, we propose a novel design of Curvature-ADaptive Point set Upsampling network (CAD-PU), the core of which is a module of curvature-adaptive feature expansion. To train CAD-PU, we follow the same motivation and propose geometrically intuitive surrogates that approximate discrete notions of surface curvature for the upsampled point set. We further integrate the proposed surrogates into an adversarial learning based curvature minimization objective, which gives a practically effective learning of CAD-PU. We conduct thorough experiments that show the efficacy of our contributions and the advantages of our method over existing ones. Our implementation codes are publicly available at https://github.com/JiehongLin/CAD-PU.

Index Terms—Point set upsampling, surface approximation, deep learning.

1 Introduction

Surface modeling of 3D objects or scenes is one of the fundamental problems in vision and graphics. Due to its nature of continuity and high-order smoothness, a surface is usually represented as various approximations, with point set, volume, and mesh as the prominent examples. With the popularity of various 3D sensors and the development of surface reconstruction via techniques of multi-view stereo [1], [2], point sets become the most direct representations acquired from real-world sensing. However, such practically captured point sets tend to be noisy, sparse, and possibly incomplete, which restricts their use for high-quality recovery of the underlying surfaces. *Point set upsampling* aims to improve the quality by producing a denser and more regular point set result [3], [4], [5], such that it serves as a better approximation of the underlying surface.

Point set upsampling is by nature an ill-posed problem. Given an input, sparse point set as a poor surface approximation, there exist infinitely numerous solutions whose subsamplings produce the same input. Compared with its 2D counterparts, e.g., image super-resolution [6], the problem of point set upsampling is even more challenging considering that the target is again a surface approximation whose realization is less well defined, which makes it less obvious to specify the upsampling objectives. As such, existing methods take different criteria to achieve the goal. For example, a recent line of deep learning research [3], [5] promotes a uniform distribution of points on the surface, and designs corresponding network modules to meet the criterion; more classical results derive shape operators on the discrete point set via modeling of surface approximation, where curvature-related criteria are often suggested [7], [8]. While local regularity, e.g., uniformity, is a nice property of discrete point sets, it is how the points globally distribute on the surface that determines the quality of geometric and topological surface approximations [9], [10].

In this work, we are motivated to develop deep learning solutions of point set upsampling by formulating it as a problem of learning to improve surface approximation. Following the analyzing framework of moving least squares (MLS) [7], we identify factors critical to the objective by pairing the surface approximation error bounds of the input and output point sets. Given a fixed budget of points in the upsampling result, the analysis suggests that more points should be distributed onto high-curvature surface areas in order for the result to be optimal for surface approximation. To implement the motivation, we propose in this work novel designs hinging on approximate versions of *mean curvature* defined at local surface patches. Specifically, we propose Curvature-ADaptive Point set Upsampling network (CAD-PU) whose key design is a module of *curvature-adaptive feature expansion*. The proposed module is able to expand point-wise features in a globally curvature-adaptive manner, while simultaneously maintaining the local regularity implicitly; as a result, high-quality upsamplings are made possible by the subsequent layers of geometric feature learning and regression of upsampled points. To train CAD-PU, we follow the same motivation of learning to improve surface approximation, and propose geometrically intuitive

- *J. Lin, X. Shi, K. Chen and K. Jia are with the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China.*
  Emails: lin.jiehong@mail.scut.edu.cn, ausxian@mail.scut.edu.cn, chenk@scut.edu.cn, kuijia@scut.edu.cn
- *Y. Gao is with Tencent AI Lab, Shenzhen, China.*
  Email: ethan.y.gao@gmail.com
- *K. Jia is the corresponding author.*

surrogates that approximate discrete notions of surface curvature for the upsampled point set. We further integrate the proposed surrogates into an adversarial learning based curvature minimization objective, which yields a practically effective training of CAD-PU. We conduct ablation studies that confirm the efficacy of our novel designs. Comparative experiments with existing methods, including those of optimization [11] and learning based approaches [3], [4], [5], show the quantitative and qualitative advantages of our method in terms of supporting high-quality surface recovery via point set upsampling. Experiments with noisy inputs and real scans also demonstrate the robustness of our method.

2 Related Works

Optimization of point set upsampling Optimization based methods are paired with the development of point set surface modeling and approximation. They typically define shape operators by which surface recovery and/or point set upsampling can be achieved. For example, built on the MLS projection, Alexa *et al.* [7] propose to upsample a point set by computing the Voronoi diagram on the MLS surface and inserting new points at the vertices of this diagram. Lipman *et al.* [12] introduce a different operator of locally optimal projection (LOP) for point resampling and surface approximation. Huang *et al.* [13] further develop a weighted LOP in order to generate noise- and outlier-free point sets, and an edge-aware point set resampling (EAR) method [11] that first resamples point sets away from edges and then progressively approaches the edge singularities. For point set denoising and completion, Wu *et al.* [14] propose a new representation by associating each surface point with an inner point residing on the meso-skeleton, and consequently, optimization benefits from global geometry constraints from the meso-skeleton.

Deep learning point set upsampling More recently, deep representation learning of point sets is proposed for both semantic [15], [16] and generative tasks [17], including those for point set upsampling [3], [4], [5] and completion [18]. Yu *et al.* [3] firstly propose a data-driven solution of PU-Net that exploits multi-branch convolutions to expand multi-level point features for prediction of point set upsampling. Wang *et al.* [4] present MPU, a patch-based progressive upsampling network that learns different levels of details in multiple steps, resulting in improved upsampling. Li *et al.* [5] make use of the power of generative adversarial networks (GAN) and propose PU-GAN, which sets the existing state of the art for deep learning based point set upsampling.

The present work follows the recent success of deep learning solutions, and improve both the architectural designs and learning objectives such that they are more aligned with the classical results of point set surface approximation [9]. Experiments show that our connection of classical and modern strategies sets the new state of the art.

3 Point Set Upsampling for an Improved Approximation of the Underlying Surface

Assume a training set $\{\mathcal{P}_i, \mathcal{Q}_i^*\}_{i=1}^M$ of M sample pairs, where each $\mathcal{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^N$ represents an input, possibly sparse, point set and $\mathcal{Q}^* = \{\boldsymbol{q}_i^* \in \mathbb{R}^3\}_{i=1}^{rN}$ represents the ground-truth output whose number of points is r factors of that of $\mathcal{P}$. In this work, we consider a learning task of point set upsampling that aims to learn a generator $G(\cdot)$, e.g., a deep network, such that given any test $\mathcal{P}$ sampled from an underlying surface $\mathcal{S}^*$, the generator produces $\mathcal{Q} = G(\mathcal{P})$ that can better approximate $\mathcal{S}^*$.

For any pair $\{\mathcal{P}, \mathcal{Q}\}$, assume that their underlying surface $\mathcal{S}^*$ is C^∞ smooth. By differential geometry we know that $\mathcal{S}^*$ can be locally represented as functions defined on coordinate systems corresponding to its local surface patches. Following the analyzing framework of moving least squares (MLS) [7], when using polynomials as the local functions over 2D domains centered at individual points $\{\boldsymbol{q} \in \mathcal{Q}\}$ and when the local domains are properly defined [7], we have $\mathcal{S}_\mathcal{Q} \in C^\infty$ as an approximation of $\mathcal{S}^*$, where $\mathcal{S}_\mathcal{Q}$ denotes a surface reconstructed from the point set $\mathcal{Q}$; we correspondingly write as $\mathcal{S}_{\boldsymbol{q}}$ for the local, polynomial approximation of the surface patch centered at any $\boldsymbol{q} \in \mathcal{Q}$. We similarly define $\mathcal{S}_\mathcal{P} \in C^\infty$ as another approximation of $\mathcal{S}^*$, and $\{\mathcal{S}_{\boldsymbol{p}}\}$ contains its local approximations. The objective of point set upsampling is thus to produce $\mathcal{Q} = G(\mathcal{P})$ such that $\mathcal{S}_\mathcal{Q}$ constructed from an increased number of points improves the approximation of $\mathcal{S}^*$ over $\mathcal{S}_\mathcal{P}$, i.e., $E(\mathcal{S}_\mathcal{Q}, \mathcal{S}^*) \leq E(\mathcal{S}_\mathcal{P}, \mathcal{S}^*)$.

Assume that the polynomial functions are of degree n. From [9] we know that approximating a local surface patch with $\mathcal{S}_{\boldsymbol{q}}$ gives the error bound $\|\mathcal{S}_{\boldsymbol{q}} - \mathcal{S}_{\boldsymbol{q}}^*\| \leq C(\|\mathcal{S}_{\boldsymbol{q}}^{*(n+1)}\|) \cdot \omega_{\boldsymbol{q}}^{n+1}$, where $\omega_{\boldsymbol{q}}$ denotes the width of local 2D domain centered at $\boldsymbol{q}$ and C is a constant depending on the $(n+1)^{th}$ local surface derivatives. When $\{\mathcal{S}_{\boldsymbol{q}_i}\}_{i=1}^{rN}$ cover the surface with no holes, we have that $\mathcal{S}_\mathcal{Q}$, as the union of $\{\mathcal{S}_{\boldsymbol{q}_i}\}_{i=1}^{rN}$, approximates $\mathcal{S}^*$ with the following error bound

$$E(\mathcal{S}_\mathcal{Q}, \mathcal{S}^*) = \sum_{i=1}^{rN} \|\mathcal{S}_{\boldsymbol{q}_i} - \mathcal{S}_{\boldsymbol{q}_i}^*\| \leq \sum_{i=1}^{rN} C(\|\mathcal{S}_{\boldsymbol{q}_i}^{*(n+1)}\|) \cdot \omega_{\boldsymbol{q}_i}^{n+1}. \tag{1}$$

We similarly have the following error bound for $\mathcal{S}_\mathcal{P}$

$$E(\mathcal{S}_\mathcal{P}, \mathcal{S}^*) = \sum_{i=1}^{N} \|\mathcal{S}_{\boldsymbol{p}_i} - \mathcal{S}_{\boldsymbol{p}_i}^*\| \leq \sum_{i=1}^{N} C(\|\mathcal{S}_{\boldsymbol{p}_i}^{*(n+1)}\|) \cdot \omega_{\boldsymbol{p}_i}^{n+1}. \tag{2}$$

The bounds (1) and (2) tell that approximation errors depend on a coupled factor of local surface derivatives and widths of the corresponding local 2D domains. When $n = 1$, i.e., $\mathcal{S}_\mathcal{Q}$ and $\mathcal{S}_\mathcal{P}$ are piecewise linear approximations of $\mathcal{S}^*$, the bounds are directly relevant to local surface curvatures. We consider this case in the following analysis. Note that it is quite reasonable to set $n = 1$ to have piecewise linear surface approximations, since our visual system is insensitive to surface smoothness beyond second order [19].

By comparing (1) and (2) we have that, under the condition that the residing of upsampled $\mathcal{Q} = \{\boldsymbol{q}_i\}_{i=1}^{rN}$ on $\mathcal{S}^*$ follows the same distribution as that of $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^{N}$, point set upsampling reduces the approximation error since, on average, $\omega_{\boldsymbol{q}} \leq \omega_{\boldsymbol{p}}$. To further reduce the approximation error via point set upsampling, we are motivated by the following two observations:

1) given a fixed budget of rN points in the upsampled $\mathcal{Q}$, distributing more points to surface patches at $\{\boldsymbol{q}\}$ that have higher values of curvature (and thus

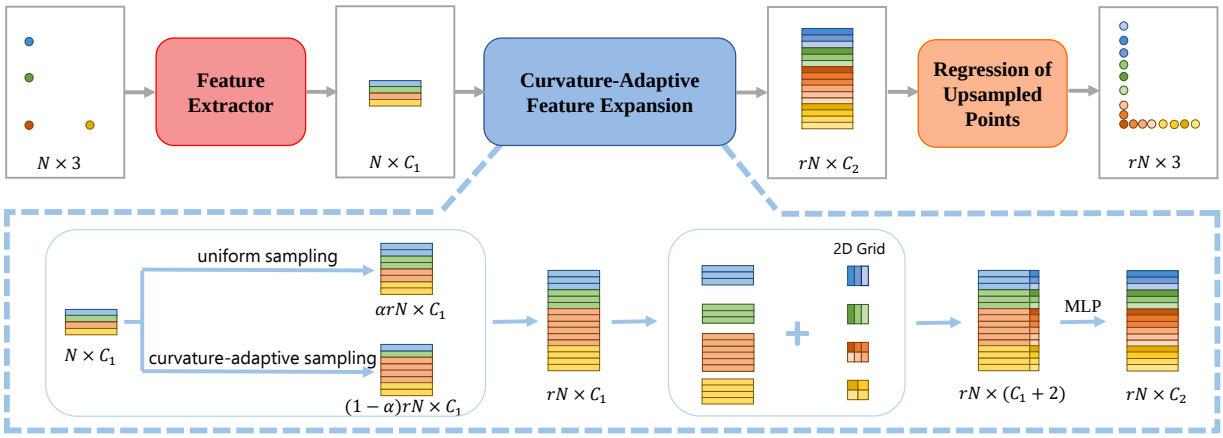

Fig. 1: Architecture of our proposed Curvature-ADaptive Point set Upsampling network (CAD-PU). Given an input set of N points, the network first extracts point-wise features using a same feature extraction module as in [4]. It then expands the N point-wise features as rN ones using our proposed module of curvature-adaptive feature expansion. Coordinates of the upsampled points are finally obtained by a simple regression. The key of curvature-adaptive feature expansion is a hybrid scheme that combines uniform and curvature-adaptive samplings of point-wise features for expansion. Discrimination among the sampled, possibly duplicate, features is also injected by augmenting the features with 2D coordinates at corners of regular grids defined on an arbitrary but fixed 2D plane; this is necessary to make the upsampled points spatially spread out. We train CAD-PU using a novel objective aiming for an improved approximation of the underlying surface via point set upsampling. For better illustration, different points and their associated feature vectors are coded with different colors.

higher values of $\{C(\|\mathcal{S}_{\boldsymbol{q}}^{*(2)}\|)\}$) reduces the corresponding widths $\{\omega_{\boldsymbol{q}}\}$, and thus reduces the overall RHS summation in the bound (1);

2) given that the underlying surface $\mathcal{S}^*$ for a test $\mathcal{P}$ is unknown, an optimal $\mathcal{S}^*$ with smaller summation of local curvatures $\sum_{i=1}^{rN} \|\mathcal{S}_{\boldsymbol{q}_i}^{*(2)}\|$ achieves a lower error bound (1), which is also aligned with the desired property of *fair* surface [20] [1].

To implement the first observation, we propose *Curvature-Adaptive Feature Expansion* as the key module in our proposed generator of *Curvature-ADaptive Point set Upsampling network (CAD-PU)*, which is specifically designed to distribute more points to high-curvature surface patches during the point set upsampling process. For the second one, we propose an objective to train CAD-PU by minimizing discrete and approximate surrogates of surface curvatures for the upsampled point set. These technical designs constitute the main contributions of our proposed CAD-PU, whose details are presented shortly.

4 The Proposed Method

In this section, we first present the elements of our proposed Curvature-ADaptive Point set Upsampling network (CAD-PU), whose illustration is given in Fig. 1. We then introduce our learning objectives that train CAD-PU to generate upsampled point sets.

4.1 Curvature-Adaptive Point-Upsampling Network

CAD-PU processes an input point set by independently and in parallel upsampling its point subsets corresponding to local surface patches. Without loss of generality, we again use $\mathcal{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^N$ to denote an input point subset, whose coordinates are fed into a *feature extractor* to learn point-wise features $\mathcal{F}^1 = \{\boldsymbol{f}_i^1 \in \mathbb{R}^{C_1}\}_{i=1}^N$. Our proposed module of *curvature-adaptive feature expansion* then learns from $\mathcal{F}^1$ to produce $\mathcal{F}^2 = \{\boldsymbol{f}_i^2 \in \mathbb{R}^{C_2}\}_{i=1}^{rN}$, which is finally used to regress the output point subset $\mathcal{Q} = \{\boldsymbol{q}_i \in \mathbb{R}^3\}_{i=1}^{rN}$.

4.1.1 Feature Extraction

We use the feature extractor proposed in [4], [5] to learn point-wise features $\mathcal{F}^1 = \{\boldsymbol{f}_i^1\}_{i=1}^N$, which adopts EdgeConv [16] as its basic blocks. Specifically, the feature extractor stacks 4 dense EdgeConv modules via skip-connections. Each EdgeConv acts on local graphs that are dynamically updated based on feature similarities; by max-pooling the embedded features of graph edges, it is advantageous in capturing non-local features of each point. More specifics of the feature extractor are given in the supplementary material.

4.1.2 Curvature-Adaptive Feature Expansion

This is the key module in our proposed CAD-PU to achieve point set upsampling. It expands at a rate r the input $\mathcal{F}^1 = \{\boldsymbol{f}_i^1\}_{i=1}^N$ of N point-wise features as $\mathcal{F}^2 = \{\boldsymbol{f}_i^2\}_{i=1}^{rN}$ of rN ones. A common practice of feature expansion duplicates $r-1$ copes of $\mathcal{F}^1$ to have $\mathcal{F}^2$, which is adopted in the recent works [3], [4], [5]. Since each point-wise feature $\boldsymbol{f}^1 \in \mathbb{R}^{C_1}$ serves as a function approximator that encodes the geometry centered around a local surface patch, as analyzed in Section 3, this practice is expected to reduce the approximation error w.r.t. the underlying surface $\mathcal{S}^*$ by covering $\mathcal{S}^*$ with an increased number of surface patch approximators, where the improved covering follows the same distribution as how the input N points distribute on $\mathcal{S}^*$. Given a fixed budget of points, Section 3 suggests that a better approximation is achieved when the distribution of points on a surface is *curvature-adaptive*, i.e., more points and their corresponding local approximators are located around high-curvature surface areas. However, for the task of point

1. The fairness of a surface is related to its aesthetic perception. There is no a unique definition of surface fairness; a surface is generally considered *fair* if its curvatures or variations of curvature are globally minimized [20].

set upsampling, the distribution of input points is *practically not guaranteed* to be curvature-adaptive; consequently, feature expansion used in [3], [4], [5] is suboptimal in terms of achieving a better approximation of the underlying $\mathcal{S}^*$.

In this work, we opt for a curvature-adaptive feature expansion scheme whose specifics are as follows. Given the input $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^N$, we first approximate the *mean curvature* $\kappa(\boldsymbol{p})$ for each $\boldsymbol{p} \in \mathcal{P}$ via computation of the *surface variation* [21] at $\boldsymbol{p}$. Let $\mathcal{N}(\boldsymbol{p})$ contain the k nearest neighbors of $\boldsymbol{p}$ in $\mathcal{P}$. The covariance matrix $\boldsymbol{V}(\boldsymbol{p}) \in \mathbb{R}^{3\times 3}$ can be constructed from $\mathcal{N}(\boldsymbol{p})$ as $\boldsymbol{V}(\boldsymbol{p}) = \frac{1}{k}\sum_{\boldsymbol{p}'\in\mathcal{N}(\boldsymbol{p})}(\boldsymbol{p}'-\boldsymbol{p})(\boldsymbol{p}'-\boldsymbol{p})^\top$. Let the three eigenvalues of $\boldsymbol{V}(\boldsymbol{p})$ be $\lambda_3 \geq \lambda_2 \geq \lambda_1 \geq 0$, and the mean curvature $\kappa(\boldsymbol{p})$ at $\boldsymbol{p}$ is simply approximated as $\lambda_1/(\lambda_1+\lambda_2+\lambda_3)$. Given $\{\kappa_i\}_{i=1}^N$ corresponding to input points $\{\boldsymbol{p}_i\}_{i=1}^N$ and their features $\mathcal{F}^1 = \{\boldsymbol{f}_i^1\}_{i=1}^N$, our curvature-adaptive feature expansion samples point-wise features from $\mathcal{F}^1$ according to the following probability

$$w_i = \frac{\log(\kappa_i + 1 + \epsilon)}{\sum_{j=1}^N \log(\kappa_j + 1 + \epsilon)}, \tag{3}$$

where ϵ is a small constant and $\sum_{i=1}^N w_i = 1$. We use the logarithmic function in (3) to stabilize computation of $\{w_i\}_{i=1}^N$ in case that the individual curvatures in $\{\kappa_i\}_{i=1}^N$ vary severely. It is straightforward to know that given (3), point-wise features with high curvatures are sampled more frequently. However, due to the approximate nature of $\{\kappa_i\}_{i=1}^N$ computed via surface variation from the possibly sparse set of $\mathcal{P}$, feature expansion relying purely on probabilistic sampling according to $\{w_i\}_{i=1}^N$ may produce noisy results that deviate from the original shape defined by $\mathcal{P}$, as indicated by the ablation studies in Section 5.3. We instead use a hybrid manner to practically implement our proposed curvature-adaptive feature expansion. Specifically, given $\mathcal{F}^1$, we *uniformly* sample $\alpha r N$, $0 \leq \alpha \leq 1$, point-wise features and *probabilistically* sample $(1-\alpha)rN$ ones according to $\{w_i\}_{i=1}^N$, and combine them to form $\widehat{\mathcal{F}}^2 = \{\hat{\boldsymbol{f}}_i^2 \in \mathbb{R}^{C_1}\}_{i=1}^{rN}$ of rN features. Such a hybrid manner makes a balance between preserving the geometries defined by $\mathcal{P}$ and adapting distribution to high-curvature areas during expansion of point-wise features, and achieves high-quality upsamplings in practice.

There will be *on average* r duplicate copies of point-wise features in $\widehat{\mathcal{F}}^2$ that are from a same $\boldsymbol{f}^1 \in \mathcal{F}^1$. Effective point set upsampling requires injection of discrimination into each group of duplicate features, in order to learn features of higher layer for generation of upsampled points that better approximate the underlying surface $\mathcal{S}^*$. To this end, we first track indices of the sampled rN features and organize them into groups each of which contains duplicates of certain $\boldsymbol{f}^1 \in \mathcal{F}^1$. We follow the trick used in [5] by pre-fixing a 2D plane, on which a regular grid of 2D corner points are defined whose coordinates (e.g., $[1,0]$, $[1,1]$, $[0,1]$, $[-1,1]$, $[-1,0]$, etc) spread away from the origin. Without loss of generality, let $\widehat{\mathcal{G}}^2 = \{\hat{\boldsymbol{f}}_i^2\}_{i=1}^{|\widehat{\mathcal{G}}^2|}$ denote a group of such duplicate features, and $\{\boldsymbol{x}_i \in \mathbb{R}^2\}_{i=1}^{|\widehat{\mathcal{G}}^2|}$ denote the corresponding set of ordered grid corners on the 2D plane. We augment each $\hat{\boldsymbol{f}}_i^2 \in \widehat{\mathcal{G}}^2$ with the corresponding $\boldsymbol{x}_i$, giving rise to the new feature $[\hat{\boldsymbol{f}}_i^2; \boldsymbol{x}_i] \in \mathbb{R}^{C_1+2}$. The augmentation applies to all groups of duplicate features, followed by a multilayer perceptron (MLP) that learns to map (C_1+2)-dimensional point-wise features as C_2-dimensional ones, and produces the output $\mathcal{F}^2 = \{\boldsymbol{f}_i^2 \in \mathbb{R}^{C_2}\}_{i=1}^{rN}$. Fig. 1 gives the illustration. Note that the pre-fixed 2D plane is arbitrarily defined, and the augmented coordinates of grid points on the plane play roles spiritually similar to *anchors* used in object detection [22], [23], from which geometric features of upsampled points are learned in subsequent network layers.

4.1.3 Regression of Upsampled Points

Given the expanded point-wise features in $\mathcal{F}^2 = \{\boldsymbol{f}_i^2\}_{i=1}^{rN}$, we simply use an MLP of 3 fully-connected (FC) layers to regress coordinates of upsampled points that give the final output $\mathcal{Q} = \{\boldsymbol{q}_i \in \mathbb{R}^3\}_{i=1}^{rN}$ of our proposed CAD-PU. $\mathcal{Q}$ is expected to better approximate the underlying surface $\mathcal{S}^*$ by a denser and more curvature-adaptive surface covering. To achieve the goal, we propose loss functions that instantiate the upsampling objective motivated in Section 3 to train CAD-PU, as presented shortly.

4.2 Training Objectives

Given the training set $\{\mathcal{P}_i, \mathcal{Q}_i^*\}_{i=1}^M$, analysis in Section 3 motivates a learning criterion of surface curvature minimization. Using the motivated criterion as a regularizer $R(\cdot)$ gives the following objective to train the generator G of CAD-PU

$$J(G; \{\mathcal{P}_i, \mathcal{Q}_i^*\}_{i=1}^M) = \sum_{i=1}^M L(G(\mathcal{P}_i), \mathcal{Q}_i^*) + \beta \cdot R(G(\mathcal{P}_i)), \tag{4}$$

where β is a penalty parameter, and $L(\cdot,\cdot)$ is a distance of point set whose use is to make the resulting $\mathcal{Q} = G(\mathcal{P})$ as close to the training ground-truth Q^* as possible. The use of regularization $R(G)$ is important to make the learned G generalize to testing samples. We specify our realizations of $L(\cdot,\cdot)$ and $R(\cdot)$ as follows.

Given any training pair $\{\mathcal{P}, \mathcal{Q}^*\}$ and the prediction $\mathcal{Q} = G(\mathcal{P})$, we follow [24] and use Earth Mover's Distance (EMD) as a distance measure between point sets

$$L(\mathcal{Q}, \mathcal{Q}^*) = \min_{\psi:\mathcal{Q}\to\mathcal{Q}^*} \sum_{\boldsymbol{q}\in\mathcal{Q}} \|\boldsymbol{q} - \psi(\boldsymbol{q})\|_2, \tag{5}$$

where $\psi : \mathcal{Q} \to \mathcal{Q}^*$ indicates a bijection. EMD in fact solves an assignment problem by finding the optimal one-to-one correspondence between points in $\mathcal{Q}$ and $\mathcal{Q}^*$, such that the sum of their point-wise distances is the smallest. To efficiently compute EMD, we use the $(1+\epsilon)$ approximation scheme [25].

The regularizer $R(G)$ is motivated to learn a generator $G(\cdot)$ such that the resulting $\mathcal{Q} = G(\mathcal{P})$ for any input $\mathcal{P}$ represents a surface $\mathcal{S}^*$ whose local curvatures are small; consequently, the approximation error $E(\mathcal{S}_\mathcal{Q}, \mathcal{S}^*)$ in (1) is bounded to be small as well. A surface with smaller values of local curvatures also satisfies the desired surface property of smoothness and fairness [20]. To implement the regularizer, given the discrete point set of prediction $\mathcal{Q}$, we rely on the following surrogate that approximately computes the discrete version of *mean curvature* centered at any $\boldsymbol{q} \in \mathcal{Q}$. This is similar to the computation of mean curvature via surface variation in Section 4.1.2, but we here incorporate the generator G, i.e., $\mathcal{Q} = G(\mathcal{P})$, into the computation such that parameters of G can be optimized

$$\kappa(\boldsymbol{q}; \mathcal{Q}) = \frac{1}{k} \sum_{\boldsymbol{q}'\in\mathcal{N}(\boldsymbol{q})} \left|\langle (\boldsymbol{q}' - \boldsymbol{q})/\|\boldsymbol{q}' - \boldsymbol{q}\|_2, \boldsymbol{n}_{\boldsymbol{q}}\rangle\right|, \tag{6}$$

where $\mathcal{N}(\boldsymbol{q})$ contains the k nearest neighbors of $\boldsymbol{q}$ in $\mathcal{Q}$, and $\boldsymbol{n_q} \in \mathbb{R}^3$ denotes the unit normal vector of the surface at $\boldsymbol{q}$. The term (6) measures the averaged angle between the normal vector and the vector defined by pointing $\boldsymbol{q}$ towards each $\boldsymbol{q}'$ of its neighboring points. Since $\boldsymbol{n_q}$ is orthogonal to the tangent plane of the surface at $\boldsymbol{q}$, each inner product in (6) characterizes how the normals vary directionally in the local neighborhood $\mathcal{N}_{\boldsymbol{q}}$, thus approximately measuring the local, directional curvature, and an average of $|\mathcal{N}_{\boldsymbol{q}}| = k$ inner products in (6) approximately measures the local, mean curvature. While $\boldsymbol{n_q}$ can be approximated via eigen-decomposition of the covariance matrix $\boldsymbol{V}(\boldsymbol{q})$ constructed from $\mathcal{N}(\boldsymbol{q})$, in practice, we use $\boldsymbol{n}_{\boldsymbol{q}^*}$ associated with the point in the ground-truth $\mathcal{Q}^*$ that is closest to $\boldsymbol{q}$, which can be pre-computed and efficiently retrieved during the network training. Except for $\kappa(\boldsymbol{q}; \mathcal{Q})$ of (6), we want to further leverage the local geometries in the ground-truth $\mathcal{Q}^*$ to regularize the learning of $\mathcal{Q} = G(\mathcal{P})$, and compute the following discrete notion that bears similarity with mean curvature for a local surface neighborhood $\mathcal{N}^*(\boldsymbol{q})$ defined by embedding any $\boldsymbol{q} \in \mathcal{Q}$ into its ground-truth $\mathcal{Q}^*$

$$\widetilde{\kappa}(\boldsymbol{q}; \mathcal{Q}^*) = \frac{1}{k} \sum_{\boldsymbol{q}^{*\prime} \in \mathcal{N}^*(\boldsymbol{q})} \left| \left\langle (\boldsymbol{q}^{*\prime} - \boldsymbol{q}) / \|\boldsymbol{q}^{*\prime} - \boldsymbol{q}\|_2, \boldsymbol{n}_{\boldsymbol{q}^{*\prime}} \right\rangle \right|, \tag{7}$$

where $\mathcal{N}^*(\boldsymbol{q})$ contains the k nearest neighbors of $\boldsymbol{q}$ in $\mathcal{Q}^*$, and $\boldsymbol{n}_{\boldsymbol{q}^{*\prime}}$ denotes the unit normal vector at the neighbor $\boldsymbol{q}^{*\prime}$. Combining (6) and (7) gives our realization of the regularizer $R(\mathcal{Q}) = R(G(\mathcal{P}))$

$$R(\mathcal{Q}; \mathcal{Q}^*) = \frac{1}{rN} \sum_{\boldsymbol{q} \in \mathcal{Q}} \kappa(\boldsymbol{q}; \mathcal{Q}) + \widetilde{\kappa}(\boldsymbol{q}; \mathcal{Q}^*). \tag{8}$$

Note that (8) averages over rN discrete notions of absolute values corresponding to the rN points of $\{\boldsymbol{q}_i \in \mathcal{Q}\}_{i=1}^{rN}$; minimizing (8) is thus robust to allow a small portion of them to have relatively higher curvature values, while achieving a smaller value of total curvature.

A Strategy of Adversarial Training Adversarial training has shown its power in learning generative models, including those for generation of point sets [5], [17]. Following [5], we introduce a discriminator $D(\cdot)$ that is trained to differentiate the distribution of $\{\mathcal{Q} = G(\mathcal{P})\}$ from that of $\{\mathcal{Q}^*\}$, giving rise to the following problem

$$\min_G J(G; \{\mathcal{P}_i, \mathcal{Q}_i^*\}_{i=1}^M) + \frac{\gamma}{2} \sum_{i=1}^M [D(G(\mathcal{P}_i)) - 1]^2, \tag{9}$$

$$\min_D \frac{1}{2} \sum_{i=1}^M D(G(\mathcal{P}_i))^2 + [D(\mathcal{Q}_i^*) - 1]^2, \tag{10}$$

where γ is a weight parameter. We use PointNet [15] to implement our discriminator $D(\cdot)$, which serves as a simple and effective model of point set classification.

5 Experiments

5.1 Setups

We use a benchmark dataset of 147 object surface models collected by [5] for our experiments. These models include simple and smooth ones (e.g., a cup) and also geometrically complex ones (e.g., a statue of Buddha). Following [5], we use the same 120 objects in the dataset as training models and use the remaining ones for testing. For each training object, we follow [5] and segment its surface into 200 overlapped patches, resulting in a total of $24,000$ training surface patches; training of our CAD-PU and other deep learning methods [3], [4], [5] are conducted on the surface patch level. We prepare the input and output pairs of training point sets as follows. For each surface patch, we uniformly sample 256 points as training input; for training output, we first uniformly sample $10,000$ points from each surface patch and compute their point-wise curvatures (cf. Section 4.1.2), and we then select from them $1,024$ points in a curvature-adaptive manner based on point-wise probabilities computed by (3). Note that for comparison with existing methods [3], [4], [5] whose training outputs are uniformly sampled from object surfaces, we simply uniformly sample $1,024$ points from each surface patch as their training output. In Section 5.2, we also investigate how these methods perform when using curvature-adaptive point sets as their training outputs.

We conduct testing of different methods as follows. For each testing object, we uniformly sample $2,048$ points from its surface and cluster them into 8 subsets of equal size based on spatial proximity; the obtained 256 points per subset are used as input of patch-wise upsampling. Given the default upsampling rate of $r = 4$, we produce $1,024$ output points for each subset, and the final result is formed by a simple aggregation of $1,024 \times 8 = 8,192$ points. Given the different choices of training output point sets used in our CAD-PU and existing methods [3], [4], [5], for a meaningful comparison, we choose to evaluate the upsampling result by measuring how better it can reconstruct the underlying surface. Specifically, for an upsampled result of $8,192$ points, we first use the de-facto standard method of screened Poisson reconstruction [26] to reconstruct its surface mesh, and then uniformly sample two sets of $100,000$ points respectively from the reconstructed surface and the ground-truth one, which enables us to measure the error of surface reconstruction based on point set distance. In this work, we use Chamfer distance (CD) and Hausdorff distance (HD) as the metrics of point set distance. We note that such a quantitative evaluation is more consistent with visual perception of point set upsampling as an improved approximation of the underlying surface.

Implementation details of our CAD-PU are as follows. We use Adam [27] to train the generator and discriminator under a two time-scale update rule (TTUR) [28] for 120 epochs, where the initial learning rates are 0.001 and 0.0001, respectively. Learning rates are decayed by 0.7 every $50,000$ iterations, and the batch size is 28. We set the hyperparameters α, β, γ, k, and ϵ respectively as 0.5, 0.15, 0.005, 12, and 0.01, which work stably well.

5.2 Comparisons with Existing Methods

We compare our CAD-PU with the state-of-the-art deep learning methods of PU-Net [3], MPU [4], and PU-GAN [5], and also with the optimization-based EAR [11]. Results of these methods are obtained using their publicly released codes, with tuning of their optimal hyperparameters. The methods of PU-Net, MPU, and PU-GAN are trained by output point sets uniformly sampled from object surface patches, which are different from ours; except for direct

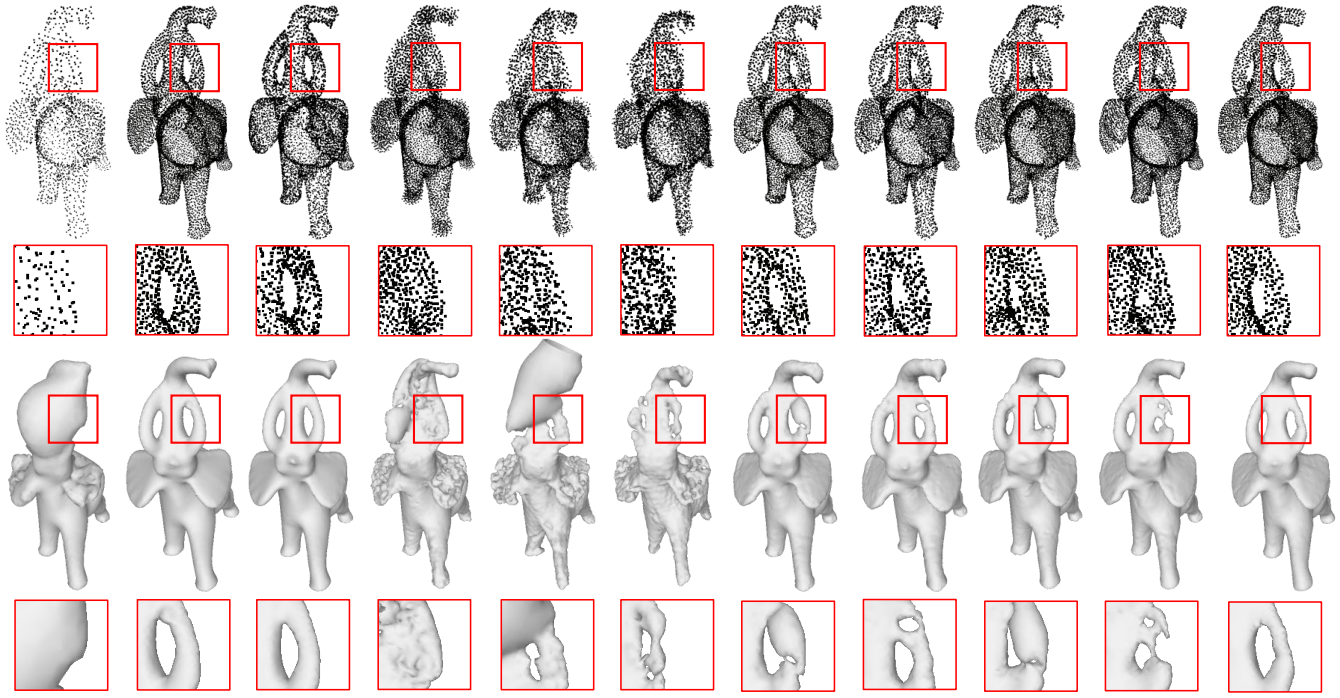

Fig. 2: Qualitative comparisons of different methods on an testing object of elephant. PU-Net (uni.) and PU-Net (cur.) mean that the results are obtained by training respectively with *uniformly* and *curvature-adaptively* distributed point sets. The same applies to MPU and PU-GAN. *More comparative results on other testing objects are shown in the supplementary material.*

comparisons, it is also interesting to observe how these methods perform when using the same training outputs of curvature-adaptive point sets as our CAD-PU does.

Quantitative results in Table 1 show that our CAD-PU outperforms all existing methods, no matter what training point sets they use; training MPU and PU-GAN using curvature-adaptive point sets may not bring benefits, suggesting that network designs and learning objectives of these methods are suboptimal in improving surface approximation via point set upsampling. Qualitative results of different methods on an example of elephant are shown in Fig. 2, where each upsampling result is accompanied with its surface reconstruction. Superiority of our method over existing ones is consistent with those observed in Table 1. Our CAD-PU is particularly advantageous in recovering complex surface geometries at high-curvature areas, e.g., thin structures of surface, while other methods may wrongly glue together spatially close, but originally isolated surface parts. More qualitative results of other testing objects are shown in the supplementary material.

TABLE 1: Quantitative comparisons among different methods. Results are obtained by averaging over the 27 testing objects of the dataset collected in [5]. We evaluate point set upsampling results based on how they can reconstruct the underlying surfaces. We use Chamfer distance (CD) and Hausdorff distance (HD) as the point set based metrics of surface reconstruction error. Refer to Section 5.1 for how the errors are computed.

Methods	CD(10^{-3})	HD(10^{-3})
EAR [11]	1.44	20.75
PU-Net [3]	2.25	30.53
PU-Net [3] with curvature-adaptive GT	1.57	21.76
MPU [4]	0.85	19.24
MPU [4] with curvature-adaptive GT	2.38	36.87
PU-GAN [5]	0.72	16.07
PU-GAN [5] with curvature-adaptive GT	1.55	25.14
CAD-PU	**0.57**	**15.83**

5.3 Ablation Studies

In this section, we conduct ablation studies on the following two aspects of CAD-PU. Table 2 reports the quantitative results, with qualitative example results shown in Fig. 3.

Effect of curvature-adaptive feature expansion We first evaluate the effect of curvature-adaptive feature expansion in our CAD-PU, by controlling the values of α. When $\alpha = 1$, all the features of input points are uniformly sampled for expansion by r times; when $\alpha = 0$, they are sampled proportionally based on their curvatures. Table 2 shows that the best performance achieves at $\alpha = 0.5$, confirming the advantage of our hybrid sampling scheme for point set upsampling via feature expansion.

Effect of the regularizer (8) The regularizer (8) is proposed to improve the surface approximation via point set upsampling. Its efficacy is verified in Table 2 in terms of both CD and HD metrics. In terms of visual quality, the regularizer is particularly helpful to suppress generation of outlier points, as shown by the example in Fig. 3.

5.4 Testing of Robustness

We evaluate the robustness of our method in terms of the following two aspects, by comparing with the state-of-the-art method of PU-GAN [5].

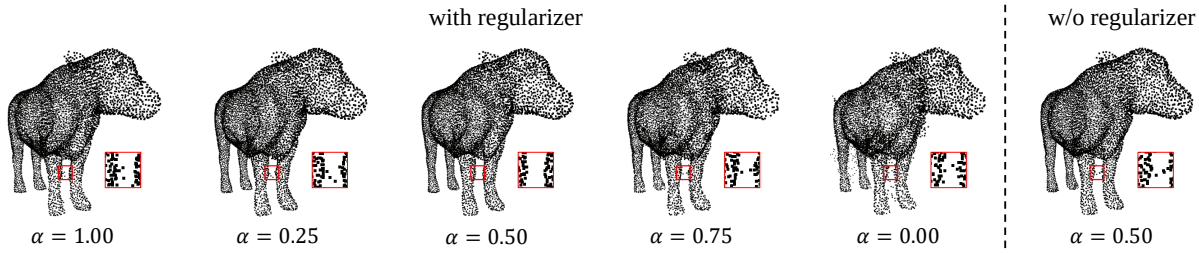

Fig. 3: Qualitative results of ablation studies on the effects of curvature-adaptive feature expansion and the regularizer (8) in our proposed CAD-PU, where the curvature-adaptive degree is controlled by the values of α.

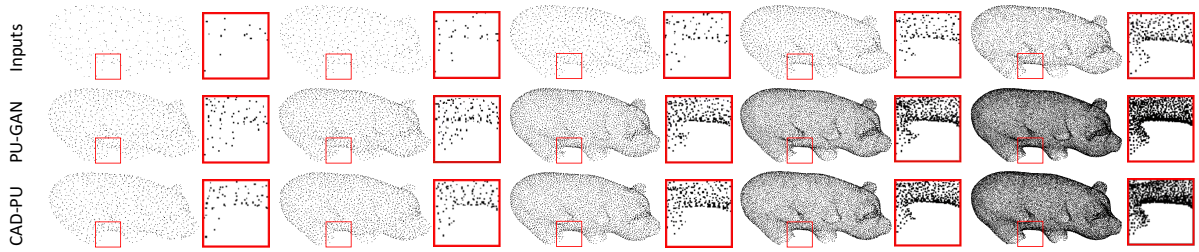

Fig. 4: Qualitative results of robustness testing against varying working scales of spatial extent. The numbers of total input points per testing object (from left to right) are 256, 512, 1,024, 2,048, and 4,096, respectively.

TABLE 2: Ablation studies on our proposed CAD-PU. We report averaged results of Chamfer distance (CD) and Hausdorff distance (HD) over the 27 testing objects of the dataset collected in [5].

α	regularizer (8)	CD(10^{-3})	HD(10^{-3})
0.50	×	2.16	40.54
1.00	✓	1.81	33.18
0.75	✓	1.26	21.90
0.50	✓	**0.57**	**15.83**
0.25	✓	1.83	30.86
0.00	✓	2.58	51.72

TABLE 3: Testing of robustness against varying scales of spatial extent. We report averaged results of Chamfer distance (CD $\times 10^{-3}$) and Hausdorff distance (HD $\times 10^{-3}$) over the 27 testing objects of the dataset collected in [5].

Metrics	Methods	Number of input points per object				
		256	512	1,024	2,048	4,096
CD	PU-GAN [5]	2.63	2.13	1.29	0.72	0.67
	CAD-PU	**2.11**	**1.64**	**1.16**	**0.57**	**0.37**
HD	PU-GAN [5]	40.99	32.90	21.09	16.07	15.56
	CAD-PU	**38.61**	**29.68**	**19.41**	**15.83**	**11.52**

TABLE 4: Testing of robustness against varying levels of input noise. We add point-wise Gaussian perturbations with different standard deviations to input points of the 27 testing objects collected in [5]. Averaged results of Chamfer distance (CD $\times 10^{-3}$) and Hausdorff distance (HD $\times 10^{-3}$) are reported.

Metrics	Methods	Standard deviations of input noise				
		0	0.001	0.005	0.01	0.02
CD	PU-GAN [5]	0.72	0.86	1.14	1.54	2.62
	CAD-PU	**0.57**	**0.78**	**1.05**	**1.44**	**1.87**
HD	PU-GAN [5]	16.07	19.58	24.13	30.16	40.52
	CAD-PU	**15.83**	**18.63**	**22.83**	**25.03**	**33.29**

Robustness against varying scales of spatial extent Both our method and PU-GAN conduct patch-level upsampling, which upsamples a fixed number of input points per surface patch by a specified factor r. When the number of total input points for a testing object varies, it amounts to applying the methods to a varying working scale of spatial extent that contains the same number of points, or equivalently, it amounts to conducting parallel upsamplings at a varying number of surface patches for the testing object. To investigate how our method and PU-GAN perform when the number of input points per testing object varies, we respectively use 256, 512, 1,024, 2,048, and 4,096 input points for any testing object, which correspond to conduct parallel upsamplings respectively of 1, 2, 4, 8, and 16 surface patches. Fewer working patches suggest that the methods are to recover larger-scale surface geometries via point set upsampling; conversely, more working patches suggest that the methods are to recover smaller-scale details of surface geometries. Results in Table 3 tell that across a range of working scales, our CAD-PU outperforms PU-GAN consistently. Qualitative results in Fig. 4 seem suggest that the advantage of our method is more obvious in the regime of smaller working scales.

Robustness against varying levels of input noise Points obtained in real-world settings inevitably contain noise. To investigate how our method and PU-GAN perform against noise perturbations, we add Gaussian noise of varying levels (with standard deviations of 0.001, 0.005, 0.01, and 0.02) in a point-wise manner to the testing inputs. Quantitative results in Table 4 show that our method outperforms PU-GAN, with larger margins for noise-contaminated inputs. Examples in Fig. 5 tell that given noise contaminations, PU-GAN tends to degrade its performance with generation of point outliers, while our results are relatively stable.

5.5 Real-world Evaluation

We finally evaluate our method for upsampling point sets obtained from real-world scans. We apply CAD-PU and PU-GAN to LiDAR-scanned street scenes from Kitti [29], as shown in Fig. 6. Such real-world settings are challenging since the input points are rather sparse and irregular. Since no ground truth is available, we visualize the comparative results in Fig. 6. PU-GAN tends to generate points in blank areas, resulting in obscure object boundaries. In contrast, our

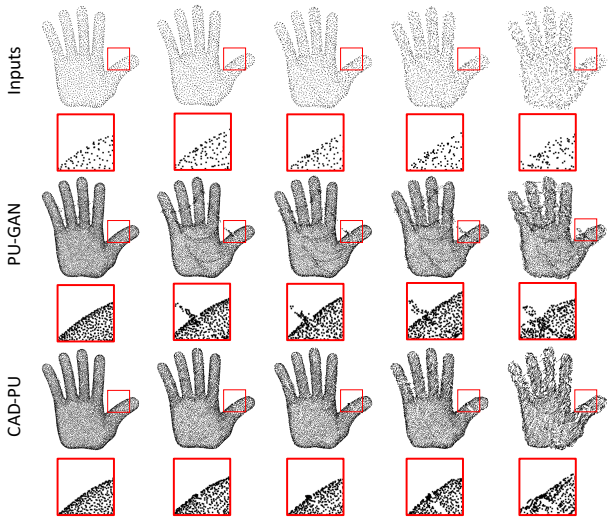

Fig. 5: Qualitative results of robustness testing against varying levels of input noise. The standard deviations of point-wise Gaussian perturbations (from left to right) are 0, 0.001, 0.005, 0.01, and 0.02, respectively.

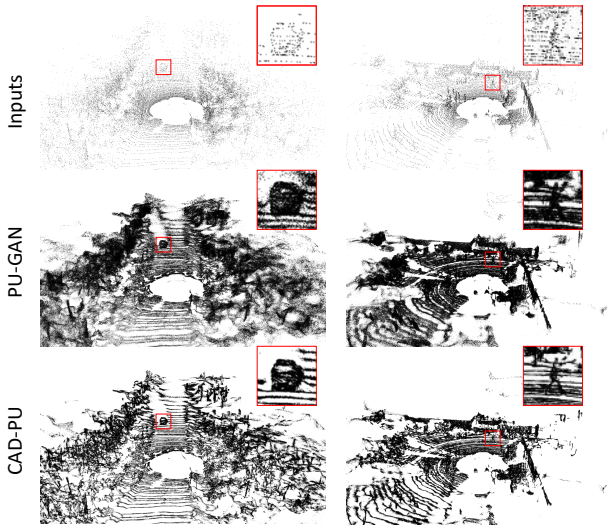

Fig. 6: Real-world testing by applying PU-GAN [5] and our CAD-PU to LiDAR-scanned street scenes from Kitti [29].

method recovers sharper geometries at surface boundaries, due to our internal mechanism of high-curvature attention.

6 Conclusion

This work is motivated to improve surface approximation by learning point set upsamping. To this end, we first analyze the approximation error bounds of the input and output point sets, and identify point-wise curvatures as an important controlling factor that determines the quality of upsampled results. Based on the analysis, we have proposed a novel network design of CAD-PU and the corresponding learning objective. Both quantitative and qualitative results show the advantages of our method over existing ones. Encouraging results are also obtained by applying CAD-PU to LiDAR-scanned street scenes. In future research, we are interested in learning better upsamplings of point sets obtained from real scans of objects and/or scenes, such that downstream tasks of point set analysis would be facilitated.

References

[1] A. Kar, C. Häne, and J. Malik, "Learning a multi-view stereo machine," in *NIPS*, 2017, pp. 365–376.

[2] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 767–783.

[3] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogni*, 2018, pp. 2790–2799.

[4] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3d point set upsampling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogni*, 2019, pp. 5958–5967.

[5] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: A point cloud upsampling adversarial network," in *Proc. IEEE Int. Conf. Comput. Vis.*, October 2019.

[6] P. Milanfar, *Super-Resolution Imaging*. CRC Press, 2017.

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

[8] C. Lange and K. Polthier, "Anisotropic smoothing of point sets," *Comput. Aided Geometric Design*, vol. 22, 2005.

[9] D. Levin, "The approximation power of moving least-squares," *Math. Comp*, vol. 67, no. 224, 1998.

[10] J. Dai, W. Luo, S.-T. Yau, and X. Gu, "Geometric accuracy analysis for discrete surface approximation," *Comput. Aided Geometric Design*, vol. 24, no. 6, pp. 323–338, 2007.

[11] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang, "Edge-aware point set resampling," *ACM trans. on graphics*, vol. 32, no. 1, p. 9, 2013.

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

[13] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM trans. on graphics*, vol. 28, no. 5, p. 176, 2009.

[14] S. Wu, H. Huang, M. Gong, M. Zwicker, and D. Cohen-Or, "Deep points consolidation," *ACM Trans. on Graphics*, vol. 34, no. 6, p. 176, 2015.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. IEEE Conf. Comput. Vis. Pattern Recogni*, vol. 1, no. 2, p. 4, 2017.

[16] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. on Graphics*, vol. 38, no. 5, p. 146, 2019.

[17] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *Int. Conf. Machine Learning*, 2018, pp. 40–49.

[18] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *3DV*, 2018, pp. 728–737.

[19] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. MIT Press, 2010.

[20] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon mesh processing*. AK Peters/CRC Press, 2010.

[21] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *Proc. Conf. Vis.* IEEE Computer Society, 2002, pp. 163–170.

[22] R. Girshick, "Fast r-cnn," in *ICCV*, 2015, pp. 1440–1448.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[24] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogni*, 2017, pp. 605–613.

[25] D. P. Bertsekas, "A distributed asynchronous relaxation algorithm for the assignment problem," in *IEEE Conf. on Decision and Control*, 1985, pp. 1703–1704.

[26] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. on Graphics*, vol. 32, no. 3, pp. 1–13, 2013.

[27] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6626–6637.

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

APPENDIX A
NETWORK SPECIFICS OF CAD-PU

We have introduced our proposed method of Curvature-ADaptive Point set Upsampling network (CAD-PU) in Sec. 4. In this section, we present specifics of the three network modules.

Feature Extraction For point-wise feature extraction, we adopt a same extractor as in [4], [5], which stacks 4 modules of Dense EdgeConv [16] via skip-connections, as illustrated in Fig. 7-(a). We present its specifics for completion of the present paper. Specifically, the input of each Dense EdgeConv module (cf. Fig. 7-(b)) is the output point-wise features of its previous one, except the first one that takes 24-dimensional point-wise features lifted from point coordinates by a fully-connected (FC) layer; the local graph for each point in the module is defined by searching its K nearest neighbors based on l_2 distances in the feature space, from which K edge features associated with the center point are computed and concatenated with the duplicated, input point-wise feature, followed by 3 densely-connected layers of 1×1 convolution to produce the embedded features; these features are finally max-pooled to produce the output of the Dense EdgeConv module. Each skip-connection takes as input the input and output features of its previous Dense EdgeConv module, and processes them via concatenation and FC based feature transformation, as illustrated in Fig. 7-(c).

Curvature-Adaptive Feature Expansion This is the key module in our proposed CAD-PU. We have spelled out its designing details in Sec. 4.1.2. The only parametric component of the module is its final multilayer perceptron (MLP), which is formed by 2 FC layers respectively of 128 neurons.

Regression of Upsampled Points We use an MLP of 3 FC layers to regress the coordinates of upsampled points. The 3 FC layers have 128, 64, and 3 output neurons, respectively.

APPENDIX B
ADDITIONAL RESULTS OF QUALITATIVE COMPARISONS WITH EXISTING METHODS

In Fig. 8, we show more examples of qualitative comparisons among different methods on the testing objects of the dataset collected in [5].

APPENDIX C

In Sec. 5.2, we have compared our CAD-PU with existing methods, including the deep learning-based PU-Net [3], MPU [4], and PU-GAN [5], and also the optimization-based EAR [11]. As we have elaborated in Sec. 5.1, the existing deep learning-based methods use training ground truths of output point sets that are *uniformly* sampled from the object surfaces, while our CAD-PU uses curvature-adaptive ones; to enable the comparisons, we have used an evaluation criterion that measures how better the upsampling results obtained by different methods are able to reconstruct the underlying surfaces; we implement the evaluation criterion by first reconstructing the surface mesh from each upsampling result, and then computing the point set distance-based errors between the reconstructed surface and the ground-truth one, where Chamfer distance (CD) and Hausdorff distance (HD) are used as the metrics of point set distance. Under such an evaluation criterion, we have also reported in Sec. 5.2 the results of PU-Net, MPU, and PU-GAN when training them with the same curvature-adaptive output point sets as those used by our CAD-PU.

For an interest of reference, we supplement here the quantitative errors of different methods by *directly* comparing each upsampling result with the ground-truth point set, i.e., the manner of evaluation used in existing methods [3], [4], [5]. In other words, when the training outputs are uniformly sampled point sets, we compute the CD and HD

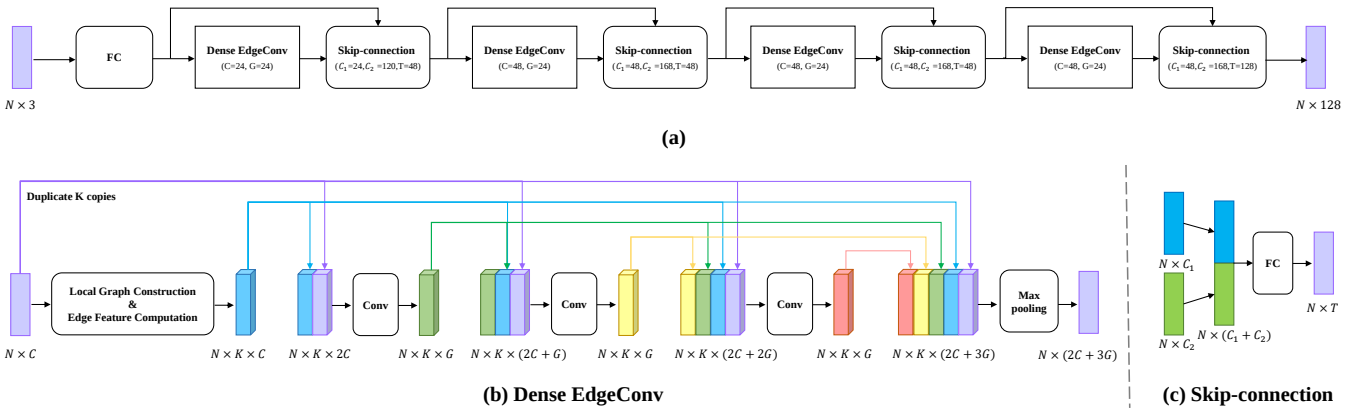

Fig. 7: (a) An illustration of the feature extractor used in CAD-PU, which is the same as that in [4]. It stacks 4 modules of Dense EdgeConv (b) via skip-connections (c). N is the number of input points, and C, G, C_1, C_2, and T denote the respective numbers of feature dimensions used in (b) and (c). In (b), we search $K = 16$ nearest neighbors for each point to define the local graph and compute the corresponding edge-wise features. Color-coded columns represent feature tensors of different sizes.

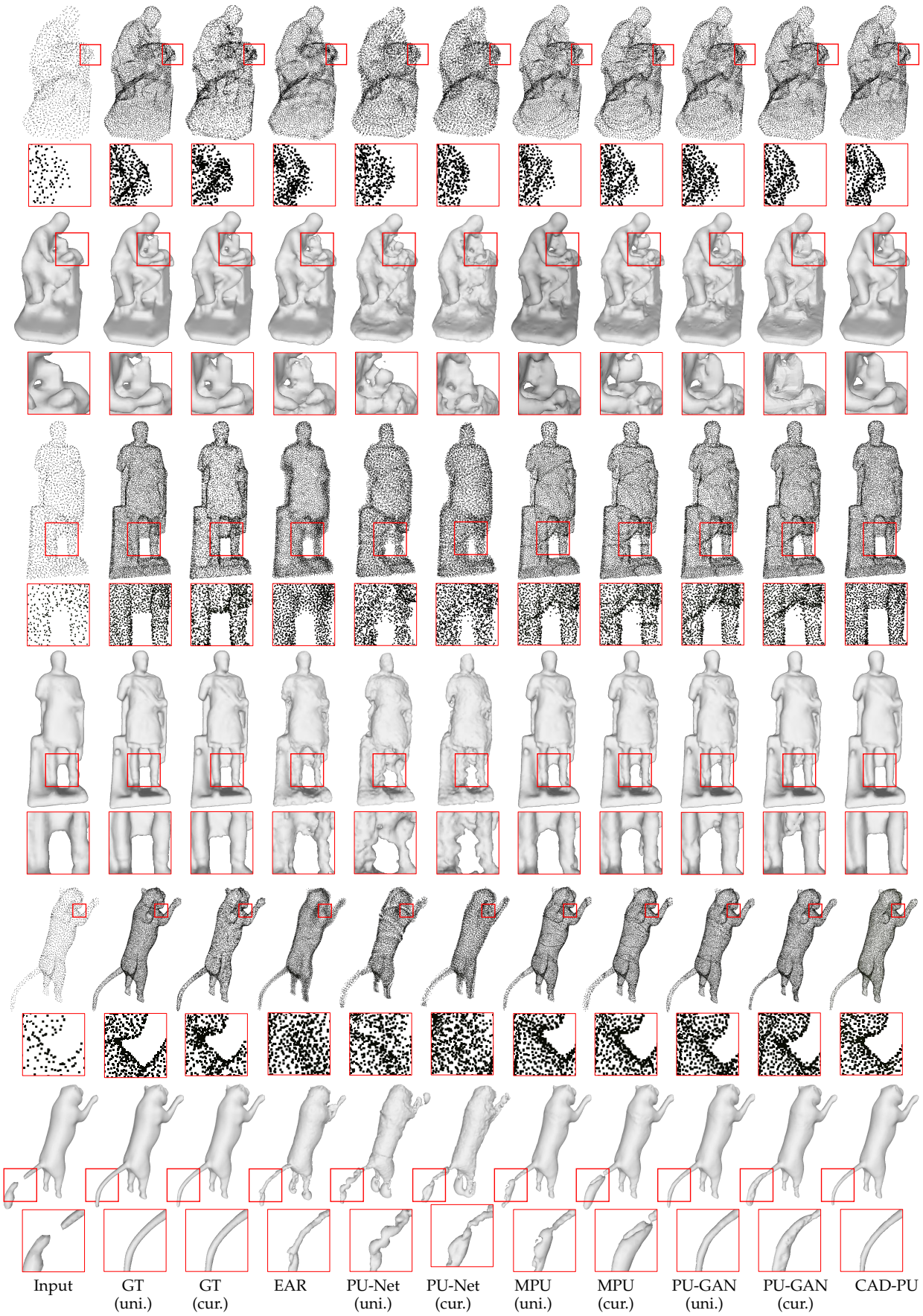

Fig. 8: Qualitative comparisons of different methods on testing objects collected in [5]. PU-Net (uni.) and PU-Net (cur.) mean that the results are obtained by training respectively with *uniformly* and *curvature-adaptively* distributed point sets. The same applies to MPU and PU-GAN.

TABLE 5: Quantitative errors of different methods by directly computing the Chamfer distance (CD) and Hausdorff distance (HD) between each upsampled point set and its ground truth. We consider two cases whose training ground truths of output point sets are distributed either uniformly or in a curvature-adaptive manner. Results are obtained by averaging over the 27 testing objects collected in [5].

Training ground truth	Methods	CD (10^{-3})	HD (10^{-3})
Uniform point set	PU-Net [3]	0.72	8.94
	MPU [4]	0.28	**2.33**
	PU-GAN [5]	**0.24**	4.55
Curvature-adaptive point set	PU-Net [3]	0.66	7.42
	MPU [4]	0.29	3.13
	PU-GAN [5]	0.27	8.04
	Our CAD-PU	**0.26**	**2.34**

values between each testing result and the ground truth of uniformly sampled point set; when the training outputs are curvature-adaptively sampled point sets, we compute the CD and HD values between each testing result and the ground truth of curvature-adaptive point set. Table 5 reports the quantitative errors of different methods under the two training cases.

Remarks We emphasize that for the quantitative errors of the two training cases reported in Table 5: (1) they are *not directly comparable* since their ground-truth point sets are different, and (2) they *do not* well reflect the quality of each upsampled point set as an improved approximation of the underlying surface.