PCEDNet : A Lightweight Neural Network for Fast and Interactive Edge Detection in 3D Point Clouds

CHEMS-EDDINE HIMEUR, THIBAULT LEJEMBLE, THOMAS PELLEGRINI, MATHIAS PAULIN, LOIC BARTHE, and NICOLAS MELLADO, CNRS, IRIT, Université de Toulouse, France

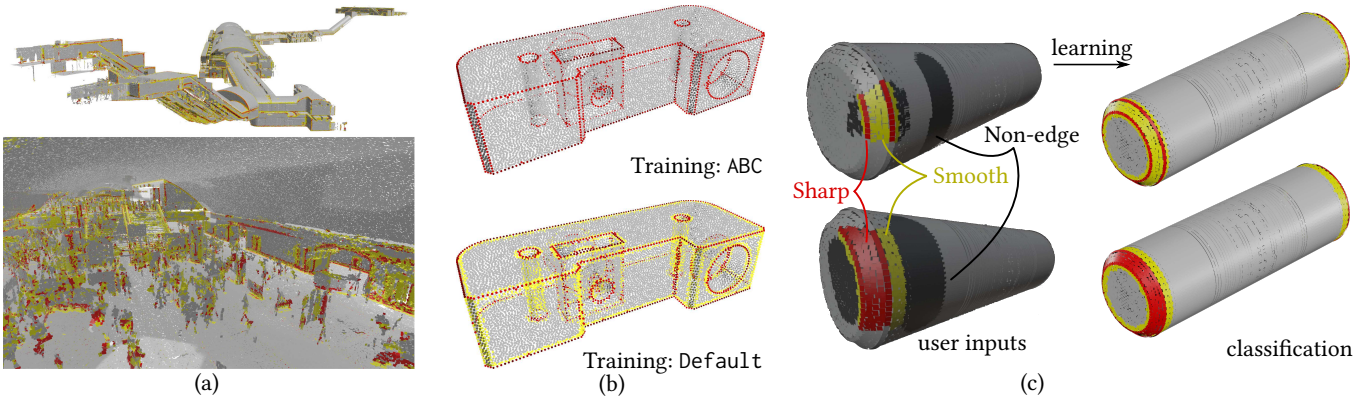

Fig. 1. Three examples of edge detection in point clouds by our PCEDNet neural network. It handles both: (a) the imperfect edges of large scale scans (here 12 million vertices) subject to irregular sampling and noise while detecting both sharp (in red) and smoother (in yellow) edges in few minutes (here less than 6) - and - (b) accurate CAD data on which it can focus on sharp edges if desired, in a few seconds for this model. (c) Our network can also be trained in a few seconds to detect edges following the edge definition provided by a user in an interactive model annotation. We show two annotations corresponding to different user expectations. Most of the processing is precomputed and at runtime edges of this model are classified in less than a second.

In recent years, Convolutional Neural Networks (CNN) have proven to be efficient analysis tools for processing point clouds, e.g., for reconstruction, segmentation and classification. In this paper, we focus on the classification of edges in point clouds, where both edges and their surrounding are described. We propose a new parameterization adding to each point a set of differential information on its surrounding shape reconstructed at different scales. These parameters, stored in a *Scale-Space Matrix (SSM)*, provide a well suited information from which an adequate neural network can learn the description of edges and use it to efficiently detect them in acquired point clouds. After successfully applying a multi-scale CNN on SSMs for the efficient classification of edges and their neighborhood, we propose a new lightweight neural network architecture outperforming the CNN in learning time, processing time and classification capabilities. Our architecture is compact, requires small learning sets, is very fast to train and classifies millions of points in seconds.

CCS Concepts: • **Computing methodologies** → **Point-based models**; **Shape Analysis**; *Neural networks.*

Additional Key Words and Phrases: Point clouds processing, neural networks, edge detection, datasets, energy efficiency, low resource computing

ACM Reference Format:
Chems-Eddine Himeur, Thibault Lejemble, Thomas Pellegrini, Mathias Paulin, Loic Barthe, and Nicolas Mellado. 2021. PCEDNet : A Lightweight Neural Network for Fast and Interactive Edge Detection in 3D Point Clouds. *ACM Trans. Graph.* 41, 1, Article 10 (February 2021), 21 pages. https://doi.org/10.1145/3481804

Authors' address: Chems-Eddine Himeur, chems-eddine.himeur@irit.fr; Thibault Lejemble, thibault.lejemble@irit.fr; Thomas Pellegrini, thomas.pellegrini@irit.fr; Mathias Paulin, mathias.paulin@irit.fr; Loic Barthe, loic.barthe@irit.fr; Nicolas Mellado, nicolas.mellado@irit.fr, CNRS, IRIT, Université de Toulouse, Toulouse, France.

0730-0301/2021/2-ART10 $15.00
https://doi.org/10.1145/3481804

1 INTRODUCTION

Nowadays, acquired point clouds is a very common and widespread representation. The large range of acquisition devices and Computer Vision techniques massively generate clouds with tens or hundreds of millions of points. The shapes sampled by these large unstructured data are arbitrarily complex, and their processing remains extremely tedious. Edges are fundamental features for processing point clouds and their automatic detection is thus useful in a wide range of applications in the fields of computer vision (e.g. feature extraction), computer graphics (e.g. contour line reconstruction) and others. Despite regular advances over the years, it remains an open, very challenging problem.

In general, edges are defined strictly as sharp edges, e.g. for manufactured objects [Koch et al. 2019], or as feature lines for objects. When asked to draw feature lines, users tend to follow more complex rules that may vary from people to people [Cole et al. 2008]. This leads to a lack of clear theoretical definition of an edge, especially on 3D surfaces acquired from models including features of different scales and more or less damaged/clean edges (e.g. stone or plastered buildings, progressively smoothed edges, polished mechanical parts, etc). In addition, an edge can be considered as sharp or smooth depending on the observation scale. This generates contextualized and potentially ambiguous interpretation of what edges are. For instance, different persons produced different annotations

on the models provided for the recent feature curve detection contest [Thompson et al. 2019]. These models include a large variety of different edges, from sharp to smooth and rounded. This underlines that the frontier between an edge and a smooth surface (i.e. a feature boundary and a feature continuity) remains subjective, especially in the case of real world models.

Machine Learning (ML) approaches have gain a lot of interest for their ability to efficiently reproduce theoretically "fuzzy" or complicated classifications for which we are able to provide a sufficiently large set of annotated data. At first glance, they thus seem particularly attractive to create efficient edge detectors. However, in the context of point cloud processing, ML remains very challenging to use as there is neither natural ordering [Zaheer et al. 2017] nor intrinsic parameterization of the input data. In the past years, patches of points have been used by several approaches to apply Convolution Neural Networks (CNN) for classification [Qi et al. 2017], local shape property estimation [Guerrero et al. 2018], and continuous sharp edges classification/ reconstruction [Wang et al. 2020].

For edge detection, geometric approaches avoid using patches by considering geometric descriptors parameterizing each individual point [Demarsin et al. 2007; Weinmann et al. 2013]. Following this idea, Hackel et al. [2016] propose to train a random forest classifier taking as input geometric quantities obtained at multiple scales by linear regression in order to classify points belonging to edges.

In this paper, we introduce both a new way to individually parameterize points, together with a dedicated edge detection lightweight neural network classifier called *Point Cloud Edge Detection Network* (`PCEDNet`). Points are parameterized with a so-called *Scale-Space Matrix* (SSM) (Section 3.2) encoding extrinsic geometric properties of a locally reconstructed surface surrounding each point of the input point cloud at multiple scales. The use of this multi-scale parameterization allows us to propose a compact architecture (Section 3.3) based on a simple neural network enabling both its training in seconds or few minutes on small data sets, and the processing of 500 thousands of points per second on average with our current CPU implementation (Table 10). This fast training and the limited requirement on annotated data allow to easily specialize our network to specific edge definitions, as illustrated in Figures 1-a and 1-b. It is fast enough to allow a user to interactively label small sub-parts of a point cloud and let our network annotate millions of points in seconds (Figure 1-c, Section 5.7).

The choice of our SSM values is validated by an ablation study (Section 5.1), and we evaluate the effectiveness of our network by comparing its speed and accuracy with existing geometric edge detection approaches, point-based processing networks, and two baselines: a Convolutional Neural Network (`CNN`) and a Fully Connected neural network (`FC`) (Section 3.4). Our results (Section 5) demonstrate the superiority of our `PCEDNet` over previous works, and illustrate its efficiency on a large variety of data, from CAD models to real world examples with tens of millions of points (Figures 1-a and 25).

Current networks processing point clouds rely on very deep architectures in an end-to-end strategy. While the way our point parameterization is computed is not novel in geometry processing, the use of these parameters structured in our SSM as input of a network is new. Our proposition illustrates how a simple network exploiting these multi-scale geometric surface descriptors overpasses current approaches. We believe that its superiority in training and evaluation speed, scalability and resource requirements (computation, memory) makes it an example for the creation of lightweight point cloud processing solutions opening new perspectives regarding the system interactivity and adaptation to user's wishes. This is also a step forward to reach real-time point cloud processing with limited resources and low energy consumption (Section 5.8), which are both becoming very important challenges when also considering embedded systems and environmental impact.

2 RELATED WORK

In this section, we first present the way unorganized point clouds are parameterized before being processed for geometric learning (Section 2.1). We introduce the different existing architectures for point-based machine learning and we discuss methods for edge detection from point clouds (Section 2.2). We then review geometric approaches (Section 2.3) dedicated to this topic.

2.1 Point cloud parameterization

Point clouds are most of the time defined as unordered and unstructured set of points sampling an unknown surface. When using neural networks on point clouds, a first challenge is to define a regular structure of the cloud that fits a network architecture, i.e. to *parameterize* the point cloud according to the network architecture. Several approaches have been proposed to tackle this challenge. A first class of approaches parameterizes unstructured point clouds in regular grids, e.g. using series of images taken from different viewpoints [Boulch et al. 2018, 2017; Kalogerakis et al. 2017; Su et al. 2015], or using voxel grids [Maturana and Scherer 2015; Qi et al. 2018, 2016; Wu et al. 2015; Zhou and Tuzel 2018]. The main limitation of these approaches is the cells memory requirement that limits their scalability to very large models especially when detecting thin structures and details, as edges.

A second class of approaches processes each point and its neighborhood, so that the point coordinates are directly processed by the network. PointNet and its variants [Guerrero et al. 2018; Qi et al. 2017; Qi et al. 2017] use multi-layer perceptron (MLP) to consecutively sample and group point coordinates and build geometric features around a point. Several approaches have been proposed to extend convolution to 3D point clouds, using local spectral convolution [Wang et al. 2018], parameterized convolutional filters [Xu et al. 2018], transformed points [Li et al. 2018], sparse lattices [Su et al. 2018], adaptive kernels [Boulch 2019] and kernel point convolution [Thomas et al. 2019].

The strong benefit of point-based convolutions is to allow the design of networks whose first layers learn features at multiple scales directly from the point locations. On the other hand, the characterization of geometric structures relies on the way convolution layers capture those features, only based on the local spatial organization of points. On real data, point sets also include sampling variation, noise, outliers and missing data that also have to be learned.

2.2 Network architectures

Most of the point-based geometry analysis neural networks are tailored for point cloud segmentation [Dai et al. 2017] and classification [Hackel et al. 2017; Zhirong Wu et al. 2015]. They follow the rational introduced by PointNet, where point coordinates are abstracted by successive layers, then reduced using max pooling for feature extraction and finally processed with MLP for the final decision. Such architectures aim at abstracting the input point cloud and estimating high level or semantic properties.

Some approaches also aim at processing point clouds according to their local geometric properties, e.g. normal estimation [Boulch and Marlet 2016]. PCPNet [Guerrero et al. 2018] learns local shape properties (e.g. normal vectors) directly from raw point clouds. Interestingly, this work suggests a multi-scale architecture where several networks process the surrounding of the analyzed point at increasing neighborhood size (one network per size). The features learned by the different networks are stacked to form a feature vector, and processed by a fully connected network to produce the final decision. Recently, the deep Learning Point Network architecture [Lê et al. 2020] improves the implementation of convolution-based point cloud processing networks such as PointNet, DGCNN [Wang et al. 2019] and SpiderNet [Zhao et al. 2020].

Overall, all these networks have been designed to extract semantic information from point clouds, which justify their both deep and large architectures. In our more geometric and specialized context, these architectures become unnecessarily complex while requiring long training and processing times.

2.3 Edge detection

Detecting edges in unstructured point clouds is usually cast as a sharp feature, a feature contour or a curve detection problem. It is often the first step of constrained surface reconstruction algorithms and over the years, many approaches have been proposed in this context. We refer the interested reader to the survey by Berger et al. [2017]. A standard approach is to compute at each point a geometric descriptor using the eigenstructure of the covariance matrix [Gumhold et al. 2001]. It can be a ratio between the eigenvalues, taking into account their evaluation at different scales [Pauly et al. 2003] or not [Xia and Wang 2017], or directly a curvature estimation [Lin et al. 2015; Nguyen et al. 2018]. The ratio between eigenvalues is considered as a more reliable parameter and it is, for instance, used in the CGAL Library [Alliez et al. 2021] with a Delaunay-based feature estimation [Mérigot et al. 2011]. While well established, all these approaches suffer from a sensibility to noise and they perform at a given scale with a strong dependence to a decision threshold. The methods introduced by Pauly et al. [2003] and Bazazian et al. [2015] consider curvature ratio at different scales that reduces the dependence to the scale of analysis and the sensibility to noise, but they remain subject to a decision threshold.

Another family of methods relies on Moving Least Squares surface reconstruction [Demarsin et al. 2007; Ni et al. 2016; Weber et al. 2012]. Using this reconstruction, edge detection can be performed on a Gaussian map clustering computed in a local neighborhood [Weber et al. 2010]. Adaptive reconstruction kernels [Fleishman et al. 2005] can also be combined with polynomial fit [Daniels II et al. 2008]. Other approaches rely either on subspace detection and feature intersection computation [Fernandes and Oliveira 2012], on the mean-shift algorithm to select the farthest points from the centroid of their neighborhood [Ahmed et al. 2018], on the average of neighbors altitude over a local tangent plane [Li and Hashimoto 2017], or on the intersection of planes detected using RANSAC [Mitropoulou and Georgopoulos 2019]. In other contexts, edges are defined as a specific type of lines on the surface. Lin et al. [2015] use RANSAC to spatially regularize the response of a sharp feature detector. Recently, Hackel et al. [2016] proposed a data-driven method where the points are classified and then structured using a graph-based approach. Extending local sharp feature detectors by a global analysis improves the robustness of the detection but limits the scalability, a critical aspect when processing large sets of point clouds defined by tens of millions of points or more.

With an architecture extending PU-Net [Yu et al. 2018], `ECNet` [Yu et al. 2018] is a network consolidating edges after up-sampling the point cloud and detecting its edges. `ECNet` is thus a very deep network requiring important resources for training and processing points at inference time, while being limited in scalability by its up-sampling. Finally, the recent `PIENet` [Wang et al. 2020] first trains two networks based on PointNet++ [Qi et al. 2017] for respectively classifying edge points and corner points. The classified points are filtered with a non-maximal suppression, and clustered by feature using the third variant of PointNet++. A resulting set of curves is then generated by a two-headed PointNet network [Qi et al. 2017]. Even if it is closely related to our proposal, the Pie-Net edge classification is based on a concatenation of several networks with deep architecture, and it is thus also prone to high resource requirements.

In this work, as suggested by Hackel et al. [2016], we propose to take advantage of the discrimination offered by geometric descriptors in a machine learning approach. We increase the robustness to noise and the adaptation to the different feature size and shape by the use of stable multi-scale descriptors including derivatives over scales. We then use these parameters together with a specifically designed neural network. By relying on this set of descriptive parameters, our network avoids the deep design of existing approaches by only requiring a very compact architecture that recursively combines the features learned at different analysis scales.

3 METHOD

3.1 Problem statement

Given a surface sampled by a point cloud, a sharp edge is commonly defined as a tangential discontinuity. Weber et al. [2010] define edges as sharp features (crests and valleys) between two meeting planes, as well as corners at the point of intersection between three or more planes. Hackel et al. [2016] call them "wire-frame contours", and define edges as linear features along which the orientation (normals) of the underlying surface exhibits an unusual discontinuity.

There are at least two main reasons why these definitions are restrictive and not really practical for the analysis of point clouds:

(1) **Sparsity**: Points of acquired point clouds are unequally distributed over the object surfaces. Sharp edges are by nature very sparse, and it is very unlikely that the points of a point

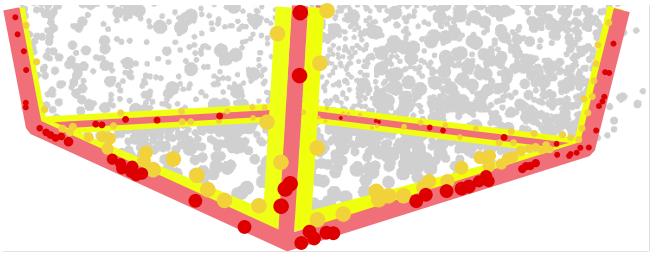

Fig. 2. Synthetic point cloud sampling a cube with sharp edges (red lines). Due to the sparse sampling and noise, only a few points lie on edges (red points). Detecting the edge surrounding (yellow points) improves the robustness to noise and sampling variation.

cloud actually sample the exact edge of the underlying surface as illustrated by the red points in Figure 2.

(2) **Rounding**: Acquired point clouds are composed by points sampling real-world objects, on which edges are always more or less rounded (Figure 16)/damaged (Figure 25). For instance, two facades of a building might be locally connected by a continuous curved surface considered as an edge at the scale of the building, which may be unlikely to be detected at finer scales such as the one of the bricks.

In order to handle these situations, we propose to classify points as **sharp-edge** when they lie on a sharp edge, and as **smooth-edge** when they are (1) near a sharp edge or (2) on a rounded edge. Other points are then labeled as **non-edge**. Depending on the application context and the artifacts found in the data, one might consider either one or the two edge classes in the results (we experimentally validate this proposition of two edge classes in Section 5.1). In this paper, we denote our network as `PCEDNet` when it is used with three classes and as `PCEDNet-2C` when it is used with two classes. Considering that we do not have a universal and practical definition of an edge, we propose to learn the point-based classification of this feature from examples. For instance, when edges have similar geometries and discretization over all models, `PCEDNet-2C` may be preferred while `PCEDNet` is suggested when the edge definition is more fuzzy. In that case, the **sharp-edge** class is to be used to annotate what is considered as an edge (whatever sharp or smooth/rounded) and the **smooth-edge** class is rather to annotate points at proximity of the points tagged as sharp-edges or on features that are close to what is considered to be an edge. We do not suggest any automatic or threshold-based annotation approach as, so far, they are not robust enough and they do not enable the flexibility of letting the user defining his definition of edges through annotations.

In order to be robust to acquisition artifacts, sampling issues and edge roundness, we first reconstruct the surface described by the input points using a robust reconstruction algorithm at multiple scales. We then compute geometric descriptors of the reconstructed surfaces, which are parameters to be processed by a machine learning algorithm (see Section 3.2). We show how this parameterization can be used with a common CNN and a fully connected neural network (Section 3.4), and we propose a dedicated architecture (Section 3.3) that outperforms these networks as well as previous edge classification approaches.

3.2 Scale-Space Matrix

Our approach is inspired by the Growing Least Squares (GLS) approach [Mellado et al. 2012], where the geometry surrounding a point is described by the differential properties of the surfaces reconstructed from neighborhoods of increasing size. We first present the basics of GLS, and then introduce the Scale-Space Matrix (SSM), which wraps in a regular structure the differential properties of the surfaces reconstructed at different scales.

The GLS extends the concept of Scale-Space analysis [Lindeberg 1993; Witkin 1987] to point-based shape analysis. The key idea is to detect pertinent geometric structures and scales as *stabilities* in scale-space. Stabilities are found when the magnitude of the derivatives of the surface is minimized when the scale varies. Due to its multi-scale nature, this approach disambiguates between noise and features, and detects geometric features defined at arbitrary scales. As such, points on a rounded and a sharp edge have discriminative descriptors and can be disambiguated during classification.

We denote $\mathcal{S}^t$ the *continuous surface* reconstructed at scale t, defined as the 0-isosurface of a scalar field $S^t(\mathbf{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}$. We use the Algebraic Point Set Surfaces (APSS) [Guennebaud and Gross 2007] to reconstruct continuous surfaces from raw point clouds. This approach has been proven to be fast and stable at large scales [Guennebaud et al. 2008]. As many other previous work, the scale is controlled by varying the size of a neighborhood ball centered around the evaluation point.

In its original formulation, the pertinent scale extraction introduced by Mellado et al. [2012] combines several descriptors measuring the variation of local relief (i.e. the algebraic distance from the point to the locally reconstructed surface) $\frac{\delta\tau(\mathbf{x},t)}{\delta t}$, normal orientation $\frac{\delta\eta(\mathbf{x},t)}{\delta t}$ and mean curvature $\frac{\delta\kappa(\mathbf{x},t)}{\delta t}$, where $\tau(\mathbf{x},t) : \mathbb{R}^4 \rightarrow \mathbb{R}$, $\eta(\mathbf{x},t) : \mathbb{R}^4 \rightarrow \mathbb{R}^3$ and $\kappa(\mathbf{x},t) : \mathbb{R}^4 \rightarrow \mathbb{R}$ are scale-invariant properties of the surfaces $\mathcal{S}^t(\mathbf{x})$ (mathematical formulations are presented in Appendix A). They also use $\phi(\mathcal{S}^t(\mathbf{x}))$, the residuals of the fitting process, as confidence value.

In this work we propose to extend this idea further by measuring the surface variation in scale and space around each sample $\mathbf{p}_i$ of the input point cloud. We do not seek at defining an hand-crafted descriptor, but rather providing differential properties of the surface that are discriminative for edge detection. According to the GLS formalism, the parameters τ, η and κ are differentiable both in scale and space, which leads to a Jacobian matrix of 5×4 entries (see the description in Appendix A). By keeping only the quantities that are invariant by rigid transformations and not linearly dependent, we obtain the following feature vector X_i^t:

$$X_i^t = \left[\tau_i^t \quad \kappa_i^t \quad k1_i^t \quad \frac{\delta\tau_i^t}{\delta t} \quad \frac{\delta\kappa_i^t}{\delta t} \quad \phi(S_{\mathbf{p}_i}^t)\right] \tag{1}$$

where $\tau_i^t = \tau(\mathbf{p}_i, t)$ and $\kappa_i^t = \kappa(\mathbf{p}_i, t)$. $k1_i^t$ measures the magnitude of the first principal curvature of the surface, and is computed from $\frac{\delta\eta_i^t}{\delta\mathbf{x}}$. According to our experiments, high values of τ, which measure the local relief, helps to disambiguate between rounded and sharp edges. The scale derivatives $\frac{\delta\tau_i^t}{\delta t}$ and $\frac{\delta\kappa_i^t}{\delta t}$ denote the stability of the reconstruction scale s_j.

The Scale-Space Matrix (SSM) defines a structured parameterization of the surfaces $\mathcal{S}^t$ described by the feature vectors X_i^t computed

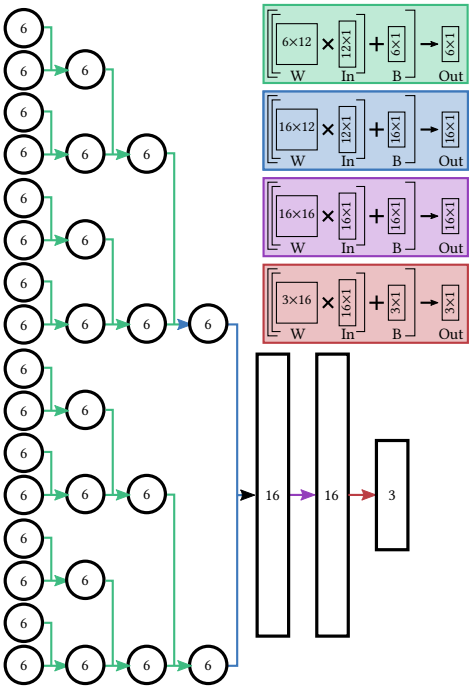

Fig. 3. Our PCEDNet architecture. The left column of circles represents the input sixteen 6-dimensional vectors X_i^t. They are successively pairwise concatenated and processed by dense layers coupled with 6 neurons as depicted in the green box (W: weights, In: input and B: bias). The output vector of size 12 is then processed by a multi-layer perceptron (boxes blue, purple and red) for providing the output classification.

at N scales on each of the M points of the cloud. It is thus defined as follows:

$$SSM = \begin{bmatrix} X_1^1 & X_2^1 & X_3^1 & \dots & X_M^1 \\ X_1^2 & X_2^2 & X_3^2 & \dots & X_M^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_1^N & X_2^N & X_3^N & \dots & X_M^N. \end{bmatrix}. \quad (2)$$

The minimum scale $s_{\min}$ is defined as the mean distance between points computed using the k nearest neighbors of each point (k=10), and the maximum scale $s_{\max}$ is 10% of the diagonal of the point cloud axis aligned bounding box (the effect of the variation of these values is presented in Section 5.9). According to previous work [Bronstein and Kokkinos 2010; Mellado et al. 2015], we use a logarithmic scale sampling to obtain scale-invariant feature vectors, such that:

$$s_i = \left(\frac{s_{\max}}{s_{\min}}\right)^{\frac{i-1}{N-1}} * s_{\min}\ ,\ i = 1..N. \quad (3)$$

For all our experiments we use 16 scales distributed according to Equation 3 (the use of different number of input scales is discussed in Section 5.2). Even though we carefully selected the set of descriptors defining our SSM for their geometric meaning with respect to edge detection, we validate the optimality of this choice when parameterizing our neural network by an ablation study presented in Section 5.1.

3.3 Our network: PCEDNet

The architecture of our Point Cloud Edge Detection Network, denoted PCEDNet, is depicted in Figure 3. The input data is provided to the network as sixteen 6-dimensional vectors X_i^t per point, each vector corresponding to a scale (Figure 3-left column). The sixteen scale vectors are concatenated by groups of two in order to form eight 12-dimensional vectors. More precisely, the first scale is grouped with the second one, the third one with the fourth one, and so on. The idea is to halve the number of scales iteratively until obtaining a single 12-dimensional vector as a final feature representation (four left columns of black circles in Figure 3).

All the layers are fully-connected layers, also called dense layers (linear layer with biases, depicted in Figure 3-green box), followed by a sigmoid activation function. For the first layer, each vector of size 12 is given to a dense layer comprised of 6 neurons. There are eight input vectors and each vector is fed to its own 6-neuron layer. The weights are not shared between these small layers, allowing to process the scales differently, as would do a convolution layer, but here, we combine scales by groups of two. The subsequent three layers perform similarly, but with 48, 24, 12 neurons, respectively.

The final 12-dimensional feature vector is given to a multi-layer perceptron (MLP) responsible for the classification (MLP architecture is presented in Figure 3-blue/purple/red boxes). This MLP is composed of two 16-neurons dense+sigmoid layers followed by the output layer with 3 neurons and a softmax activation function. The total number of weights is about 2.1k, which makes our architecture very compact.

The goal of grouping different scales together is to observe the input shape at different scales in a simultaneous and more intricate way than without grouping them. We expect that this eases the simultaneous detection of higher-scale and sharp geometric properties of the input point clouds. Another reason is that such a representation helps the model to cope with noise in the point clouds.

The "tree structure" merging scales two by two iteratively can be seen as a variant of a grouped convolution operation. It preserves the high computing speed of keeping the scales separate while also mixing the scales layer after layer. It is expected to reduce the chance of being biased by a specific scale and the sensitivity to scale-dependent noise.

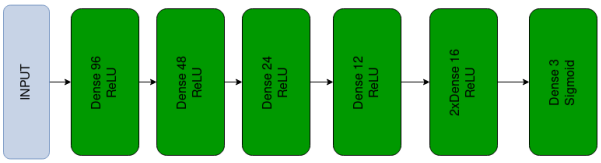

Fig. 4. Architecture of the FC baseline

3.4 Baseline models

The use of this new architecture is justified only if it performs better than more common choice of networks. We thus incorporate as baselines two networks in our evaluation:

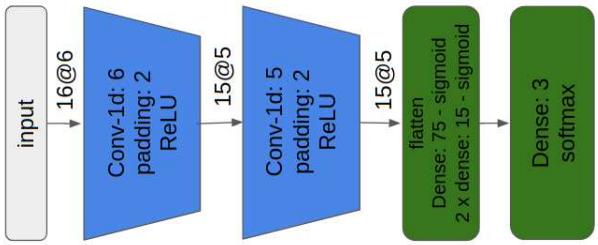

Fig. 5. Architecture of the CNN baseline

- FC: A *Fully Connected* variant of PCEDNet where all scales are connected (Figure 4).
- CNN: A 1-d *Convolutional Neural Network* with a number of layers similar to our PCEDNet architecture (Figure 5).

FC. This architecture presented in Figure 4 allows us to measure the impact of the pairwise scale connection introduced in PCEDNet. FC and PCEDNet architectures differ by the four first layers: instead of consecutively combining scales by pairs, the FC baseline combines all the scales in unique dense layers while keeping the same reduction rate, i.e. 16-8-4-2 scales. We use the same number of neurons than PCEDNet to get comparable architectures. In total, the FC baseline is comprised of about 6663 weights (three times more than PCEDNet). The input data is provided to the network by concatenating the sixteen 6-dimensional vectors used by PCEDNet in a single 96-dimensional vector.

CNN. We chose a standard convolutional neural network architecture presented in Figure 5, composed of two blocks: a first block for representation learning with two 1D convolution layers, each followed by a ReLU activation function, and a second block dedicated to classification with three fully-connected layers and one 3D output layer. We tried a large number of variants and the one we present is the best we obtained by trials and errors. The two convolution layers are composed of 15 filters, with a kernel size of 6 and 5, respectively. Zero padding of size 2 was used to get outputs of shape 15×5. The three dense layers are composed of 75, 15 and 15 neurons, respectively, followed by the output layer with 3 neurons. A sigmoid activation function is used after all the dense layers, except the output layer that uses a softmax non-linearity function. In total, CNN is composed of about 8.7k weights. The input data is also provided to the network as sixteen 6-dimensional vectors. Using the different scales as channels enables a potentially larger expressivity power to the model since each convolution filter uses different weights to combine the scales.

4 EXPERIMENTAL SETUP

4.1 Point cloud dataset

We measure the efficiency and adaptability of our network on three different datasets with ground truth for learning and evaluation: edges on point clouds with acquisition artifacts (Default), sharp edges on CAD models (ABC) and annotated curves on challenging shapes (SHREC). Each dataset is split in three sets: training set (denoted T, used for training), validation set (denoted V, used for learning to monitor accuracy) and evaluation set (denoted E). Visual evaluations are also conducted on a set of acquired models. Point clouds with classification results are shown in the website accompanying the submission.

In addition to annotated learning sets, we also show in Section 5.7 how PCEDNet can learn from small set of data annotated interactively by a user and classify point clouds without requiring additional predefined training data.

Default *dataset.* We design this dataset to emulate geometric structures regularly found in acquired point clouds. It is as small as possible, in order to demonstrate that a few annotated data used for training our network in a short time are enough for an effective detection (see timings in Table 2). The Default dataset is composed of 8 synthetic models containing edges of different shapes illustrated in Figure 6 and 7, and detailed in Table 1. We also included some of our models characterized by varying densities and shapes (i.e. Cube, Cone, Fandisk and 2-cube), with different levels of noise generated by a random motion on the points following a Gaussian distribution with standard deviation denoted σ (the different values of σ per model are given in Table 1-Noise). We defined the validation set by selecting a random subset within the training set with an equal number of points across the three classes (1k per class, 3k in total). We also created an evaluation set with specific features to better evaluate the networks against noise and sharp angles. These point clouds, not used for training, are taken from the testing set introduced by Bazazian et al. [2015] to evaluate the algorithm Covariance Analysis (denoted CA in the following). The models used for training, validation and evaluation are detailed in Table 1-Usage.

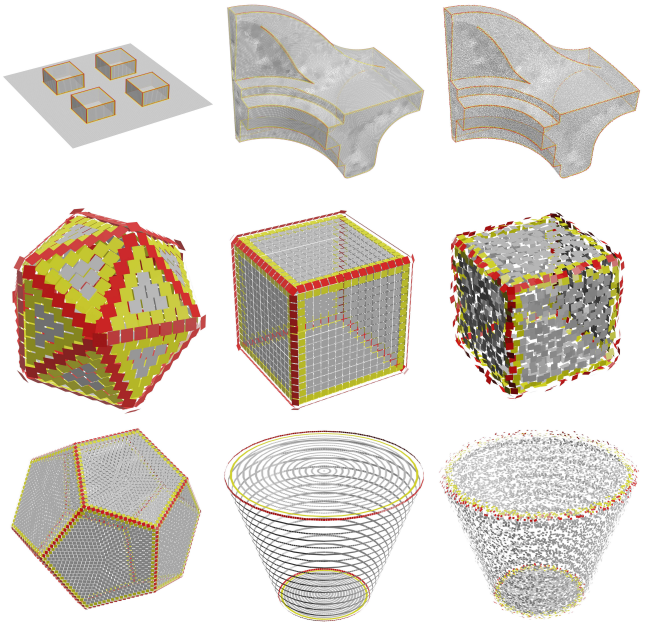

Fig. 6. Training set of the dataset Default, with sharp edges points in red, and smooth edges points in yellow.

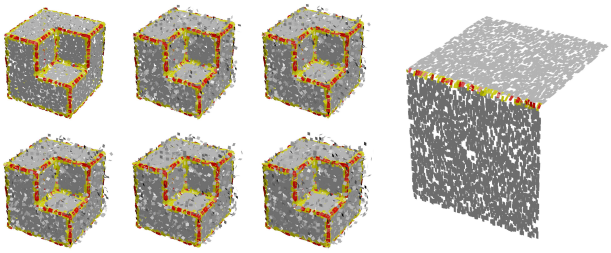

Fig. 7. Evaluation set of the dataset Default. Left: 2-cube, with increasing Gaussian noise (σ ranging between 0 and 0.14 units, cube edge length 1). Right: Angle (90 degrees).

Dataset	Model	Usage	#k points	% points/class			Noise (σ)
				Edge	S-Edge	Other	
Default	Cube	T, V	1.5	12.2	21.8	65.9	0-0.1
	Icosahedron	T, V	0.6	34.6	46.7	18.6	-
	Cone	T, V	7.7	6.1	11.7	82.2	0-0.1
	Dodecahedron	T, V	12	3.5	7.7	88.7	-
	Fandisk	T, V	106.5	5.5	5	89.4	0-0.1
	4-cube	T,V	43.9	6.8	6.6	86.5	-
	2-cube	E	7.2	6	17.3	76.6	0 - 0.01 0.02 - 0.03 0.12 - 0.14
	Angle	E	6	0.6	1.6	97.8	-
Default (total)		T	287.9	9	11	80	-
		V	3	33.3	33.3	33.3	-
		E	55	4.9	14.54	80.56	-
SHREC		T	89.4	13.98	-	86.02	-
		V	94.4	8.68	-	91.32	-
		E	654.7	10.57	-	89.43	-
ABC		T	3212.4	4.66	-	95.34	-
		V	690	5.79	-	94.21	-
		E	312314	5.52	-	94.48	0 - 0.04

Table 1. Statistics of the datasets used for training (T), validation (V) and evaluation (E). The Edge column gives the percentage of points of the class sharp-edge, the S-Edge, the one of the class smooth-edge and Other is for the non-edge class.

Table 1 presents quantitative details (including the values used for σ), and the point clouds are shown in Figures 6, 7 and in the joined website.

ABC *dataset.* We use the ABC dataset [Koch et al. 2019] to evaluate the performance of our network on the detection of sharp edges in CAD models. We use the chunk 000, which contains 7167 models represented as OBJ files. We generate the point clouds by exporting the vertices and the normal vectors of the meshes. The ground truth classification is produced using the vertices associated to a sharp feature in the feature files. A notable difference with the Default dataset is that no ground truth information is provided for the smooth-edge label. Only sharp edges are considered and other smoother features are thus labeled as non-edge. We define the training and evaluation sets by randomly selecting 200 and 50 models respectively (see the model list in the joined website).

SHREC *dataset.* We use the dataset produced for feature curve extraction by Thompson et al. [2019] in order to evaluate the performance of our network on challenging data annotated by humans. As for ABC, we consider the vertices and normal vectors of the given meshes, and mark the annotated vertices as edge vertices. Similarly, we define the training and evaluation sets by randomly selecting models, e.g., (M5, M6 and M14) and (M2, M11) respectively. This procedure is not ideal as these models exhibit very heterogeneous edge shapes, but we found it fairer than randomly sampling the point clouds.

Acquired point clouds. We also perform visual evaluations on 9 acquired point clouds whose number of vertices are reported in Table 10. Loudun 1 and Loudun 35 are down-sampled versions (of respectively 1 and 35 million vertices) of a photogrammetric acquisition of the Square Tower of Loudun (France). We used Euler, Empire and Lans as provided by Monszpart et al. [2015], Pisa Cathedral as provided by Mellado et al. [2015], Munich as provided by Hackel et al. [2016], Paris rue Madame as provided by Serna et al. [2014], and we downloaded Church and Train Station from Sketchfab[1].

Normal estimation. The normals used for the GLS computation are either directly those estimated with the acquisition technique (e.g. photogrammetry, laser scanner), which are actually provided for most current acquisition devices, or those computed automatically from the point samples (we used Meshlab [Cignoni et al. 2008] to estimate normals on Default dataset and on Euler, Lans, Church and Train Station). We did not encounter any difficulty related to normal estimation. Eventually, oriented normals may be avoided and replaced by unoriented normals [Chen et al. 2013a], which could be easily estimated using local fitting.

4.2 Networks training

Implementation details. We implemented PCEDNet in C++ and ran our experiments on a 10-cores Intel Xeon(R) CPU E5-2640 v4 (20 threads), with 128GB RAM. We use the Ponca library [Mellado et al. 2020] for surface fitting and derivative computation. These surface fitting computations (GLS) are performed on two 10-cores Intel Xeon(R) CPU E5. The network modeling and evaluation for PCEDNet and FC is implemented in our own prototype using Eigen [Guennebaud et al. 2010] for linear algebra. The baseline CNN is implemented, trained and evaluated using Pytorch.

Categorical cross-entropy is used as objective function for both the baseline architectures and PCEDNet. On each dataset, CNN, FC and PCEDNet are trained for 200, 200 and 40 epochs respectively, all reaching 98% accuracy on the validation sets. Weights are initialized randomly using the Glorot and Bengio [2010] method. For the three architectures, learning rate is set to 0.01, momentum to 0.9, with batch size of 100 points.

Training. As illustrated in Table 1, the three training sets contain mostly non-edge surfaces. The small number of sharp and smooth edge samples (4-14% of the points) implies that we are in presence of a highly unbalanced training set biased towards the non-edge class. We handle this issue by generating balanced batches of points during training and validation. Batches are balanced by replicating sharp and smooth edge points until the number of points in each class is equal. We recorded training curves for PCEDNet on Default and ABC datasets (see Figure 8 and Table 6). As we can see in Figure 8-bottom-left, the behavior of the validation loss may appear unusual, even

[1]Church: https://sketchfab.com/3d-models/christ-church-and-dublin-city-council-b5f6bcce8ebc44a3b4bbb6b0fef067b3. Train Station: https://sketchfab.com/3d-models/station-rer-6c636ca4793345e8ae12beb97b7d6359

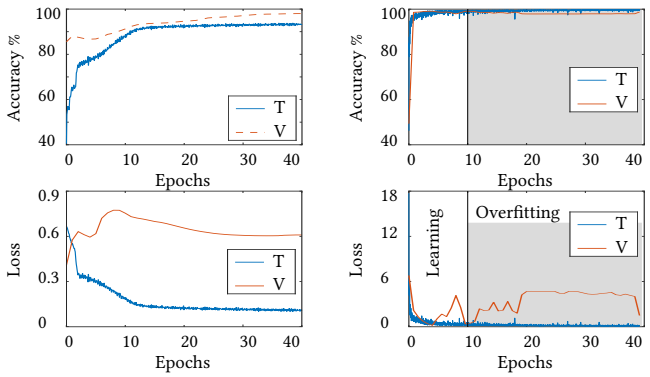

Fig. 8. PCEDNet: learning curves measured on Training (T) and Validation (V) sets on the (left) Default and (right) ABC. For the latter, learning stops automatically at epoch 10, when loss and accuracy stop to be improved.

though in practice, it is sometimes observed in machine learning. During the first epochs, starting from a random initialization, the network does not generalize yet on the validation set, which may result in an increase of the loss. Then, during the following epochs, even though decreasing, the loss may remain higher than its value at the random initialization, which may be due to the strong impact of wrong predictions on the loss value. The validation accuracy keeps increasing during training, probably because it is less sensitive to wrong predictions than the loss. Overall, training usually takes a couple of minutes for PCEDNet, a dozen for the baseline FC, and more than one hour for CNN.

	GLS	Training	Total
PCEDNet (Default)	19.32 s	2:52 m	3:11 m
FC (Default)	19.32 s	13:58 m	14:17 m
CNN (Default)	19.32 s	1:19:08 h	1:19:27 h
PCEDNet-2C (ABC)	2:11 m	20:00 m	22:11 m
FC-2C (ABC)	2:11 m	2:01:05 h	2:03:16 h
CNN-2C (ABC)	2:11 m	9:00:00 h	9:02:11 h
PCEDNet-2C (SHREC)	4 s	28.09 s	32.09 s
FC-2C (SHREC)	4 s	10:03 m	10:07 m
CNN-2C (SHREC)	4 s	26:01 m	26:05 m

Table 2. Training times of our networks on the different datasets.

5 RESULTS

In this Section, we first validate our choice of input parameters with an ablation study (Section 5.1) and discuss the number of input scales (Section 5.2). We then compare our network with our baseline (i.e. CNN and FC), with ECNet [Yu et al. 2018] and PCPNet [Guerrero et al. 2018] and two geometric feature detection methods, i.e. the covariance analysis method proposed by Bazazian et al. [2015] (CA) and the Feature Edge Estimation (FEE) implemented in the CGAL Library [Alliez et al. 2021; Mérigot et al. 2011]. We also report the evaluation of PIENet and ECNet performed on the ABC dataset by Wang et al. [2020]. For clarity, we denote A(D) the approach A trained on dataset D, e.g., PCEDNet trained on Default is denoted PCEDNet (Default). We present training and classification times (Section 5.3), quantitative comparisons (Section 5.4), visual comparisons (Section 5.5), networks behavior on noisy data (Section 5.6), the way our PCEDNet can be used for interactive learning (Section 5.7) and some complementary experiments (sensibility to variation in sampling, choice of the maximum scale and choice of the surface reconstruction method - Section 5.9). All of the Figures of the next sections are available in full resolution on the joined website homepage, with left/right interactive comparisons between the methods. Results of the quantitative experiments on each dataset are documented in dedicated webpages presenting interactive distribution plots, histograms, tables, and a 3D point cloud viewer.

Comparison metrics. We compare positive and negative matches of the classifications w.r.t. ground truth by measuring True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). To do so, we consider the following metrics: Precision, Recall, F1-Score, Intersection over Union (IoU), Matthews Correlation Coefficient (MCC), and Accuracy, whose formulations are given in Appendix B. Note that F1 and IoU scores ignore the True Negatives and thus may be misleading for unbalanced classes, which is our case in general, i.e. the number of TP edge points is in general very small in comparison to the TN non-edge points. It is however reported for comparison as F1 and IoU are commonly used in similar contexts and it still provides a score aggregating Precision and Recall. Also note that the IoU metric is known to be very correlated to the F1 score (see their Equations in Appendix B) and they differ as the IoU is more sensitive to high errors (i.e. it is more affected by worst cases) while the F1 score tends to measure an average performance. In our case (i.e. with a low number of highly unbalanced classes), the number of TP significantly influences the result evaluation and Chicco and Jurman [2020] explain in details how a measure of the correlation between observation and prediction relying on all TP, TN, FP, FN, as the MCC, better represent the quality of the network classification.

Comparison of classifications with two and three classes. As previously stated, PCEDNet outputs three classes corresponding to sharp-edges, smooth-edges and non-edges. However if edges are defined as sharp features only (e.g. the ABC dataset), results from the smooth-edge class does not bring significant quantitative information and only the sharp-edge class is of interest. In that case, we consider the sharp-edge as the unique positive class, and merge the smooth-edge and non-edge classes as the negative class. When a fuzzier definition is used for edges, as for the SHREC dataset where both rounded and sharp edges co-exists, we present additional results where the positive class is defined either as sharp edge only, or, as the union of sharp-edges and smooth-edges classes. Accordingly, three-classes approaches are trained on two-classes datasets as if no vertex is labelled as smooth-edge, but only has sharp-edge or non-edge.

5.1 Ablation study

In this Section, we validate both the choice of input parameters and the number of output classes (i.e two or three) of PCEDNet. Quantitative results are given respectively in Tables 3 and 4.

SSM. Our Scale-Space Matrix is composed of 6 parameters computed by differentiating the GLS implicit surface both in scale and

		2-cube (σ)						Angle
		0	0.01	0.02	0.03	0.12	0.14	
All GLS derivatives (28 parameters)	P	0.529	0.305	0.348	0.413	0.310	0.340	0.327
	R	0.415	0.424	0.454	0.481	0.427	0.45	0.5
	F1	0.465	0.355	0.394	0.445	0.360	0.387	0.396
	MCC	0.435	0.310	0.353	0.407	0.315	0.345	0.400
Invariant to rigid transformations (7 parameters)	P	**0.983**	**0.839**	**0.866**	**0.882**	**0.845**	**0.853**	**0.871**
	R	0.823	0.904	0.921	0.912	0.904	0.919	**0.984**
	F1	0.897	0.871	0.893	0.897	0.874	0.885	**0.925**
	MCC	0.870	0.828	0.858	0.864	0.832	0.848	**0.924**
Our configuration (6 parameters)	P	0.966	0.807	0.832	0.868	0.817	0.826	0.724
	R	**0.897**	**0.960**	**0.970**	**0.934**	**0.961**	**0.964**	**0.984**
	F1	**0.931**	**0.877**	**0.896**	**0.901**	**0.884**	**0.890**	0.835
	MCC	**0.908**	**0.839**	**0.864**	**0.869**	**0.847**	**0.856**	0.840
Remove scale (4 parameters)	P	0.360	0.217	0.250	0.265	0.217	0.233	0.428
	R	0.340	0.489	0.513	0.696	0.475	0.495	0.416
	F1	0.350	0.301	0.337	0.385	0.298	0.317	0.423
	MCC	0.305	0.260	0.299	0.373	0.256	0.277	0.419
Remove curvature (3 parameters)	P	0.173	0.103	0.115	0.355	0.105	0.109	0.242
	R	0.292	0.290	0.277	0.783	0.290	0.286	0.222
	F1	0.217	0.152	0.163	0.489	0.155	0.158	0.232
	MCC	0.152	0.079	0.094	0.484	0.084	0.087	0.227

Table 3. Precision, recall, F1 and MCC computed on the 2-cube (with a Gaussian noise generated with different value of σ) and Angle models of the Default evaluation set (see Table 1) for the different parameterization of PCEDNet.

space. In order to measure the relevance of our parameterization, we tested four alternative sets of parameters, ranging from 28 to 3 parameters per scale. For each scenario, the PCEDNet architecture is modified as follows: the number of weights of the four first layers is set according to the number of input parameters, as well as the connection between the fourth and fifth layers. We define the different parameter sets as follows:

- 28 parameters: taking all the derivatives of the GLS descriptor in scale and space,
- 7 parameters: keeping only the derivatives that are invariant to rigid transformations (rotation, translation and scale), leading to $\left[\tau_i^t \quad \kappa_i^t \quad k1_i^t \quad k2_i^t \quad \frac{\delta\tau_i^t}{\delta t} \quad \frac{\delta\kappa_i^t}{\delta t} \quad \phi(S_{\mathbf{p}_i}^t)\right]$,
- 6 parameters (selected PCEDNet input): removing $k2$, which is linearly dependent to $k1$ and κ,
- 4 parameters: removing scale information from PCEDNet input, leading to $\left[\tau_i^t \quad \kappa_i^t \quad k1_i^t \quad \phi(S_{\mathbf{p}_i}^t)\right]$
- 3 parameters: removing curvature information from PCEDNet input, leading to $\left[\tau_i^t \quad \frac{\delta\tau_i^t}{\delta t} \quad \phi(S_{\mathbf{p}_i}^t)\right]$.

We trained these five parameter configurations on Default and reported in Table 3 the Precision, Recall and F1 scores when applied on the evaluation set. As we compare 3-classes approaches on a 3-classes dataset in this experiment, we sum-up TP and FP for both the sharp-edge and smooth-edge labels, and use non-edge for negatives. Our choice of 6 parameters is validated as it always has the higher Recall, F1 and MCC, while providing a good precision in comparison with the other sets of parameters.

Number of output classes. We measure the impact of the smooth-edge class on the quality of the sharp-edge classification by implementing a version, denoted PCEDNet-2C, in which the decision layers return scores for 2 classes instead of 3. We trained PCEDNet-2C on the Default dataset by considering points labelled as sharp-edge for one class and as the union of non-edge and smooth-edge for the other class. We report in Table 4 the scores obtained by PCEDNet and PCEDNet-2C on the evaluation set. We can see that the addition of the third class allows PCEDNet to obtain higher scores than PCEDNet-2C, for all the metrics. Note that one class could have been the union of sharp and smooth edges and the other the non-edges. This increases the false positive rate and leads to the same best parameterization, with overall lower performance. This gain in performance can be due to the unequal sampling of point clouds and the lack of robustness of geometric thresholding, that makes the accurate labeling of edges very tedious. The addition of the smooth-edge class enables a more conservative labeling of points.

		2-cube (σ)						Angle
		0	0.01	0.02	0.03	0.12	0.14	
PCEDNet-2C (Default)	P	0.576	0.298	0.366	0.364	0.308	0.330	0.435
	R	0.564	0.610	0.652	0.689	0.611	0.614	0.278
	F1	0.570	0.400	0.469	0.477	0.410	0.430	0.339
	MCC	0.539	0.374	0.445	0.458	0.383	0.402	0.344
PCEDNet (Default) (Ours)	P	**0.966**	**0.807**	**0.832**	**0.868**	**0.817**	**0.826**	**0.724**
	R	**0.897**	**0.960**	**0.970**	**0.934**	**0.961**	**0.964**	**0.984**
	F1	**0.931**	**0.877**	**0.896**	**0.901**	**0.884**	**0.890**	**0.835**
	MCC	**0.908**	**0.839**	**0.864**	**0.869**	**0.847**	**0.856**	**0.840**

Table 4. Precision, Recall, F1 and MCC computed on the 2-cube (with different Gaussian noise) and Angle models of the Default evaluation set (presented in Table 1) for PCEDNet and its two-classes implementation PCEDNet-2C. As we can see, the addition of the third class improves the classification quality of the sharp-edge class.

5.2 Different number of input scales

We now discuss the performances in training and processing times, and edge detection capabilities of our PCEDNet for 4, 8, 16, 32, 64 and 128 input scales. We modify our architecture by changing the number of pairwise concatenation/processing layers (respectively 1, 2, 3, 4, 5, 6) in its first stage (Figure 3-left, green dense layers). For all configurations, the training curves of our network on Default and ABC are similar to those presented in Figure 8. We report the timings obtained with the different number of input scales in Table 5 and we illustrate the results in Figure 9 on the Pisa Cathedral and on a model of the ABC dataset. In Figure 10 we also show the distribution of precision/recall scores for each number of input scales. Not surprisingly, computation times increase with the number of input scales. Depending on the number of different scales representing important features in a model, 8 to 32 scales may be used. For instance, as illustrated in Figures 9-bottom and 10, on the ABC dataset where edges are of similar scale and sharpness, 8 scales may be enough while Figure 9-top shows on the Pisa Cathedral model that 32 scales would improve the results on large models with several scales of interest. As a default architecture, we present PCEDNet with 16 scales as a good compromise between speed and detection rate. It provides both accurate detection on all the models we tested while enabling interactive training sessions (as presented in Section 5.7), however, if less emphasis is put on computation times, the input configuration with 32 scales may be considered. In our experiments, 64 and 128 input scales did not improve over 32 scales while being significantly slower.

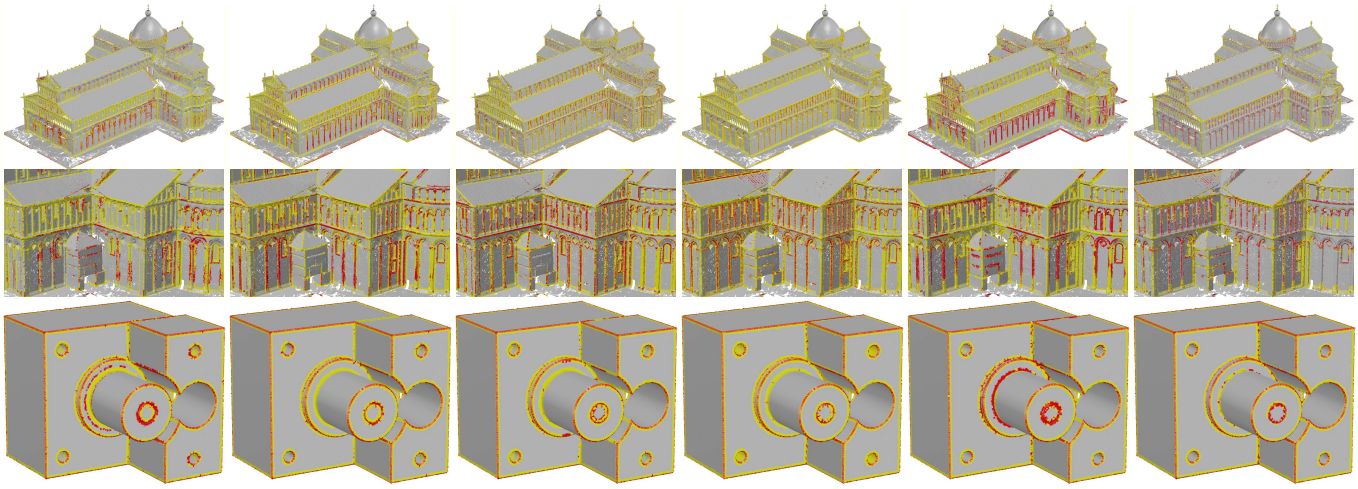

Fig. 9. Different classifications produced by PCEDNet with, from left to right, 4, 8, 16, 32, 64 and 128 input scales, on the Pisa Cathedral model (two top rows) and model 7024 of the ABC dataset (bottom row).

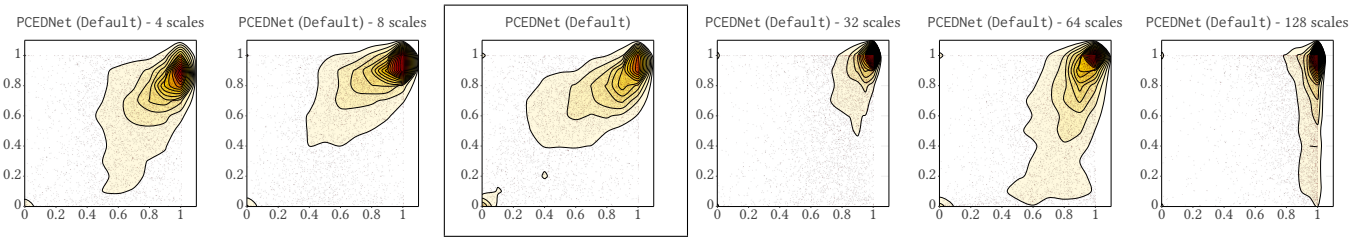

Fig. 10. Distribution of the Precision (abscissa)/Recall (ordinate) scores displayed as a scatter plot (each point cloud is a sample) and its associated density function for the ABC dataset, for the different number of input scales in PCEDNet (from left to right : 4, 8, 16, 32, 64 and 128 input scales) (except when explicitly stated, we use 16 scales for all our experiments).

	4	8	16	32	64	128
Training on Default	4.32 (47.17) s	0:06 (1:30) m	0:19 (3:11) m	3:59 (9:44) m	4:10 (15:37) m	6:18 (29:23) m
ABC	6:50 (44:38) m	0:11:20 (1:30:12) h	0:25:30 (3:00:30) h	0:51:02 (6:01:02) h	1:40:00 (12:00:00) h	3:20:00 (20:40:00) h
Pisa Cathedral	1.34 (13.80) s	2.23 (27.65) s	4.12 (58.61) s	0:07 (1:55) m	0:15 (3:44) m	0:31 (7:19) m

Table 5. Timings for training (first row) and processing (second and third) point clouds using PCEDNet for the different number of input scales (from 4 to 128). We report the network training/classification time followed in brackets by the total_time with total_time = GLS_precomputation+processing_time. For the ABC dataset (second row) we provide the times for processing the 7167 models.

5.3 Training and classification times

In this section we compare the learning and evaluation speed of our specialized approach to more versatile networks based on PointNet [Qi et al. 2017], the state of the art solution for point-based learning and semantic shape analysis. Among the different possibilities, we selected ECNet [Yu et al. 2018] , PCPNet [Guerrero et al. 2018] as they can be easily adapted to our more geometric and focused problem. We also include the evaluation of the edge classification presented on PIENet on the ABC dataset by Wang et al. [2020].

Experimental setup for compared approaches. For PCPNet, we ran a specialization training in order to adapt its original output from point normals estimation to point classification. We removed the output normalization from their architecture, and trained the last layers of the network (using the default hyperparameters provided by the authors) using our data (400 epochs on Default, and 50 epochs on ABC). This allows us to take advantage of PCPNet original training for the first layers. For ECNet, we use the pre-trained version provided by the authors (denoted ECNet (EC)), and we report the evaluation provided in Wang et al. [2020] for ECNet (ABC) and PIENet (ABC), respectively the ECNet network and PIENet trained on ABC, when applied on the ABC dataset. Note that ECNet oversamples the input point cloud, and label each generated point as edge or non-edge. We retrieve the classification of the input point cloud by assigning to each input point the label of the closest output sample. PCPNet, ECNet and PIENet approaches require more computation power and memory than provided by our 10-cores Intel Xeon-E5 to handle large data sets as ABC. For training PCPNet, and evaluating both PCPNet and ECNet, we thus used an NVIDIA Quadro RTX

	PCPNet	PIENet (ABC)	PCEDNet
Default	40:00 m *	-	0:19 (3:11) m
ABC	19:00:00 h *	23:00:00 h *	2:11 (22:11) m

Table 6. Timings for training PCPNet on the Default dataset with three classes and on the ABC dataset with two classes. For PCEDNet, we report first the training time, followed in brackets by the total_time = GLS+training _time. We also remind the PCEDNet training times with the same number of classes for comparisons. The training time of PIENet is reported from Wang et al. [2020]. Timings with a * have been performed on recent high performance GPUs (e.g., NVIDIA TITAN X).

6000 with 24GB of G-RAM and PIENet was trained on an NVIDIA TITAN X GPU in [Wang et al. 2020]. This is reminded by a * in the text and in tables when we report timings.

Training. We trained PCPNet with three classes on our Default dataset (denoted PCPNet (Default)) and with two classes on ABC (denoted PCPNet-2C (ABC)). As shown by the loss plot in Figure 11-left, training of the deep architecture of PCPNet converge to relatively high loss on our Default training set. Also, despite several attempts, PCPNet couldn't stabilise on SHREC due to the high variability of the annotations. Training on a larger dataset such as ABC (see the loss plot in Figure 11-right) is better adapted and enables the convergence of PCPNet training with low loss. Regarding the timings, PCPNet and PIENet require significantly longer training than our approach (more than 500 times slower on ABC), as reported in Table 6.

Classification. In addition to PCPNet, ECNet and PIENet, we compare our approach with CA [Bazazian et al. 2015] (for all our experiments, we used 10 neighbors and threshold=0.65, except for Munich for which we used 20 neighbors to handle sparse sampling), FEE [Mérigot et al. 2011] (we specifically adjusted parameters for each dataset), our baselines CNN and FC. We report in Figure 12 the evolution of the classification timings (in logarithmic scale) when increasing the point cloud size. We observe for PCEDNet, FC and CNN that scale-space calculation (which can be done in pre-process) requires more time than the classification itself. It also illustrates how fast our compact network architecture is, compared to more complex and computationally intensive architectures as PCPNet, ECNet, and PIENet. Finally, PCPNet and ECNet require more than 24 GB G-RAM to handle large point clouds and we thus could not provide the timings for the classification of our larger models by these networks.

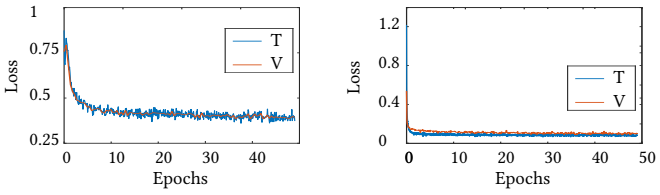

Fig. 11. PCPNet: learning curves measured on Training (T) and Validation (V) sets on the (left) Default and (right) ABC datasets.

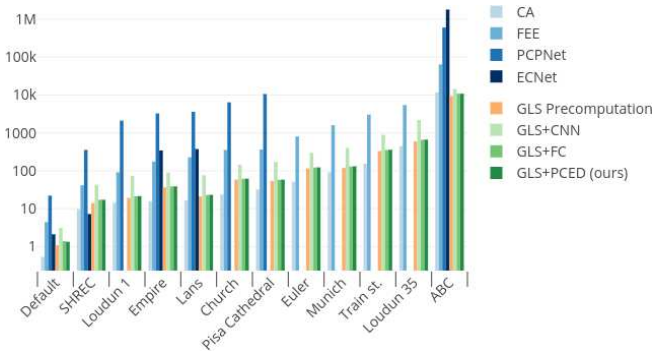

Fig. 12. Time (in seconds) required by the approaches presented in Table 10 to classify different models.

5.4 Quantitative evaluation

We now compare the classification produced by the aforementioned approaches with ground truth classification on datasets Default, ABC and SHREC.

Default *dataset.* Results are reported in Table 7 and illustrated in both Figure 14 and the accompanying website. Both CNN and FC are strong baselines getting significantly higher MCC scores than most methods from previous work. PCPNet (Default) shows a very limited detection of sharp edges. ECNet (EC) produces very precise

Method	Precision	Recall	**MCC**	F1	Accuracy	IoU
CA	0.490	0.880	0.506	0.628	0.752	0.457
FEE	0.341	0.814	0.471	0.480	0.879	0.316
FC (Default)	0.618	0.958.	0.682	0.753	0.849	0.604
CNN (Default)	0.546	0.955	0.623	0.694	0.807	0.532
PCPNet (Default)	0.722	0.198	0.301	0.310	0.789	0.183
PCEDNet (Default)	0.826	0.952	**0.857**	**0.890**	0.946	**0.802**
PCEDNet-2C (Default)	0.364	0.611	0.402	0.430	0.908	0.274
ECNet (EC)	**1.000**	0.457	0.656	0.620	**0.960**	0.450
PCEDNet-2C (EC)	0.201	**0.991**	0.383	0.334	0.759	0.200

Table 7. Quantitative evaluation on Default: median scores (see scores of the other approaches in the accompanying website).

Method	Precision	Recall	**MCC**	F1	Accuracy	IoU
CA	0.348	0.944	0.520	0.504	0.881	0.338
FEE	0.135	**1.000**	0.062	0.235	0.278	0.132
FC (Default)	0.452	0.679	0.469	0.513	0.921	0.327
CNN (Default)	0.530	0.995	0.648	0.689	0.922	0.519
PCEDNet (Default)	0.746	0.745	0.688	0.713	0.966	0.548
PCEDNet-2C (Default)	0.662	0.936	0.708	0.730	0.958	0.574
ECNet (EC)	0.425	0.648	0.423	0.460	0.910	0.294
PCEDNet-2C (EC)	0.378	0.849	0.480	0.503	0.888	0.334
ECNet (ABC)	0.487	0.573	-	0.526	-	0.356
PIENet (ABC)	0.692	0.858	-	0.766	-	0.622
FC-2C (ABC)	0.470	0.871	0.555	0.581	0.920	0.408
CNN-2C (ABC)	0.507	0.983	0.646	0.662	0.928	0.491
PCPNet-2C (ABC)	**0.954**	0.756	0.797	0.807	**0.979**	**0.668**
PCEDNet-2C (ABC)	0.735	0.984	**0.808**	**0.822**	0.970	0.597

Table 8. Quantitative evaluation on ABC: median scores (see scores of the other approaches in the accompanying website). ECNet (ABC) and PIENet (ABC) are reported from Wang et al. [2020]

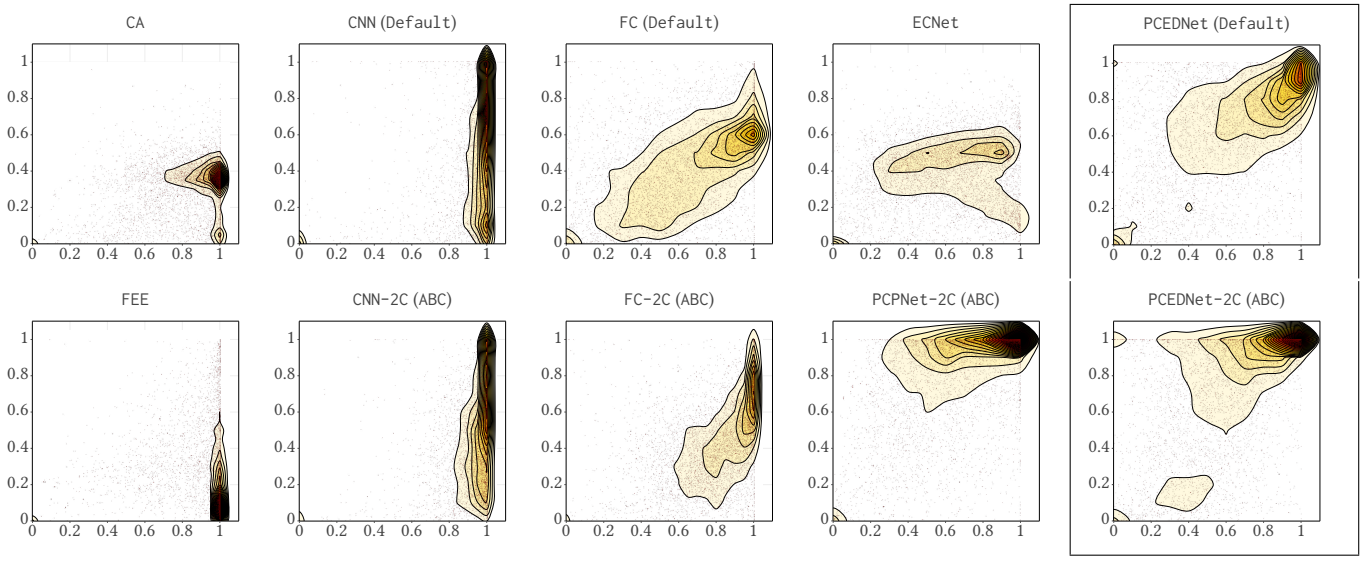

Fig. 13. Distribution of the Precision (abscissa)/Recall (ordinate) scores displayed as a scatter plot (each point cloud is a sample) and its associated density function for the ABC dataset. Interactive plots are provided in the joined website. All maps share the same color scale.

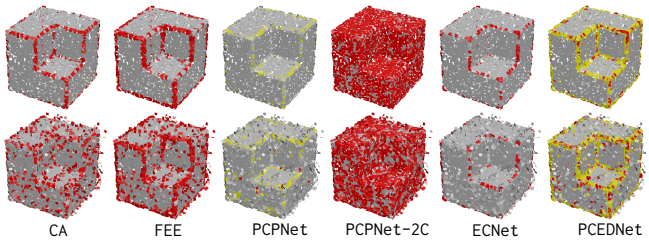

Fig. 14. Classification on Default, model 2-cube: $\sigma = 0$ (top) and 0.14 (bottom). See results of the other approaches in the accompanying website.

classifications, but overall misses to classify 50% of the points labeled as edges in ground truth (high precision and low recall). This classification by ECNet (EC) of a small subset of the edge points is emphasized by the low F1 and MCC scores that aggregate precision and recall. Regarding single-scale fitting-based approaches (i.e. CA and FEE), both approaches fail at handling outliers (for FEE, we used the following parameters: r= 0.025, R= 0.05 and th= 0.16). PCEDNet (Default) produces the best results on this dataset, getting significantly higher indicators than other approaches. Overall, PCEDNet trained on the Default dataset provides the best balance between precision and recall with the highest F1 and, more importantly, MCC scores.

Finally, we observe that PCEDNet-2C (EC) provides a better recall and a lower precision than ECNet (EC), with a lower MCC on Default. The same behavior is observed on the SHREC (Table 9) and ABC (Table 8) datasets, with a higher MCC for PCEDNet-2C on the ABC dataset. It illustrates the more conservative behavior of PCEDNet-2C (EC) while ECNet (EC) tends to detect subsets of correct edges.

ABC *dataset.* Due to the large size of ABC (7167 models) we present in Figure 13 (and in the accompanying website) the results as a precision/recall scatter plots, which provide a more readable overall view of the performance of the different approaches. For each approach, we plot each 3D model as a point sample, and display the resulting density maps as 2D-contour maps. We use the Hot colormap, with density values ranging from 50 (light yellow) to 1000 (dark red). A perfect classification result would lead to a Dirac distribution centered at location (1,1) and with magnitude 7167. Most of the analysed approaches are producing classifications with high recall, but differ by their capacity to reach high precision. Table 8 shows the median statistical scores obtained by each method on the whole dataset. Single-scale fitting-based approaches, e.g., CA and FEE (we use r= 0.4, R= 0.8 and th= 0.16 for this dataset) exhibit very high recall but low precision (lower than 0.4) with the classification of a lot of FP. As shown in supplementary materials, these approaches produce good results for very clean and simple geometries, but fail at analyzing thin objects (i.e. neighborhood might include two pieces of opposite surfaces, see FEE in Figure 15) and other intricate shapes. CNN follows a similar pattern, but is able to produce better classifications with high precision (MCC $\approx$ 0.65). FC produces results with lower quality (MCC $<$ 0.55) and higher dependency to the training set. Both FC (Default) and FC-2C (ABC) are however competitive compared to state of the art methods.

Overall, the evaluation of ECNet (EC) (in terms of MCC, F1 and Accuracy) is better than fitting-based approaches. ECNet (ABC) exhibits a higher precision and a lower recall than ECNet (EC), resulting in better general behavior. It remains however less efficient than our baselines trained on ABC, which, in terms of F1 score, are also less effective than PIENet (ABC). All the variants of our PCEDNet produce varying precision and recall but similar F1 scores above 0.7, and PCEDNet-2C (ABC) shows the best results with an MCC and F1 scores

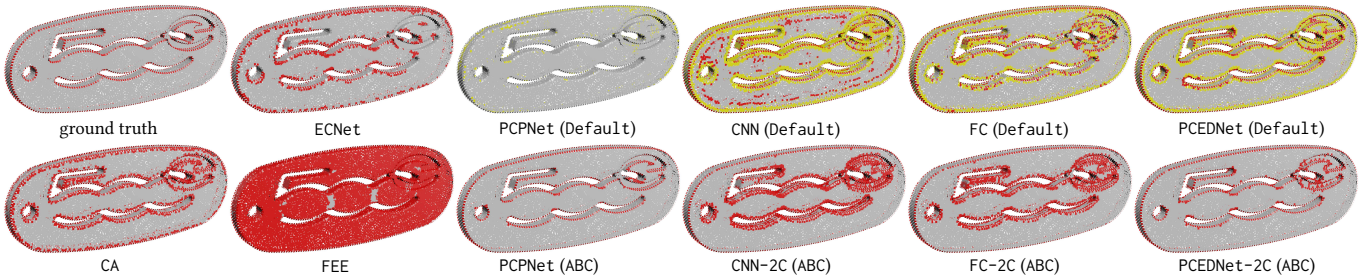

Fig. 15. Model 7029 of the ABC dataset.

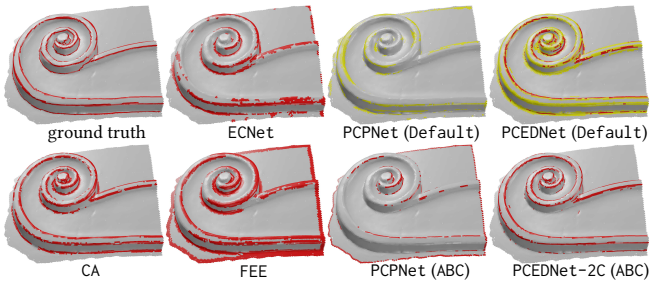

Fig. 16. Model 7 of the SHREC dataset. Results for CNN and FC can be found in the joined website.

above 0.8. Despite the large difference between the training sets of Default and ABC, our approach PCEDNet (Default) produces classifications with better MCC than competitors (MCC = 0.688), but at the cost of a recall loss. As shown in the joined website, the classifications produced by both PCEDNet (Default) and PCEDNet-2C (ABC) remain visually convincing despite quantitative differences against the ground truth. A notable exception to the global trend high-recall/variable-precision is PCPNet-2C (ABC), which exhibits higher precision/lower recall than any other approach. It also reaches comparable scores as our approach (e.g. MCC, F1, Accuracy), but visual inspection reveals that PCPNet-2C (ABC) tends to miss some edges entirely (lower recall), while our approach finds relatively thicker edges (lower precision). Also, PCPNet-2C (ABC) requires 3 days to process the entire dataset (instead of 3 hours for PCEDNet-2C (ABC)).

Method	Prec.	Recall	**MCC**	F1	Accuracy	IoU
CA	0.434	0.449	0.390	0.442	0.876	0.284
FEE	0.191	0.527	0.151	0.278	0.727	0.160
PCPNet (Default)	0.000	0.000	0.000	0.009	0.893	0.000
FC (Default)	0.392	0.538	0.391	0.455	0.880	0.294
FC (Default)$^+$	0.313	0.909	0.428	0.461	0.793	0.300
CNN (Default)	0.468	0.928	**0.570**	**0.611**	0.911	**0.440**
CNN (Default)$^+$	0.299	**0.945**	0.429	0.446	0.802	0.287
PCEDNet-2C (Default)	0.462	0.623	0.455	0.510	0.868	0.342
PCEDNet (Default)	**0.495**	0.377	0.344	0.437	0.916	0.280
PCEDNet (Default)$^+$	0.349	0.898	0.444	0.489	0.814	0.323
ECNet (EC)	0.365	0.503	0.341	0.397	0.845	0.248
PCEDNet-2C (EC)	0.254	0.747	0.285	0.365	0.676	0.223
FC-2C (SHREC)	0.406	0.715	0.392	0.480	0.849	0.316
CNN-2C (SHREC)	0.207	0.881	0.202	0.248	0.622	0.142
PCEDNet-2C (SHREC)	0.441	0.872	0.426	0.522	0.874	0.353

Table 9. Quantitative evaluation on SHREC: median scores. For 3-classes methods marked with $^+$, we compare both the smooth and sharp edges classes with the ground truth, i.e. Positives=Sharp+Smooth.

SHREC *dataset.* This dataset has been originally designed to evaluate curve detection algorithms on organic and relatively smooth objects with a comparison to human labelling. In addition, some objects have strong semantics (e.g. human face), and we observe that the ground-truth classification seems to take into account this semantic rather than strictly respecting geometric features. This makes this SHREC dataset very challenging to classify, as illustrated in Table 9 by the lower scores obtain by all approaches, in comparison to the other datasets. In particular, PCPNet (Default) fails at classifying edge points (MCC = 0). On this dataset, ECNet tends to produce noisy and thick edges, getting lower score than CA. CA and FEE (we use r= 2, R= 4 and th= 0.16 for this dataset) produce more convincing results, with CA reaching MCC= 0.39. CNN (Default) get the best scores on this dataset (both MCC and F1), but the same architecture trained on SHREC gets lower detection rate. Both FC and PCEDNet show better stability w.r.t. the training set (MCC $\approx$ 0.4 for all variants), but lower score than CNN. Regardless to the quantitative analysis, the results produced by PCEDNet (Default) are very interesting on this dataset. By enforcing the detection of both smooth and sharp edges, it better adapts to the smooth nature of the analysed objects and it provides very pertinent results as illustrated in Figure 16.

5.5 Visual evaluation

In this section we present visual results on acquired datasets without ground truth labelling. All these scenes are presented in more details in the accompanying video and in the joined website. For all the results presented in this section, we compare PCPNet (Default), PCPNet (ABC) and ECNet (EC) (on smallest scenes only), FC (Default), CNN (Default), PCEDNet (Default), CA and FEE. Table 10 shows precomputation (GLS) and classification times for all approaches. We see that processing time for PCEDNet (including precomputation) remains of the same order of magnitude than less robust approaches as CA and FEE, and outperforms PCPNet and ECNet (up to two orders of magnitude). Once trained, our approach also avoids the tedious parameter tuning of geometric methods.

Figure 17 represents a part of Lans without outliers and with very few noise. In this example, FEE detects larger scale edges cleanly, yet it misses fine details. CA detects edges, but it still struggles with surface noise and irregularities. PCPNet has an improved behavior over the noisy model and is able to detect smooth-edges. However, it still fails at classifying sharp-edges. Similar to FEE and PCPNet, FC extracts large scale edges leaving out finer details, which highlights the importance of scale separation layers in PCEDNet network

Dataset	#obj	#vert.	CA	FEE	PCPNet	ECNet	*GLS*	CNN (total)	FC (total)	PCEDNet (total)
Loudun 1	1	1M	14.30 s	1:30 m	35:00 m *	-	19.30 s	0:54 (1:13) m	1.80 (21.10) s	2.10 (21.40) s
Empire	1	1.2M	15.70 s	2:55 m	54:22 m *	5:43 m *	36.40 s	0:53 (1:29) m	1.80 (38.20) s	2.10 (38.50) s
Lans	1	1.23M	16.40 s	3:46 m	1:00:08 h *	6:13 m *	21.00 s	0:55 (1:16) m	1.80 (22.80) s	2.10 (23.10) s
Church	1	1.9M	23.40 s	5:53 m	1:46:40 h *	-	58.40 s	1:26 (2:24) m	0:02 (1:01) m	0:03 (1:01) m
Pisa Cathedral	1	2.5M	31.80 s	6:03 m	2:56:40 h *	-	53.60 s	1:57 (2:50) m	3.90 (57.50) s	4.70 (58.30) s
Euler	1	3.9M	51.20 s	13:22 m	-	-	1:55 m	3:00 (4:55) m	0:06 (2:01) m	0:07 (2:02) m
Munich	1	6M	1:31 m	26:33 m	-	-	1:59 m	4:41 (6:41) m	0:09 (2:09) m	0:11 (2:10) m
Train St.	1	12.45M	2:35 m	50:44 m	-	-	5:31 m	9:17 (14:49) m	0:19 (5:50) m	0:25 (5:57) m
Loudun 35	1	35M	7:29 m	1:30:25 h	-	-	9:52 m	26:38 (36:30) m	1:06 (10:59) m	1:09 (11:02) m
Paris rue Madame	1	20M	-	-	-	-	11:23 m	-	-	0:34 (12:31) m
Default	8	55k	0.53 s	4.41 s	22.11 s*	2.09 s*	1.08 s	2.03 (3.11) s	0.30 (1.38) s	0.25 (1.33) s
SHREC	15	654k	9.70 s	41.47 s	5:54 m *	2:48 m *	14 s	28.58 (42.58) s	3.10 (17.10) s	3.10 (17.10) s
ABC	7167	312.3M	3:15:00 h	17:50:00 h	7 d *	20 d *	2:35:00 h	1:25:00 (4:00:00) h	0:25:40 (3:00:40) h	0:25:30 (3:00:30) h

Table 10. Timing comparison for classification, where (s) stands for seconds, (m) for minutes, (h) for hours, and (d) for days. For datasets (e.g., Default, SHREC and ABC) we report the time needed to process all the models. The column *GLS* corresponds to the precomputation of the GLS descriptors. For columns CNN, FC and PCED, we report first the classification time, followed in brackets by the total_time = GLS+classification_time. Timings marked with * have been obtained using dedicated hardware (NVIDIA RTX 6000).

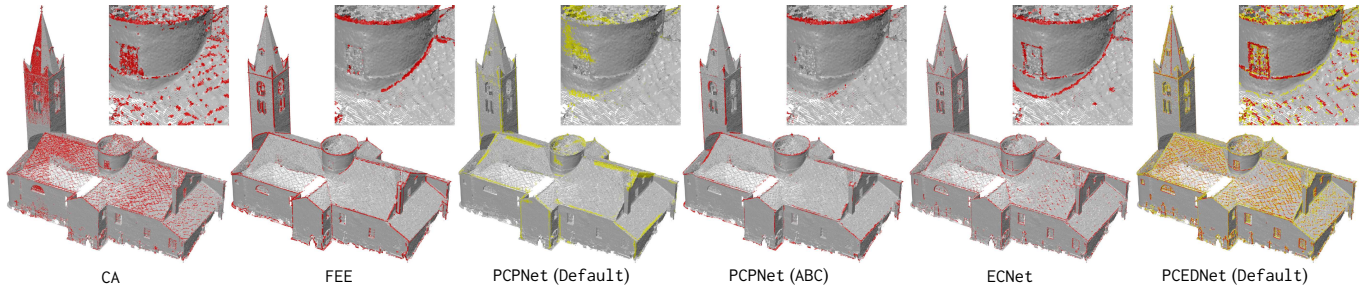

Fig. 17. Lans model. Results for CNN and FC can be found in the joined website.

architecture. Both CNN and PCEDNet produce visually convincing results, but CNN does not detect some fine details that are adequately detected by PCEDNet. ECNet also produces good results on this example, detecting most of the edges and some of the thin details.

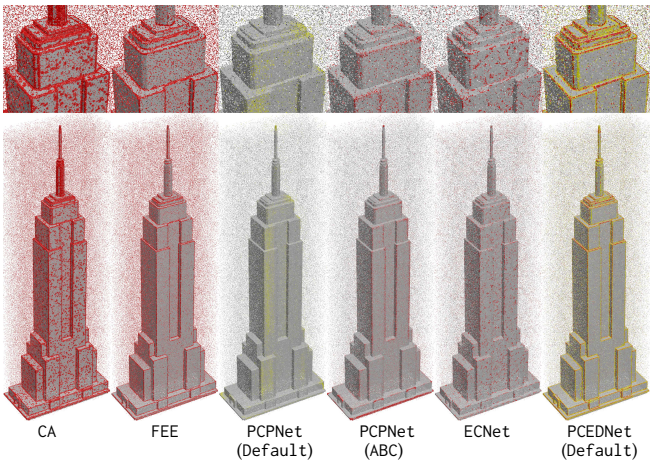

Fig. 18. Empire model. This point cloud contains a clean structure with a severe amount of outliers. Results for CNN and FC can be found in the joined website.

Figure 18 presents the behavior of each method when processing models with important noise and a lot of outliers. As observed on synthetic datasets, ECNet is strongly penalized by the outliers, and most of the flat areas are misclassified as edges. CA detects all the edge points on the synthetic models but also considers the outliers as edges. PCPNet classification is weakened by the outliers and does not provide any positive results. FC shows results similar to CA, the network is capable of extracting different edges, yet it is still very sensitive to outliers. CNN and PCEDNet both obtain a high recall on edges, with PCEDNet being the least sensitive to noise.

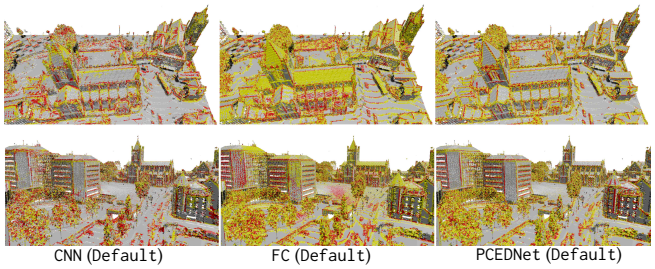

Fig. 19. Church. Results for FEE, FC and PCPNet can be found in the joined website.

CNN, FC and PCEDNet rely on the same parameterization. We thus provide in Figures 19 a closer comparison of the results produced by these architectures. In Figures 19, even though all networks generate similar results, the CNN and FC become less accurate when processing irregular surfaces as trees, cars, light poles, etc.

Figure 21 shows how our approach performs on a large point cloud (6M points) in comparison with CA and FEE (other approaches require too much memory/time). Munich exhibits common irregularities of acquired data, i.e. variable point densities, large gaps, scan noise, and volumetric objects (e.g., trees). PCEDNet (Default) produces cleaner detection and is less affected by acquisition artefacts than the other approaches.

5.6 Behavior on noisy data

Method	Precision	Recall	**MCC**	F1	Accuracy	IoU
ECNet (EC)	0.247	0.687	0.317	0.358	0.824	0.216
PCEDNet-2C (EC)	0.163	0.950	0.236	0.275	0.633	0.158
PCEDNet (Default)	**0.323**	0.375	0.267	0.319	**0.903**	0.185
PCEDNet (Default)$^+$	0.231	**0.920**	**0.358**	**0.366**	0.768	**0.222**
PCPNet-2C (ABC)	0.159	0.363	0.155	0.204	0.833	0.113
PCEDNet-2C (ABC)	0.121	0.793	0.181	0.213	0.626	0.117

Table 11. Quantitative evaluation on ABC dataset altered with a Gaussian noise of deviation $\sigma = 0.04$ (median scores). For PCEDNet trained on the Default dataset (with three classes), we show the results when considering the points of both the sharp and smooth edge classes as edges (4^{th} row, denoted with $^+$), and when considering only the sharp-edge class as edges (3^{rd} row).

We compare the classification produced by PCPNet-2C, ECNet, PCEDNet and PCEDNet-2C on the ABC dataset altered in the normal direction by a Gaussian noise of deviation $\sigma = 0.04$. See Figure 20-bottom for classification results on model 0133 (other models are included in joined website), and Figure 22 for scatter plots illustrating the distribution of the (Precision, Recall) scores on the entire dataset. As illustrated on Figure 20-bottom, the two approaches trained on ABC (PCPNet-2C and PCEDNet-2C) cannot disambiguate between noise and edges. This is due to the lack of noise in the ABC training set. Trained on more versatile data, both ECNet and PCEDNet (EC) are more robust to noise and still exhibit high recall, despite a loss in precision. In presence of noise, PCEDNet tends to classify inaccurate edge points as smooth edges. We thus propose to also evaluate the classification results considering the edge and smooth edge classes as positive matches (denoted with $^+$ in Table 11 and Figure 22). With this setting, PCEDNet (Default)$^+$ produces the best quantitative scores in Recall, MCC and F1, slightly overpassing ECNet, while being trained on smaller dataset and requiring 3 hours of computation instead of 20 days. PCEDNet (Default)$^+$, however, detects smoother edges, as on chamfers, that are avoided by ECNet, as illustrated in Figure 20 1^{st}, 3^{rd} and 4^{th} columns.

Figure 23 shows the capability of PCEDNet to classify edges on an acquired noisy model Paris rue Madame (we could not process this 12M-points dataset with PCPNet-2C and ECNet because of their hardware requirements). Figure 14-bottom provides a qualitative evaluation of the different classification methods on the model 2-cube of the Default dataset with a severe Gaussian noise (deviation $\sigma = 0.14$). In this case also, ECNet and PCEDNet exhibit a more convincing behavior.

5.7 Interactive learning

As shown in Table 6, PCEDNet requires very low training time, and is stable for very small training sets as our Default. We illustrate how such a flexible network can be trained interactively to better adapt to user wishes and data specificity.

Our interactive training system performs as follows: the user loads a point cloud on which GLS descriptors are precomputed. This is done in less than 20 seconds for 1M points. Then, the user manually labels some points of the two classes (sharp edge and non-edge) or the three classes (sharp edge, smooth edge and non-edge). We observed that the non-edge class should contain more points than the two others. This training sets are provided to the PCEDNet network initialized with random values, which learns for 5k epochs in around 10 seconds for approximately 10k input points. As for any other dataset, the number of points per class is automatically balanced during training. Once trained, the network classifies the whole point cloud in around 2 seconds per 1M points. Upon classification, the user can refine the training set according to the network output. If he does so, the network is trained again from a random initialization with the updated learning sets (as training is fast enough and modifying the previous training is less efficient).

We illustrate in Figures 1-c and 24, and in the accompanying video how a user can generate a high-quality classification corresponding to a specific edge definition defined by the annotations he provides during an interactive training session. For instance, in Figure 1-c-top, edges are defined sharp, while in Figure 1-c-bottom, edges are defined as a large scale feature including chamfers.

5.8 Energy efficiency

Energy consumption is a critical aspect of processing methods when considering their application in embedded systems or their environmental impact on global warming. In addition to evaluating the results in terms of classification and timings, we present an empirical study of the energy consumption of the different approaches for training and classification, based on their hardware requirement and processing time.

Methodology. Ideally, energy consumption should be measured during processing in order to account for the modulation of processor charge. Since we do not have such measuring device yet, we propose to roughly estimate the energy consumption based on the Thermal Design Power (TDP) of the processing unit (CPU or GPU), which measures the energy consumption under high workload. This thus provides over-estimations of the energy consumption that remain useful for the comparison of approaches with significant differences (in our case, up to several orders of magnitude, as can be seen in Tables 12 and 13). The TDP is expressed in Watts and it is provided by manufacturers. We denote E_{K} the energy, expressed in Joule (1J = 1Ws), used for processing 1K points, and we define it as follows:

$$E_{\mathrm{K}} = TDP.t_{\mathrm{K}} \qquad \text{and} \qquad t_{\mathrm{K}} = \frac{Pt.1000}{N},$$

where t_{K} is the time required for processing 1K points, Pt is the processing time and N is the number of processed points in the time Pt. The TDP of our 10 cores Intel Xeon E5-2640 v4 @2.4 GHz is 90W, the one of an NVIDIA TITAN X GPU is 250W and for an

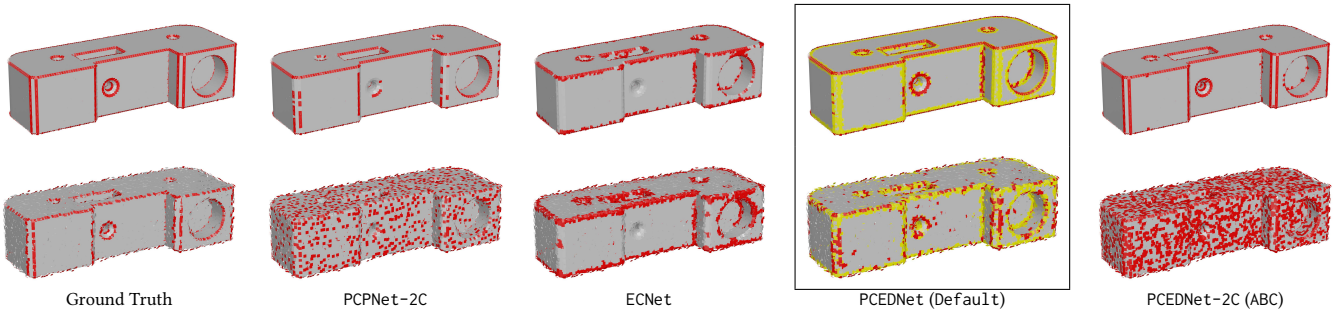

Fig. 20. Top: Classification produced by the different networks on model 0133 of the ABC dataset. Bottom: Same model altered with a random Gaussian noise of deviation $\sigma = 0.04$.

Training	PCPNet (Default)	ECNet (EC)	PIENet (ABC)	GLS	CNN (average)	FC (average)	PCEDNet (average)
Time t_K	8.34*	7.32*	25.77*	0.05	14.68 (14.73)	3.97 (4.02)	**0.43 (0.48)**
Energy E_K	2167.42*	1831.05*	6443.78*	9.16	1321.43 (1330.59)	357.60 (366.76)	**38.60 (47.76)**

Table 12. Times t_K (2^{nd} row) and processing unit energy consumption E_K (3^{rd} row) required for processing 1K points when training the different networks (1^{st} row) denoted as *name(training dataset). (average)* represents the average of the times obtained when training on the different datasets Default, ABC and SHREC. PCPNet is trained on an NVIDIA TITAN Quadro RTX 6000 GPU, and ECNet and PIENet are trained on an NVIDIA TITAN X GPU. The times and energy consumption for ECNet and PIENet are computed respectively from the statistics provided in [Yu et al. 2018] and [Wang et al. 2020].

Classification	CA	FEE	PCPNet	ECNet	PIENet (8K pts)	GLS	CNN	FC	PCEDNet
Time t_K	**0.015**	0.16	2.28*	1.32*	0.062*	0.023	0.043 (0.066)	0.0024 (0.0254)	**0.0026** (0.0256)
Energy E_K	**1.36**	14.79	592.87*	345.77*	15.63*	4.24	3.87 (8.11)	0.22 (4.46)	**0.23** (4.47)

Table 13. Times t_K (2^{nd} row) and processing unit energy consumption E_K (3^{rd} row) required for classifying 1K points with the different methods (1^{st} row). PCPNet and ECNet are run on an NVIDIA TITAN Quadro RTX 6000 GPU, and PIENet is run on an NVIDIA TITAN X GPU. The times and energy consumption for PIENet are computed from the statistics provided in [Wang et al. 2020].

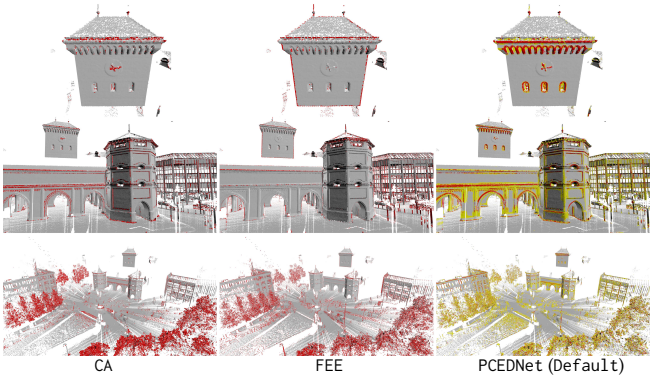

Fig. 21. Munich model (see Results for CNN and FC in the joined website).

NVIDIA Quadro RTX 6000 it is 260W. Based on Tables 1, 2, 6, 10, and the statistics provided by Wang et al. [2020] and Yu et al. [2018] for training and classification times and number of points, we show in Tables 12 and 13 the mean processing unit energy consumption E_K of each approach for respectively training and classifying 1K points.

When training (see Table 12), our approach requires significantly less energy than our baselines (one order of magnitude) and than very deep neural networks running on the GPU, e.g. PCPNet, ECNet and PIENet (up to two orders of magnitude). The low efficiency of CNN can be explained by its higher number of weights (about 4 times higher than PCEDNet) and by its implementation based on Tensorflow.

Similar trends can be observed during classification (see Table 13). We report the performance of PIENet provided by Wang et al. [2020] on models composed of around 8 thousand points. Other approaches are evaluated on larger models, up to several millions of points. Complementary experiments would be required to better evaluate PIENet classification efficiency on larger point clouds. CA requires a very little energy while PCEDNet and its baselines remains more efficient than FEE.

5.9 Complementary experiments

Variation of sampling (minimum scale). The features computed in the SSM are influenced by the minimum scale $s_{\min}$, estimated from the density of the point cloud. We show in Figure 25 how our approach behaves when changing the point cloud density by subsampling, while keeping the automatic estimation of $s_{\min}$. This is illustrated on the Loudun tower, initially composed of 35 millions points, and subsampled down to 15, 5 and 1 million points. Unsurprisingly, decreasing the resolution reduces the quantity of details and leads to thicker edges. However, our classification remains stable and PCEDNet still finds the position of edges, even for low densities, and without requiring the user to set any parameter value.

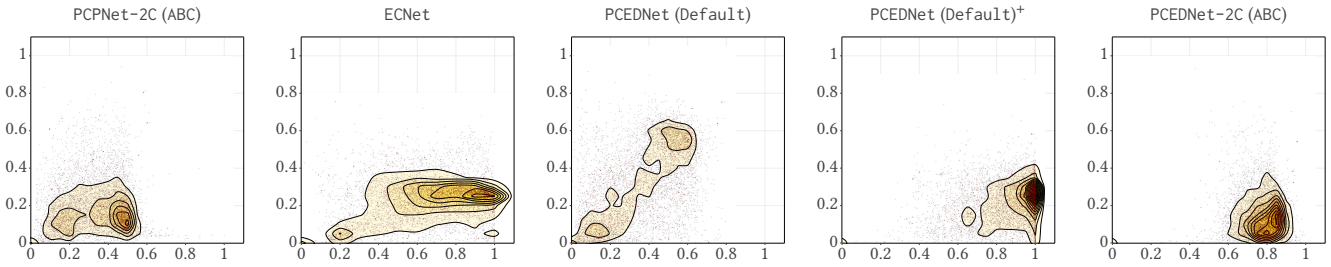

Fig. 22. Distribution of the Precision (abscissa)/Recall (ordinate) scores displayed as a scatter plot and its associated density function for the ABC dataset altered with a Gaussian noise of deviation $\sigma = 0.04$. From left to right, the detection produced by PCPNet-2C, ECNet, PCEDNet, PCEDNet (Default)$^+$ (see Table 7) and PCEDNet-2C trained on ABC.

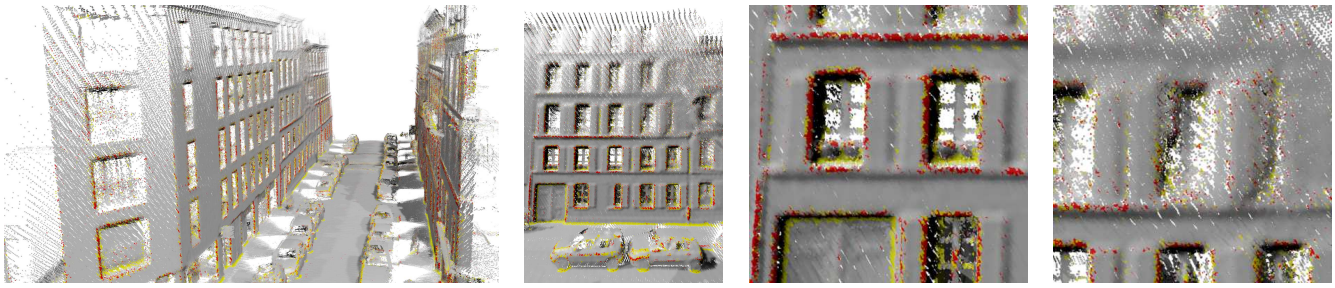

Fig. 23. Illustration of the classification produced by PCEDNet on the noisy acquired model Paris rue Madame.

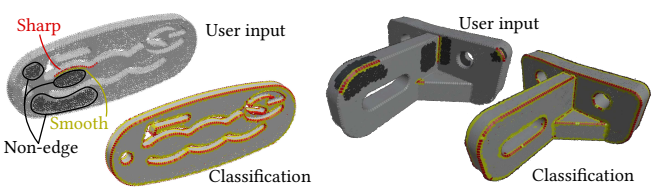

Fig. 24. Interactive training: user inputs and classification results.

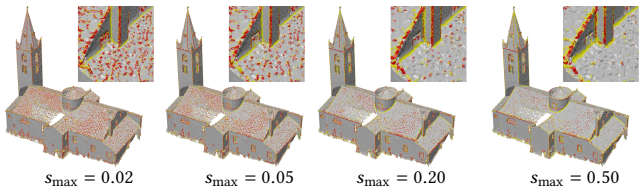

Fig. 26. Classification results when varying the maximum scale when computing the GLS for the analyzed object.

Variation of maximum scale. We also show in Figure 26 how our classification behaves when changing the maximum scale. Thanks to the logarithmic scale sampling and the scale invariance property of the SSM entries, our approach provides stable results even though the maximum scale is divided by 10 or multiplied by 2.5.

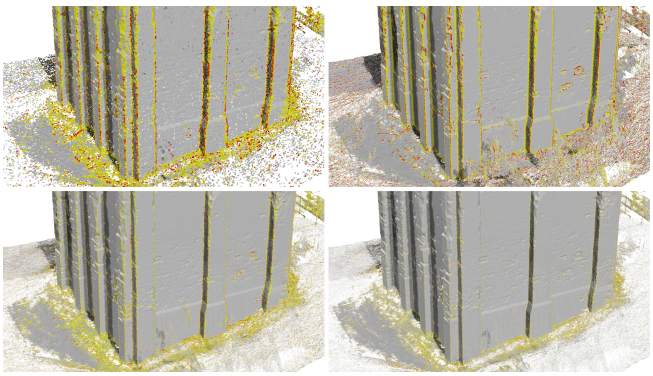

Fig. 25. Loudun: PCEDNet classification with different densities: 1, 5, 15 and 35 million points from top left to bottom right.

Surface reconstruction algorithm. Our network is parameterized using the Algebraic Point Set Surfaces (APSS) [Guennebaud and Gross 2007], which are known to be stable and reliable even at large scales. We illustrate in Figure 27 the performance of our classifier when computing the parameterization with different approaches: covariance plane fitting (also used in Bazazian et al. [2015]), plane-based point set surfaces [Alexa et al. 2001] and algebraic sphere fitting (same fitting as APSS but without the Moving Least Squares -MLS- projection). For each variant we compute similar values as our GLS descriptor, with derivatives estimated using finite differences. We clearly observe that the use of sphere fitting rather than plane fitting improves the classification, and the best results are always obtained using MLS projection. Eventually, recent more stable derivative evaluations of the APSS may also be experimented [Lejemble et al. 2021].

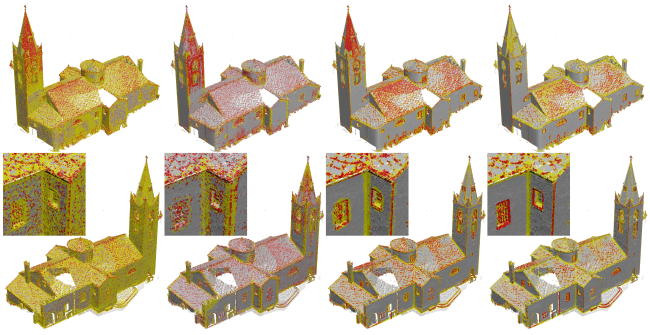

Fig. 27. Impact of the surface reconstruction algorithm on the classification. From left to right: covariance plane fitting (used by Bazazian et al. [2015]), point set surfaces [Alexa et al. 2001], algebraic sphere fitting, algebraic point set surfaces [Guennebaud and Gross 2007] (used in this work).

6 DISCUSSION

SSM and patch-based architectures. Our network relies on the SSM, a point cloud parameterization based on a set of GLS descriptors. An open question is whether the SSM could be efficiently used to provide additional parameters to the points used as input in patch-based point cloud deep processing architectures such as PointNet. This may be investigated, but we point out two issues that should be considered. The first is the significant memory overhead that will be generated on architectures that are already resource-demanding. The second is more conceptual: the goal of using the SSM is to capture local multi-scale shape descriptors at each point without having to explicitly handle their neighborhood in the network. This makes a lot of sense for a point cloud processed by point individually in the network, as we do for edges. This is less clear for a processing relying on point patches (e.g. for style or large features recognition). The first layers of these networks learn features from the spatial organisation of neighbor points and it is not easy to predict which additional parameter would be redundant (i.e. the information it brings is already captured by the network first layers) and which parameter would increase the network efficiency.

Need of local surface reconstruction. The need for locally reconstructing an approximating surface at different scales may be seen as a drawback of our method. In fact, this reconstruction does not allow to explicitly reconstruct or detect edges, and once the surface reconstructed, the problem remains unsolved. However, it enables the computation of the SSM shape descriptors that, used as input in our network, allow us to efficiently label edge points.

7 LIMITATIONS AND CONCLUSION

We introduced a new parameterization together with its dedicated neural network architecture (PCEDNet) specially designed for the classification of edges in point clouds. PCEDNet outperforms both state of the art methods such as CA, FEE, ECNet, PCPNet and PIENet, and baselines as CNN and FC. PCEDNet is also remarkably compact by being only composed of about 2100 weights. Given this small size, it is faster than the other approaches tested in this work. Both CNN and PCEDNet achieve similar very good results on synthetic point clouds, which shows that our parameterization based on GLS descriptors is very efficient to encode the point clouds features required for edge classification. Coupled with PCEDNet, it provides a very compact multi-scale representation that captures local geometric properties with reduced sensitivity with respect to noise, as can be seen on the various tests performed on real point cloud scans. The training and classification of our approach is also sufficiently fast to enable interactive training and classification from direct user inputs.

Limitations. APSS requires oriented normals, which may be tedious to compute accurately in some cases. For all our experiments with point clouds without normals, we obtained good classification results by estimating consistent normal vectors using Meshlab [Cignoni et al. 2008], without requiring human intervention. An interesting future work would be to consider alternative fitting techniques that do not require oriented normals [Chen et al. 2013b]. Regarding performance, SSM precomputation is currently the bottleneck of the approach, however we believe that a GPU implementation would enable near real-time classification, the most computationally intensive task being the neighborhood queries. Theoretically, SSM is by nature limited to surfaces only, alternative representations [Digne et al. 2018] might be considered in future work to handle lines and volumes.

Perspectives. Other experiments may be conducted to improve the network layout, for instance n by n scales concatenation. An interesting direction for future work could be the study of the extension of this architecture to other geometrical labelling tasks, but also to semantic analysis.

8 ACKNOWLEDGEMENTS

We would like to thank Archeovision for the courtesy of the Loudun dataset. This work has been partially funded by the ANR CaLiTrOp project (ANR-11-BS02-0006). Point clouds processing and visualization have been implemented in the Radium Engine [Mourglia et al. 2021]. Paris rue Madame is part of the *Paris-rue-Madame database: MINES ParisTech 3D mobile laser scanner dataset from Madame street in Paris.* MINES ParisTech created this special set of 3D MLS data for the purpose of detection-segmentation-classification research activities, but does not endorse the way they are used in this project or the conclusions put forward.

REFERENCES

Syeda Mariam Ahmed, Yan Zhi Tan, Chee-Meng Chew, Abdullah Al Mamun, and Fook Seng Wong. 2018. Edge and Corner Detection for Unorganized 3D Point Clouds with Application to Robotic Welding. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), 7350–7355.

Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. 2001. Point Set Surfaces. In *Proceedings of the Conference on Visualization '01* (San Diego, California) *(VIS '01)*. IEEE Computer Society, Washington, DC, USA, 21–28. http://dl.acm.org/citation.cfm?id=601671.601673

Pierre Alliez, Simon Giraudot, Clément Jamin, Florent Lafarge, Quentin Mérigot, Jocelyn Meyron, Laurent Saboret, Nader Salman, Shihao Wu, and Necip Fazil Yildiran. 2021. Point Set Processing. In *CGAL User and Reference Manual* (5.2.1 ed.). CGAL Editorial Board. https://doc.cgal.org/5.2.1/Manual/packages.html#PkgPointSetProcessing3

Dena Bazazian, Josep R Casas, and Javier Ruiz-Hidalgo. 2015. Fast and Robust Edge Extraction in Unorganized Point Clouds. In *Proceeding of International Confere on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 1–8.

Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. 2017. A Survey of Surface Reconstruction from Point Clouds. *Comput. Graph. Forum* 36, 1 (Jan. 2017), 301–329. https://doi.org/10.1111/cgf.12802

Alexandre Boulch. 2019. Generalizing Discrete Convolutions for Unstructured Point Clouds. (2019). https://doi.org/10.2312/3dor.20191064

Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. 2018. SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Computers & Graphics* 71 (2018), 189 – 198. https://doi.org/10.1016/j.cag.2017.11.010

Alexandre Boulch and Renaud Marlet. 2016. Deep Learning for Robust Normal Estimation in Unstructured Point Clouds. *Computer Graphics Forum* 35, 5 (2016), 281–290. https://doi.org/10.1111/cgf.12983 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12983

Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. 2017. Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks. In *Eurographics Workshop on 3D Object Retrieval*, Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov (Eds.). The Eurographics Association. https://doi.org/10.2312/3dor.20171047

M. M. Bronstein and I. Kokkinos. 2010. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1704–1711. https://doi.org/10.1109/CVPR.2010.5539838

Jiazhou Chen, Gaël Guennebaud, Pascal Barla, and Xavier Granier. 2013a. Non-Oriented MLS Gradient Fields. *Computer Graphics Forum* 32, 8 (2013), 98–109. https://doi.org/10.1111/cgf.12164 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12164

Jiazhou Chen, Gaël Guennebaud, Pascal Barla, and Xavier Granier. 2013b. Non-Oriented MLS Gradient Fields. *Computer Graphics Forum* 32, 8 (2013), 98–109. https://doi.org/10.1111/cgf.12164 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12164

Davide Chicco and Giuseppe Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* 21 (2020).

Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*, Vittorio Scarano, Rosario De Chiara, and Ugo Erra (Eds.). The Eurographics Association. https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136

Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. 2008. Where Do People Draw Lines? *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27, 3 (Aug. 2008).

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Joel Daniels II, Tilo Ochotta, Linh K. Ha, and Cláudio T. Silva. 2008. Spline-based feature curves from point-sampled geometry. *The Visual Computer* 24, 6 (2008), 449–462. https://doi.org/10.1007/s00371-008-0223-2

Kris Demarsin, Denis Vanderstraeten, Tim Volodine, and Dirk Roose. 2007. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer-Aided Design* 39, 4 (2007), 276–283.

J. Digne, S. Valette, and R. Chaine. 2018. Sparse Geometric Representation Through Local Shape Probing. *IEEE Transactions on Visualization and Computer Graphics* 24, 7 (July 2018), 2238–2250. https://doi.org/10.1109/TVCG.2017.2719024

Leandro A.F. Fernandes and Manuel M. Oliveira. 2012. A general framework for subspace detection in unordered multidimensional data. *Pattern Recognition* 45, 9 (2012), 3566 – 3579. https://doi.org/10.1016/j.patcog.2012.02.033 Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011).

Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. 2005. Robust Moving Least-Squares Fitting with Sharp Features. *ACM Trans. Graph.* 24, 3 (July 2005), 544–552. https://doi.org/10.1145/1073204.1073227

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

Gaël Guennebaud, Marcel Germann, and Markus Gross. 2008. Dynamic Sampling and Rendering of Algebraic Point Set Surfaces. *Computer Graphics Forum* 27, 2 (2008), 653–662. https://doi.org/10.1111/j.1467-8659.2008.01163.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2008.01163.x

Gaël Guennebaud and Markus Gross. 2007. Algebraic Point Set Surfaces. *ACM Trans. Graph.* 26, 3, Article 23 (July 2007). https://doi.org/10.1145/1276377.1276406

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. 2018. PCPNet: Learning Local Shape Properties from Raw Point Clouds. *Computer Graphics Forum* 37, 2 (2018), 75–85. https://doi.org/10.1111/cgf.13343

Stefan Gumhold, Xinlong Wang, and Rob Macleod. 2001. Feature extraction from point clouds. In *Proceedings, 10th International Meshing Roundtable*. 293–305.

Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. 2017. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. IV-1-W1. 91–98.

Timo Hackel, Jan D. Wegner, and Konrad Schindler. 2016. Contour Detection in Unstructured 3D Point Clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 2017. 3D Shape Segmentation with Projective Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6630–6639. https://doi.org/10.1109/CVPR.2017.702

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A Big CAD Model Dataset for Geometric Deep Learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 9593–9603.

Thibault Lejemble, David Coeurjolly, Loic Barthe, and Nicolas Mellado. 2021. Stable and efficient differential estimators on oriented point clouds. *Computer Graphics Forum* (July 2021). https://hal.archives-ouvertes.fr/hal-03272493

M. Li and K. Hashimoto. 2017. Curve Set Feature-Based Robust and Fast Pose Estimation Algorithm. *Sensors* 1782, 17 (2017).

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution On X-Transformed Points. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 820–830. http://papers.nips.cc/paper/7362-pointcnn-convolution-on-x-transformed-points.pdf

Yangbin Lin, Cheng Wang, Jun Cheng, Bili Chen, Fukai Jia, Zhonggui Chen, and Jonathan Li. 2015. Line segment extraction for large scale unorganized point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing* 102 (2015), 172 – 183. https://doi.org/10.1016/j.isprsjprs.2014.12.027

Tony Lindeberg. 1993. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA.

Eric-Tuan Lê, Iasonas Kokkinos, and Niloy J. Mitra. 2020. Going Deeper with Lean Point Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Daniel Maturana and Sebastian Scherer. 2015. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 922 – 928.

Nicolas Mellado, Matteo Dellepiane, and Roberto Scopigno. 2015. Relative scale estimation and 3D registration of multi-modal geometry using Growing Least Squares. *IEEE transactions on visualization and computer graphics* 22, 9 (2015), 2160–2173.

Nicolas Mellado, Gaël Guennebaud, Pascal Barla, Patrick Reuter, and Christophe Schlick. 2012. Growing Least Squares for the Analysis of Manifolds in Scale-Space. *Comp. Graph. Forum* 31, 5 (Aug. 2012), 1691–1701. https://doi.org/10.1111/j.1467-8659.2012.03174.x

Nicolas Mellado, Thibault Lejemble, Gaël Guennebaud, and Pascal Barla. 2020. Ponca: a Point Cloud Analysis Library. https://github.com/poncateam/ponca/.

Aikaterini Mitropoulou and Andreas Georgopoulos. 2019. An Automated Process to Detect Edges in Unorganized Point Clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2019), 99–105.

Aron Monszpart, Nicolas Mellado, Gabriel J. Brostow, and Niloy J. Mitra. 2015. RAPter: Rebuilding Man-made Scenes with Regular Arrangements of Planes. *ACM Trans. Graph.* 34, 4, Article 103 (2015), 12 pages.

Charly Mourglia, Valentin Roussellet, Loic Barthe, Nicolas Mellado, Mathias Paulin, David Vanderhaeghe, and and others. 2021. Radium Engine. https://doi.org/10.5281/zenodo.5101334

Q. Mérigot, M. Ovsjanikov, and L. J. Guibas. 2011. Voronoi-Based Curvature and Feature Estimation from Point Clouds. *IEEE Transactions on Visualization and Computer Graphics* 17, 6 (2011), 743–756.

Keith Wei Liang Nguyen, A. Aprilia, Ahmad Khairyanto, Wee Ching Pang, Gerald Gim Lee Seet, and Shu Beng Tor. 2018. Edge detection from point cloud of worn parts. *Proceedings of the 3rd International Conference on Progress in Additive Manufacturing* (2018), 595–600. https://doi.org/10.25341/D45C7S

Huan Ni, Xiangguo Lin, Xiaogang Ning, and Jixian Zhang. 2016. Edge detection and feature line tracing in 3d-point clouds by analyzing geometric properties of neighborhoods. *Remote Sensing* 8, 9 (2016), 710.

Mark Pauly, Richard Keiser, and Markus Gross. 2003. Multi-scale Feature Extraction on Point-Sampled Surfaces. *Computer Graphics Forum* 22, 3 (2003), 281–289. https://doi.org/10.1111/1467-8659.00675

Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. 2018. Frustum PointNets for 3D Object Detection from RGB-D Data. *Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

C. R. Qi, H. Su, M. Kaichun, and L. J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 77–85. https://doi.org/10.1109/CVPR.2017.16

Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. 2016. Volumetric and Multi-View CNNs for Object Classification on 3D Data. *Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., USA, 5105–5114. http://dl.acm.org/citation.cfm?id=3295222.3295263

Andrés Serna, Beatriz Marcotegui, François Goulette, and Jean-Emmanuel Deschaud. 2014. Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In *4th International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014*. Angers, France. https://hal.archives-ouvertes.fr/hal-00963812

H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M. Yang, and J. Kautz. 2018. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2530–2539. https://doi.org/10.1109/CVPR.2018.00268

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*.

H. Thomas, C. R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. 2019. KPConv: Flexible and Deformable Convolution for Point Clouds. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 6410–6419.

E. Moscoso Thompson, G. Arvanitis, K. Moustakas, N. Hoang-Xuan, E. R. Nguyen, M. Tran, T. Lejemble, L. Barthe, N. Mellado, C. Romanengo, S. Biasotti, and B. Falcidieno. 2019. Feature Curve Extraction on Triangle Meshes. In *Eurographics Workshop on 3D Object Retrieval*, Silvia Biasotti, Guillaume Lavoué, and Remco Veltkamp (Eds.). The Eurographics Association. https://doi.org/10.2312/3dor.20191066

Chu Wang, Babak Samari, and Kaleem Siddiqi. 2018. Local Spectral Graph Convolution for Point Set Feature Learning. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV (Lecture Notes in Computer Science)*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.), Vol. 11208. Springer, 56–71. https://doi.org/10.1007/978-3-030-01225-0_4

Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. 2020. PIE-NET: Parametric Inference of Point Cloud Edges. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 20167–20178. https://proceedings.neurips.cc/paper/2020/file/e94550c93cd70fe748e6982b3439ad3b-Paper.pdf

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)* (2019).

Christopher Weber, Stefanie Hahmann, and Hans Hagen. 2010. Sharp Feature Detection in Point Clouds. In *Proceedings of the 2010 Shape Modeling International Conference (SMI '10)*. IEEE Computer Society, USA, 175–186. https://doi.org/10.1109/SMI.2010.32

Christopher Weber, Stefanie Hahmann, Hans Hagen, and Georges-Pierre Bonneau. 2012. Sharp feature preserving MLS surface reconstruction based on local feature line approximations. *Graphical Models* 74, 6 (2012), 335–345.

Martin Weinmann, Boris Jutzi, and Clément Mallet. 2013. Feature relevance assessment for the semantic interpretation of 3D point cloud data. In *ISPRS Annals*, Vol. 5.

Andrew P. Witkin. 1987. Scale-space filtering. In *Readings in Computer Vision*. Elsevier, 329–332.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

S. Xia and R. Wang. 2017. A Fast Edge Extraction Method for Mobile Lidar Point Clouds. *IEEE Geoscience and Remote Sensing Letters* 14, 8 (2017), 1288–1292.

Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. 2018. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 87–102.

L. Yu, X. Li, C. Fu, D. Cohen-Or, and P. Heng. 2018. PU-Net: Point Cloud Upsampling Network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2790–2799. https://doi.org/10.1109/CVPR.2018.00295

Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. EC-Net: An Edge-Aware Point Set Consolidation Network. In *Computer Vision – ECCV 2018*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.). Springer International Publishing, Cham, 398–414.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. 2017. Deep Sets. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3391–3401. http://papers.nips.cc/paper/6931-deep-sets.pdf

Aite Zhao, Jianbo Li, and Manzoor Ahmed. 2020. SpiderNet: A spiderweb graph neural network for multi-view gait recognition. *Knowledge-Based Systems* 206 (2020), 106273. https://doi.org/10.1016/j.knosys.2020.106273

Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 2015. 3D ShapeNets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1912–1920. https://doi.org/10.1109/CVPR.2015.7298801

Yin Zhou and Oncel Tuzel. 2018. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 4490–4499. https://doi.org/10.1109/CVPR.2018.00472

APPENDICES

A GLS DESCRIPTORS AND DERIVATIVES

The APSS [Guennebaud and Gross 2007] defines a scalar field $S(\mathbf{p})_{\mathbf{u}} = [1 \quad \mathbf{p}^T \quad \mathbf{p}^T\mathbf{p}] \cdot \mathbf{u}$, where $\mathbf{u} = [u_c \quad \mathbf{u}_\ell \quad u_q]$ is the vector of field parameters. The parameters of the algebraic sphere are obtained by normalizing these field parameters: $\hat{\mathbf{u}} = \mathbf{u}/\sqrt{||\mathbf{u}_\ell||^2 - 4u_c u_q}$. In the GLS [Mellado et al. 2012], the algebraic sphere parameters are reparameterized to compute its geometric parameters: $\tau = S(\mathbf{p})_{\hat{\mathbf{u}}}(\mathbf{p})$ the local relief, $\eta = \frac{\nabla S(\mathbf{p})_{\hat{\mathbf{u}}}(\mathbf{p})}{||\nabla S(\mathbf{p})_{\hat{\mathbf{u}}}(\mathbf{p})||}$ the normal vector and $\kappa = 2\hat{u}_q$ the mean curvature.

The Scale-Space Jacobian of the GLS parameters if defined as a 5×4 matrix:

$$\begin{bmatrix} \frac{\delta\tau}{\delta\mathbf{x}} & \frac{\delta\eta_x}{\delta\mathbf{x}} & \frac{\delta\eta_y}{\delta\mathbf{x}} & \frac{\delta\eta_z}{\delta\mathbf{x}} & \frac{\delta\kappa}{\delta\mathbf{x}} \\ \frac{\delta\tau}{\delta t} & \frac{\delta\eta_x}{\delta t} & \frac{\delta\eta_y}{\delta t} & \frac{\delta\eta_z}{\delta t} & \frac{\delta\kappa}{\delta t} \end{bmatrix},$$

where $\delta\mathbf{x}$ and δt are the derivatives in scale and space respectively. $k1$ is computed by projecting $\frac{\delta\eta}{\delta\mathbf{x}}$ on the surface tangent plane, which provides an estimate of the second fundamental form.

B SCORES USED FOR QUANTITATIVE COMPARISON

The scores used in Section 5 are defined as follows:

Precision (also denoted positive predictive value – PPV) measures the proportion of positive identifications that are actually correct (the higher, the better). It is defined as:

$$\text{precision} = \frac{TP}{TP + FP}.$$

Recall (also denoted sensitivity, hit rate, or true positive rate – TPR) measures the proportion of actual positives that are correctly identified (the higher, the better). It is defined as:

$$\text{recall} = \frac{TP}{TP + FN}.$$

The Matthews Correlation Coefficient (MCC) is a correlation coefficient between the observed and predicted binary classifications; it returns a value in [-1 : 1]. A coefficient of 1 represents a perfect prediction, 0 no better than random prediction and -1 indicates a total disagreement between prediction and observation. It is defined as:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

F1 score is a measure of a test accuracy. For binary classification, F1 is defined as follows:

$$\text{F1} = 2.\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2TP}{2TP + FP + FN}.$$

Accuracy measures is the fraction of predictions our model got right. For binary classification, accuracy is defined as:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

The Intersection over Union score (IoU) is a value between 0 and 1 specifying the overlap between the prediction and the observation. A value of 1 means that the union of the predicted and the reference

sets is the same as their overlap (intersection) while a value of 0 means that there is no overlap between the predicted and the reference sets. It is defined as follows:

$$\text{IoU} = \frac{TP}{TP + FP + FN}.$$