BoundED

Neural boundary and edge detection in 3D point clouds via local neighborhood statistics

Bode, Lukas; Weinmann, Michael; Klein, Reinhard

DOI
10.1016/j.isprsjprs.2023.09.023

Publication date
2023

Document Version
Final published version

Published in
ISPRS Journal of Photogrammetry and Remote Sensing

ISPRS Journal of Photogrammetry and Remote Sensing 205 (2023) 334–351

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

BoundED: Neural boundary and edge detection in 3D point clouds via local neighborhood statistics

Lukas Bode [a], Michael Weinmann [b,*], Reinhard Klein [a]

[a] University of Bonn, Friedrich-Hirzebruch-Allee 8, Bonn, 53115, Germany
[b] Delft University of Technology, Van Mourik Broekmanweg 6, Delft, 2628 XE, Netherlands

ARTICLE INFO

Keywords:
Point cloud processing
Machine learning
Neural network
Classification
Edge detection
Boundary detection

ABSTRACT

Extracting high-level structural information from 3D point clouds is challenging but essential for tasks like urban planning or autonomous driving requiring an advanced understanding of the scene at hand. Existing approaches are still not able to produce high-quality results consistently while being fast enough to be deployed in scenarios requiring interactivity. We propose to utilize a novel set of features describing the local neighborhood on a per-point basis via first and second order statistics as input for a simple and compact classification network to distinguish between non-edge, sharp-edge, and boundary points in the given data. Leveraging this feature embedding enables our algorithm to outperform the state-of-the-art technique PCEDNet in terms of quality and processing time while additionally allowing for the detection of boundaries in the processed point clouds.

1. Introduction

3D point cloud data obtained from terrestrial or airborne laser scanning as well as depth sensors and image-based structure-from-motion have become the prerequisite for numerous applications including geographic information systems, urban planning, indoor modeling for the built environment, autonomous driving, and navigation systems. However, the sampling of scenes with arbitrary complexity in terms of unstructured data complicates the further processing of the data as e.g. required when extracting characteristic features for navigation or scene interpretation according to object instances and materials. Edges represent characteristic features that often occur at object borders as well as on surfaces (in the form of ridges or engravings) and linear scene structures like scaffolds and, hence, provide essential information regarding the underlying geometric structures. However, automatic edge detection in 3D point cloud data remains a challenging task. Whereas physical edges may not appear as sharp due to damage or cleaning (e.g. stone or plastered buildings, progressively smoothed edges, polished mechanical parts, etc.), there are also limitations inherent to the scanning approaches, especially due to the typically uneven, noisy sampling of the scene, that may result in a slight rounding effect of edges in the reconstruction. Furthermore, the sharpness, smoothness, or roundness of edges also depends on the observation scale. Therefore, there might be some ambiguity in defining edges, which may require involving further context information. In addition, with point clouds typically consisting of tens or hundreds of millions of points, efficient operators are required.

Advances in machine learning and the rapidly growing availability of 3D data have led to several supervised learning approaches for concept classification. Respective approaches include the classification of structures according to semantic categories such as facades, roofs, different forms of vegetation or pole/trunk structures using pointwise hand-crafted geometric descriptors on a single *optimal* scale (Demantké et al., 2011; Weinmann et al., 2015a,c; Hackel et al., 2016b) or multiple scales (Brodu and Lague, 2012; Blomley and Weinmann, 2017), additionally leveraging contextual information (Niemeyer et al., 2014; Weinmann et al., 2015b; Steinsiek et al., 2017; Landrieu et al., 2017), as well as approaches based on neural networks (Guo et al., 2020; Xie et al., 2020) including the initial projection of the point cloud onto synthetic 2D-images followed by a 2D-CNN-based segmentation and the final re-projection to the point cloud to obtain the segmentation result (Boulch et al., 2017; Lawin et al., 2017), 3D convolutional neural networks (Huang and You, 2016; Hackel et al., 2017) and their combination with trilinear interpolation and context integration based on conditional random fields (Tchapmi et al., 2017), flexible and deformable convolution (Thomas et al., 2019) and dilated graph convolution (Mao et al., 2022) for point clouds, PointNet variants (Qi et al., 2017b) and their combination with point cloud partitioning into superpoints (Landrieu and Simonovsky, 2018), and addressing the class-imbalance in the training data based on special loss terms like a self-amelioration loss (Li et al., 2022). Furthermore, a few works also focused on the individual classification of points according to being or not being on edges based on multi-scale features and a

* Corresponding author.
E-mail addresses: lbode@cs.uni-bonn.de (L. Bode), M.Weinmann@tudelft.nl (M. Weinmann), rk@cs.uni-bonn.de (R. Klein).

https://doi.org/10.1016/j.isprsjprs.2023.09.023
Received 25 October 2022; Received in revised form 27 September 2023; Accepted 28 September 2023
Available online 24 October 2023

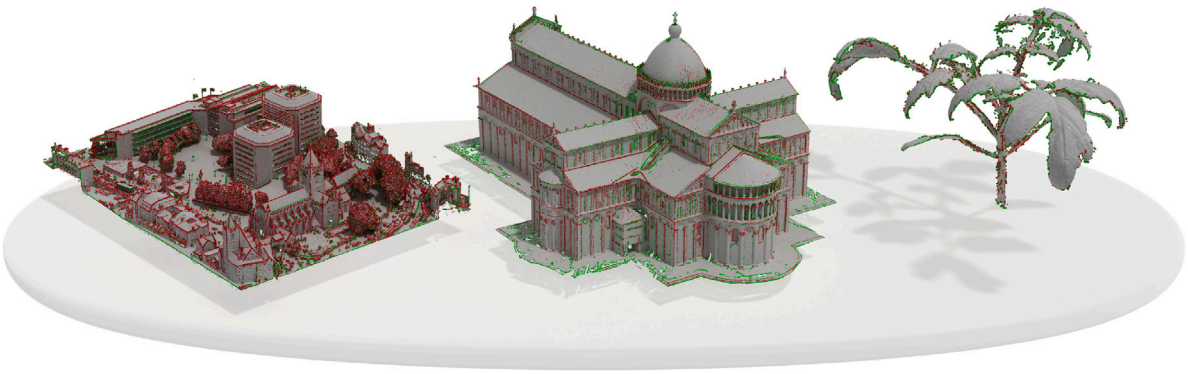

Fig. 1. Our BoundED approach extracts sharp edges and boundaries from 3D point cloud data purely based on positional data. Points classified as sharp-edge are highlighted in red while boundary points are highlighted in green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

random-forest-based classification (Hackel et al., 2016a), multi-scale features and a dedicated neural network based edge detection classifier (Himeur et al., 2021), neural-network-based pointwise distance estimation to the next sharp geometric feature (Matveev et al., 2022), binary-pattern-based filtering on local topology graphs (Guo et al., 2022), neural-network-based edge-aware point set consolidation leveraging an edge-aware loss (Yu et al., 2018), training two networks based on PointNet++ (Qi et al., 2017b) to classify points into corners and edges and subsequently applying non-maximal suppression and inferring feature curves (Wang et al., 2020), the learning of multi-scale local shape properties (e.g., normal and curvature) (Guerrero et al., 2018), and the computation of a scalar *sharpness* field defined on the underlying Moving Least-Squares surface of the point cloud whose local maxima correspond to sharp edges (Raina et al., 2018, 2019). However, extracting high-quality edge and boundary data from a large variety of different 3D point clouds fast enough to eventually be suitable for usage in embedded systems or real-time settings remains an open problem.

In this paper, inspired by the *maximum mean discrepancy* (MMD) operator (Gretton et al., 2012) which allows to compare distributions by embedding them in a feature space and comparing the mean of the respective embeddings, we propose to tackle the point classification task by training a network to distinguish between classes based on a feature embedding related to the first and second order statistics of the respective point's neighborhood. This embedding contains enough information for the classification network to learn the difference between non-edge, sharp-edge, and boundary points while at the same time being well structured and compact, making our solution very fast in terms of processing time. Various results of our *Boundary and Edge Detection* (BoundED) approach are depicted in Fig. 1. Our main contributions can be summarized as follows:

- We present a novel set of features for edge and boundary characterization and detection capturing local neighborhood information of point clouds better and being cheaper to compute than state-of-the-art approaches (Himeur et al., 2021).
- We introduce a novel filtering technique to ensure that the extracted features are robust regarding outliers and surface patches of complex geometry.
- We demonstrate the benefits of this novel feature embedding at the example of a modified state-of-the-art neural edge detection network architecture giving better results with an even smaller network.
- Our evaluation demonstrates the ability of the proposed features to capture information regarding boundary classification of points in addition to edge classification.

2. Related work

The detection of 3D edges in terms of sharp features, feature contours, or curves within unstructured point cloud data is a challenging task. In the following, we will review respective developments grouped according to conventional approaches and learning-based methods.

2.1. Conventional methods

Conventional methods include surface mesh reconstruction or graph-based approaches and analyzing local neighborhoods of each individual point based on principal component analysis (PCA). Thereby, the given connectivity information of a point with respect to its neighbors allows for a faster nearest neighbor search in comparison to unstructured point sets. However, preserving sharp edges and complex features in a reconstructed model is challenging due to smoothing effects induced by several reconstruction techniques. Directly extracting edges from unstructured point clouds has been addressed based on computing geometric descriptors per point based on the local covariance characteristics (Gumhold et al., 2001; Gelfand and Guibas, 2004). Respective variants include taking the ratio between the Eigenvalues of the local covariance matrices on a single scale (Mérigot et al., 2011; Xia and Wang, 2017) or different scales (Pauly et al., 2003; Bazazian et al., 2015), local slippage analysis to define edges between segments of rotationally and translationally symmetrical shapes such as planes, spheres, and cylinders (Gelfand and Guibas, 2004), or directly estimating curvature (Lin et al., 2015; Nguyen et al., 2018). Considering multiple scales reduces the susceptibility to noise, but such methods still rely on the suitable specification of a decision threshold. Non-parametric edge extraction has been achieved via kernel regression (Öztireli et al., 2009) or Eigenvalue analysis (Bazazian et al., 2015). Others focused on detecting depth-discontinuities based on finding triangles with oblique orientations or finding triangles with long edges (Tang et al., 2007) or focusing on high-curvature points given as the extremum of curvatures (Fan et al., 1987) or curvature-guided region growing (Rusu et al., 2008). In addition, edge detection has been approached based on normal variation analysis (Che and Olsen, 2018), 3D Canny edge detection (Monga et al., 1991), the combination of normal estimation and graph theory (Yagüe-Fabra et al., 2013), alpha-shapes (Edelsbrunner and Mücke, 1994), or boundary detection via DBSCAN-based detection and segmentation of 3D planes (Chen et al., 2022).

Further approaches followed a moving least-squares (MLS) surface reconstruction with the subsequent detection of 3D edges based on a Gaussian map clustering computed within a local neighborhood (Demarsin et al., 2007; Weber et al., 2010, 2012; Ni et al., 2016). The consideration of higher-order local approximations of non-oriented input gradients in MLS-based reconstruction has been used for the computation of continuous non-oriented gradient fields (Chen et al., 2013), which allows a better preservation of surface or image structures. Another possibility to achieve continuously differentiable surfaces consists in exploring the scale-space for MLS (Mellado et al., 2012). Furthermore, a scalar *sharpness* field defined on the underlying Moving

Least-Squares surface of the point cloud has been proposed, where local maxima correspond to sharp edges (Raina et al., 2018, 2019). Other approaches include the combination of adaptive reconstruction kernels (Fleishman et al., 2005) and spline fitting (Daniels et al., 2008), the detection of boundary points and internal points as well as the subsequent application of a Fast-Fourier-Transform-based edge reconstruction to avoid the need to define a specific order for polynomial curve fitting (Mineo et al., 2019), the use of subspace detection and feature intersection (Fernandes and Oliveira, 2012), mean-shift-based selection of the most distant points with respect to the centroid of their neighborhood (Ahmed et al., 2018), the use of locally defined curve set features (Li and Hashimoto, 2017), the intersection of automatically detected planes (Mitropoulou and Georgopoulos, 2019), the filtering of potential feature points according to their local topology graph based on binary patterns (Guo et al., 2022), or RANSAC-based spatial regularization of sharp feature detector responses (Lin et al., 2015). In addition, gradient-based edge detection with a subsequent non-maxima suppression and edge linking into linear and smooth structures (Xia and Wang, 2017) has been investigated.

2.2. Learning-based methods

Along the rapid progress in machine learning, learning-based approaches have been proposed for classifying individual points as *edge* or *non-edge*. Besides approaches based on least square regression or support vector machines (Wang et al., 2019) that, however, had not been investigated in a general scenario, this can be achieved by the use of multi-scale features with a random forest-based edge classification (Hackel et al., 2016a) or neural network-based edge classifier relying on either MLS-based (Himeur et al., 2021) or neural (Yu et al., 2022) pointwise features. Other approaches include the neural network-based pointwise distance estimation to the next sharp geometric feature (Matveev et al., 2022), or neural network-based edge-aware point set consolidation (Yu et al., 2018) and 3D semantic edge detection based on a two-stream fully-convolutional network to jointly perform edge detection and semantic segmentation (Hu et al., 2020). A further method (Wang et al., 2020) trains two neural networks to classify points into corners and edges based on a PointNet++-like architecture (Qi et al., 2017b). After a subsequent non-maxima suppression of the classified points and their PointNet++-based clustering, a two-headed PointNet (Qi et al., 2017a) generates the final set of curves. This concatenation of deep networks induces a high computational burden and relies on high resource requirements. In addition, the learning of multi-scale local shape properties (e.g., normal and curvature) (Guerrero et al., 2018) and the use of CNNs for adaptive feature extraction from observations in a camera and laser-scanner setup (Xiao et al., 2019) have been investigated. Furthermore, the prediction of part boundaries within a 3D point cloud based on a graph convolutional network has been proposed (Loizou et al., 2020). This approach, however, relies on all processed point clouds consisting of a fixed number of points and, thus, is unsuitable for the general scenario of processing point clouds of arbitrary size that we address with our approach. Further purely on boundary detection focused methods include the initial extraction of the exterior boundary based on neighborhood characteristics and the subsequent analysis regarding whether a point belongs to a hole boundary (Trinh et al., 2015), and approaches based on a deep neural network (Tabib et al., 2020).

There are also a few image-based approaches that initially convert the 3D point cloud data into images (Lin et al., 2015). Subsequently, a line segment detector (Von Gioi et al., 2008) is used to extract lines in 2D, which are backprojected to the point cloud. Another approach (Lu et al., 2019) relies on an initial segmentation of the point cloud into planar regions based on region growing and merging, which is followed by a plane-wise point projection into a 2D image and a final 2D contour extraction and backprojection to get the respective line segment in 3D space.

With our approach, we follow the avenue of neural network-based edge and boundary detection within 3D point clouds. We take inspiration from the *maximum mean discrepancy* (MMD) operator (Gretton et al., 2012) for the definition of a local feature embedding with respect to first- and second-order statistics of a local point's neighborhood, and we show that this embedding allows robust detection of non-edge, sharp-edge, and boundary points already with a compact network, thereby enabling fast inference times.

3. Methodology

With our approach, which we denote as BoundED, we aim at the robust and fast detection of non-edge, sharp-edge, or boundary points within given point clouds. For this purpose, we leverage the combination of a novel filtering technique to detect outliers and close unconnected surfaces in point neighborhoods, a local encoding of feature characteristics based on the maximum mean discrepancy (MMD) operator with respect to the local first- and second-order statistics, and their efficient classification based on a compact multi-layer perceptron (MLP) (see Fig. 2). In the following sections, we provide detailed descriptions regarding these aspects as well as respective implementation details.

3.1. Feature computation

To compute meaningful features as input for the consecutive neural classification step, we generalize the idea of dividing a set of 3D points into two disjoint subsets and analyzing their respective covariances introduced by Bode et al. (2022) in the context of image denoising.

Let $\mathcal{P} = \{\boldsymbol{p}_i\}$ for $i = 1, \dots, n$ be the given 3D point cloud consisting of n points. Using the k-nearest neighbors (k-NN) operator $\mathrm{NN}_k(\boldsymbol{p}, \mathcal{P})$, we extract local neighborhoods $\mathcal{N}_{i,k} = \mathrm{NN}_k(\boldsymbol{p}_i, \mathcal{P})$ with k points each. Throughout the remainder of this section, the neighborhood size k is omitted for notational simplicity.

For sufficiently dense point clouds in the absence of noise, this neighborhood represents a roughly disc-shaped set of points. In order to be invariant to the scale and sampling of the given point cloud, the point sets are normalized individually before features can be extracted. We propose to utilize the covariance matrix $\boldsymbol{K}_i = \mathrm{cov}(\mathcal{N}_i)$ for this purpose. By conducting an SVD of the covariance matrix

$$\boldsymbol{K}_i = \boldsymbol{U}_i \boldsymbol{\Sigma}_i \boldsymbol{V}_i^T \tag{1}$$

singular values $\sigma_{i,j} = \boldsymbol{\Sigma}_{i,jj}$ can be read from the diagonal entries of the matrix $\boldsymbol{\Sigma}_i$. Without loss of generality, these singular values are assumed to be sorted in descending order, i.e. $\sigma_{i,1} \geq \sigma_{i,2} \geq \sigma_{i,3}$. Intuitively, these singular values are directional variances with directions being given by the corresponding Eigenvectors. Since $\mathcal{N}_i$ is roughly disk-shaped, $\sigma_{i,1}$ and $\sigma_{i,2}$ can be seen as variance in direction of the disk's perpendicular semiaxes. Note, that in general $\sigma_{i,1}$ and $\sigma_{i,2}$ are similar but not equal as the points $\mathcal{N}_i$ will never represent a perfect uniformly sampled disk in practice. For the purpose of normalization, the neighborhood is centered around the origin according to the neighborhood's center of mass

$$\bar{\mathcal{N}}_i = \frac{1}{|\mathcal{N}_i|} \sum_{\boldsymbol{p} \in \mathcal{N}_i} \boldsymbol{p} \tag{2}$$

and scaled by the average standard deviation along the semiaxes:

$$\hat{\mathcal{N}}_i = \left\{ \frac{2}{\sqrt{\sigma_{i,1}} + \sqrt{\sigma_{i,2}}} (\boldsymbol{p} - \bar{\mathcal{N}}_i) \mid \boldsymbol{p} \in \mathcal{N}_i \right\}. \tag{3}$$

Besides normalization of the neighborhood, this SVD and in particular the Eigenvector $\boldsymbol{n}_i$ corresponding to $\sigma_{i,3}$ is utilized for further processing as this vector together with the neighborhood's center of mass $\bar{\mathcal{N}}_i$ defines a least-squares fitted plane to $\mathcal{N}_i$. Note that, in contrast to other approaches like e.g. PCEDNet (Himeur et al., 2021), by using this Eigenvector $\boldsymbol{n}_i$ as normal, our BoundED does not rely on any

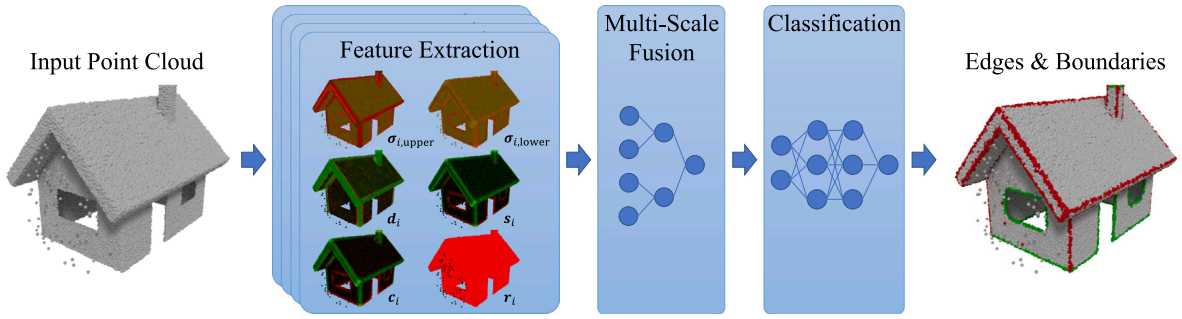

Fig. 2. Overview of our BoundED approach: Based on an input point cloud, several features describing the local geometry are extracted on multiple scales. After pairwise fusion of the features for different scales, we classify the input points by an MLP leveraging the fused features as either non-edge, sharp-edge, or boundary points.

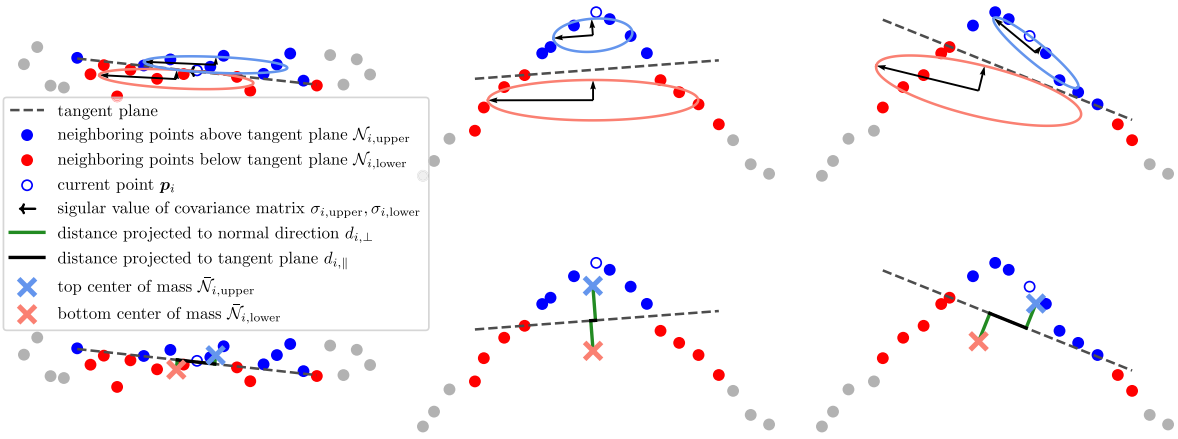

Fig. 3. Features extracted from the local neighborhood of a point. Points can be classified as non-edge or sharp-edge by analyzing their neighborhood with respect to singular values and means of points above and below a least-squares fitted tangent plane. Planar neighborhoods (left) tend to have similar values for $\sigma_{i,\text{upper}}$ and $\sigma_{i,\text{lower}}$ while having low values for $d_{i,\perp}$. Sharp-edge neighborhoods (middle) exhibit a larger difference in $\sigma_{i,\text{upper}}$ and $\sigma_{i,\text{lower}}$ as well as large $d_{i,\perp}$. In contrast, neighborhoods of points close to sharp edges (right) have higher $d_{i,\parallel}$ than neighborhoods of points directly on the edge.

precomputed normals but only on the 3D positions of the points. We have observed, that the orientation of $\boldsymbol{n}_i$ can be unstable near outliers. Thus, we only consider the $\lfloor k/2 \rfloor$ points closest to $\hat{\mathcal{N}}_i$ for this step. According to this plane, the neighborhood is partitioned into two disjoint subsets

$$\mathcal{N}_{i,\text{upper}} = \left\{ \boldsymbol{p} \in \hat{\mathcal{N}}_i \mid \langle \boldsymbol{p}, \boldsymbol{n}_i \rangle \geq 0 \right\} \tag{4}$$

$$\mathcal{N}_{i,\text{lower}} = \left\{ \boldsymbol{p} \in \hat{\mathcal{N}}_i \mid \langle \boldsymbol{p}, \boldsymbol{n}_i \rangle < 0 \right\}. \tag{5}$$

As depicted in Fig. 3, an analysis of these provides valuable information regarding local geometry. We propose to analyze the subset's statistics to capture this information. In particular, singular values $\sigma_{i,\text{upper},j}$ and $\sigma_{i,\text{lower},j}$ for $j \in \{1,2,3\}$ are computed by means of individual SVDs of the covariance matrices of $\mathcal{N}_{i,\text{upper}}$ and $\mathcal{N}_{i,\text{lower}}$ respectively. Additionally, the distance between the centers of mass of both subsets

$$\bar{\mathcal{N}}_{i,\text{upper}} = \frac{1}{|\mathcal{N}_{i,\text{upper}}|} \sum_{\boldsymbol{p} \in \mathcal{N}_{i,\text{upper}}} \boldsymbol{p} \tag{6}$$

$$\bar{\mathcal{N}}_{i,\text{lower}} = \frac{1}{|\mathcal{N}_{i,\text{lower}}|} \sum_{\boldsymbol{p} \in \mathcal{N}_{i,\text{lower}}} \boldsymbol{p} \tag{7}$$

decomposed into perpendicular and tangential components is calculated as

$$d_{i,\perp} = \langle \bar{\mathcal{N}}_{i,\text{upper}} - \bar{\mathcal{N}}_{i,\text{lower}}, \boldsymbol{n}_i \rangle \tag{8}$$

$$d_{i,\parallel} = \| (\bar{\mathcal{N}}_{i,\text{upper}} - \bar{\mathcal{N}}_{i,\text{lower}}) - d_\perp \boldsymbol{n}_i \|_2. \tag{9}$$

Intuitively, low values for $d_{i,\perp}$ indicate that the local neighborhood $\mathcal{N}_i$ is near planar and thus the probability for $\boldsymbol{p}_i$ being part of a sharp edge is small. In contrast, high values are found in areas with a high amount of geometric detail or noise. A large tangential distance $d_{i,\parallel}$ can indicate, that an edge is close-by, but $\boldsymbol{p}_i$ may not necessarily be coincident (see Fig. 3).

Furthermore, inspired by Bendels et al. (2006), to improve detection of outliers and boundaries, the perpendicular and tangential components of the distance between $\boldsymbol{p}_i$ and the center of mass of its k nearest neighbors are computed as:

$$s_{i,\perp} = \langle \boldsymbol{p}_i - \bar{\mathcal{N}}_i, \boldsymbol{n}_i \rangle \tag{10}$$

$$s_{i,\parallel} = \| (\boldsymbol{p}_i - \bar{\mathcal{N}}_i) - s_\perp \boldsymbol{n}_i \|_2. \tag{11}$$

While not necessarily always following this observation, points at boundaries tend to have large $s_{i,\parallel}$ and at the same time small $s_{i,\perp}$. Intuitively, the neighbors of points at boundaries are all on one side which indicates that they are far away from the center of mass of their neighborhood. If $\boldsymbol{p}_i$ is an outlier near a well-defined surface, the corresponding $s_{i,\perp}$ tends to be large.

In summary, the analysis yields the following features: the singular values $\boldsymbol{\sigma}_{i,\cdot} = (\sigma_{i,\cdot,1}, \sigma_{i,\cdot,2}, \sigma_{i,\cdot,3})^T$ of the upper and lower subsets respectively, the perpendicular and tangential distances between the centers of mass of both subsets $\boldsymbol{d}_i = (d_{i,\perp}, d_{i,\parallel})^T$, and the perpendicular and tangential distances between the point $\boldsymbol{p}_i$ and the center of mass of its neighborhood $\boldsymbol{s}_i = (s_{i,\perp}, s_{i,\parallel})^T$. Thus, we assemble a per-point 10D feature vector according to

$$\hat{\boldsymbol{x}}_i = (\boldsymbol{\sigma}_{i,\text{upper}}, \boldsymbol{\sigma}_{i,\text{lower}}, \boldsymbol{d}_i, \boldsymbol{s}_i)^T. \tag{12}$$

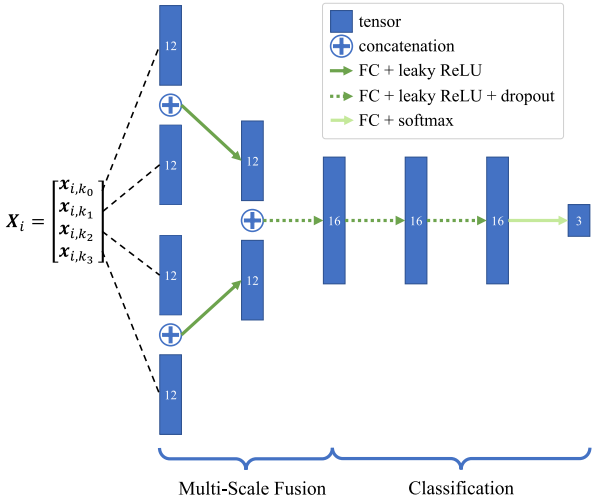

Fig. 4. Architecture of the multi-scale fusion and classification network consisting of fully connected (FC) layers, leaky rectified linear unit (leaky ReLU) activations, dropout, and softmax function. Features computed on multiple scales are combined in a pairwise manner and afterwards processed by an MLP to classify a point as non-edge, sharp-edge, or boundary.

3.2. Neighborhood filtering

During the previously described feature extraction process, the tangent plane fit at a point $\boldsymbol{p}_i$ can be heavily influenced by outlier points or close-by parallel surfaces contained in the neighborhood $\mathcal{N}_{i,k}$ of k nearest neighbors, which in consequence also influences the partitioning and therefore increases problem complexity for the classification network. To avoid this, we introduce an additional neighborhood filtering step before extracting the features. For this, we first estimate the local point density by calculating the median of the distances between each point and its closest neighbor in the neighborhood $\rho_{i,k}$. Afterwards, a graph is built from the neighborhood connecting each point with every other point. In this graph, we search for points that cannot be reached from the point to be classified by a path, in which each edge is shorter than $\lambda_d\, \rho_{i,k}$ with λ_d being a hyperparameter. The respective points are removed from the neighborhood as they do not contain meaningful information regarding the surface patch to be classified.

This technique can be efficiently implemented as follows: Let $\boldsymbol{C} \in \{0,1\}^{k\times k}$ be an adjacency matrix with entries

$$\boldsymbol{C}_{xy} = \begin{cases} 1 & \text{if } |\boldsymbol{p}_x - \boldsymbol{p}_y| < \lambda_d\, \rho_{i,k}, \\ 0 & \text{else,} \end{cases} \tag{13}$$

describing a graph with edges that connect two points $\boldsymbol{p}_x, \boldsymbol{p}_y \in \mathcal{N}_{i,k}$ if the distance between them is less then $\lambda_d\, \rho_{i,k}$. Furthermore, let $\boldsymbol{v}_{p_i} \in \{0,1\}^k$ with

$$\boldsymbol{v}_{p_i,x} = \begin{cases} 1 & p_x = p_i, \\ 0 & \text{else,} \end{cases} \tag{14}$$

be a vector marking the point to be classified. Then, we can find points of local neighborhood $\mathcal{N}_{i,k}$ reachable via paths of arbitrary length according to our adjacency matrix $\boldsymbol{C}$ through a series of matrix–vector-multiplications

$$\boldsymbol{v} = \boldsymbol{C}^k \cdot \boldsymbol{v}_{p_i}, \tag{15}$$

Intuitively, this can be seen as growing a minimum spanning tree starting from point $\boldsymbol{p}_i$. Usually, less than 20 matrix–vector-multiplications are needed to calculate the final connectivity in practice, since a minimum spanning tree of depth 20 is in most cases sufficient to reach all connected points.

Using this filtering technique on the neighborhood retrieved for an outlier point will usually remove almost all other points from the neighborhood not allowing for reasonable feature extraction afterwards. Hence, we calculate the ratio of retained points and total points $r_{i,k}$ in the neighborhood. As outliers are not part of an actual surface, all points with $r_{i,k} < \lambda_t$ are directly classified as non-edge.

Through a series of experiments, we have found $\lambda_d = 4$ and $\lambda_t = 0.1$ to work well for most cases. Thus, these settings are used for all our experiments presented in this paper. Note, that λ_d and λ_t are relative quantities related to the local neighborhood statistics, making them applicable for a large variety of point clouds. While we expect that better values for λ_d and λ_t could be derived from a global statistical analysis of the point cloud, we opted for the simple heuristic described above since our focus is on speed and we, therefore, constrain our analysis to the patch level. The motivation for $\lambda_d = 4$ is based on the assumption that the surface structure does not vary too much on a local level and sparse points that vary too much from the local median distance can be considered as outliers. Regarding λ_t, the number of retained points after the filtering should still exceed a certain percentage (in our case 10%) within a local neighborhood to allow the representation of a surface patch/structure. We rely on our statistical feature-based classification network to handle any potential failure cases.

3.3. Multi-scale feature embedding

In order to classify points $\boldsymbol{p}_i$ of a point cloud $\mathcal{P}$ as non-edge, sharp-edge, or boundary, the per-point data $\boldsymbol{X}_i$ is individually processed by a small MLP. $\boldsymbol{X}_i$ relies on computing $\hat{\boldsymbol{x}}_{i,k}$ on m different scales k_0, …, $k_{(m-1)}$, i.e. choosing neighborhoods containing varying numbers of points k, for each point $\boldsymbol{p}_i$. Inspired by the GLS (Mellado et al., 2012) features utilized by PCEDNet (Himeur et al., 2021), we add the tangential and perpendicular distances

$$c_{i,k,\perp} = \langle \bar{\mathcal{N}}_{i,k} - \overline{(\mathcal{N}_{i,k_0} - \mathcal{N}_{i,k})}, \boldsymbol{n}_{i,k_0} \rangle \tag{16}$$

$$c_{i,k,\parallel} = \| (\bar{\mathcal{N}}_{i,k} - \overline{(\mathcal{N}_{i,k_0} - \mathcal{N}_{i,k})}) - c_{i,k,\perp} \boldsymbol{n}_{i,k_0} \|_2. \tag{17}$$

between the center of mass $\bar{\mathcal{N}}_{i,k}$ of each scale k and the center of mass of points of the largest scale's neighborhood $\bar{\mathcal{N}}_{i,0}$ as well as the ratio of retained neighborhood points previously calculated during the filtering process to each $\hat{\boldsymbol{x}}_{i,k}$:

$$\boldsymbol{x}_{i,k} = (\hat{\boldsymbol{x}}_{i,k}, \boldsymbol{c}_{i,k}, r_{i,k})^T, \tag{18}$$

where $\boldsymbol{c}_{i,k} = (c_{i,k,\perp}, c_{i,k,\parallel})^T$. The complete multi-scale per-point features can be written in matrix form as

$$\boldsymbol{X}_i = \begin{bmatrix} \boldsymbol{x}_{i,k_0,1} & \cdots & \boldsymbol{x}_{i,k_0,13} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_{i,k_{(m-1)},1} & \cdots & \boldsymbol{x}_{i,k_{(m-1)},13} \end{bmatrix}. \tag{19}$$

These multi-scale features are fused in a pair-wise manner similarly to PCEDNet (Himeur et al., 2021) as depicted in Fig. 4, before being processed by the classification MLP itself.

3.4. Network architecture

For our experiments, we use features computed on four different scales using 128, 64, 32, and 16 neighboring points respectively. In contrast to PCEDNet, BoundED uses fewer scales, i.e. 4 instead of 16. However, to accommodate for the lost network depth due to using fewer scales, an additional hidden layer is added to the classification MLP, giving the network a total of 1.7k learnable parameters. For training the network, a focal loss (Lin et al., 2017) with $\gamma = 2$ is used as training batches are usually very unbalanced due to the small number of edge points compared to non-edge points in most point clouds. Additionally, as only very few boundary points are included in the *Default++* dataset due to the fact that it originally has not been designed for boundary detection, we duplicate those until the training

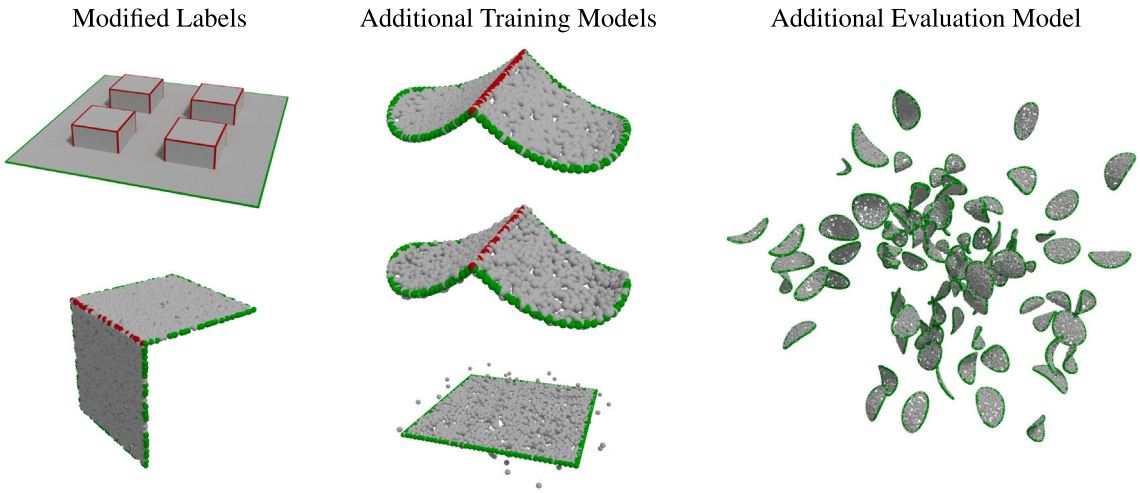

Fig. 5. Adjustments made to the *Default* dataset to facilitate boundary detection. The labels of two point clouds (left) are modified to identify already included boundary points correctly. Three simple point clouds (middle) are added to the training and validation set to improve the coverage of potential boundary point cases. For an additional evaluation, we add a further point cloud (right) to ensure a sufficient representation of boundary points in the evaluation data.

data consists of an equal number of sharp-edge and boundary points. Furthermore, we propose to add dropout (Srivastava et al., 2014) with $p = 0.5$ to the classification layers to prevent overfitting and facilitate a more stable training process.

3.5. Implementation details

Our algorithm is implemented using PyTorch (Paszke et al., 2019) for feature extraction as well as the neural network and its training. For finding the local neighborhood of points, the k-NN implementation of PyTorch3D (Ravi et al., 2020) is used. Due to the point sets $\mathcal{N}_{i,k,\text{upper}}$, $\mathcal{N}_{i,k,\text{lower}}$ containing different numbers of points for different $\boldsymbol{p}_i$, we employ masking to efficiently vectorize the task and fully utilize the tremendous computation capabilities of modern GPUs during the feature extraction phase. The network is trained using the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and learning rate 0.001. The batch size is set to 16 384. The number of training iterations varies between used datasets and is described in detail in Section 4.1.

4. Results and discussion

In the following, the effectiveness of the proposed combination of our novel multi-scale features and our compact classification network is evaluated quantitatively as well as qualitatively on several different datasets. We focus mostly on the comparison with the state-of-the-art point cloud edge detection network PCEDNet by Himeur et al. (2021) as it is the most relevant previous work due to also being designed to be fast and compact. Similar to our BoundED approach, they rely on feeding their classification network with multi-scale per-point features allowing for a direct comparison of the used embeddings. Furthermore, boundary detection capabilities of our network are assessed. Finally, an experiment on corrupted data as well as an ablation study regarding the chosen features and the employed number of scales further validate our results.

4.1. Datasets

We train and evaluate our approach on several different datasets and provide comparisons to other point cloud edge detection algorithms. To allow for a direct comparison with PCEDNet (Himeur et al., 2021), their *Default* dataset as well as the publicly available *ABC* (Koch et al., 2019) dataset are used.

Default. Introduced by Himeur et al. (2021), this dataset is designed to be as small as possible in order to facilitate very short training times with only a few simple hand-labeled point clouds to train on but still generalize well to arbitrary other point clouds. It contains 9 point clouds for training as well as 7 different point clouds for evaluation. To form the validation set, 1000 points are randomly sampled from each class. Despite containing three different classes of points, i.e. non-edge, sharp-edge, and smooth-edge, originally, this work focuses on non-edge and sharp-edge classification only and therefore treats smooth-edge points as non-edge points in all results. We train BoundED for 3000 iterations on this dataset.

ABC. The *ABC* dataset published by Koch et al. (2019) is a very large collection of CAD models accompanied with triangle meshes and feature annotations among other data. Point clouds are generated from triangle meshes by simply removing all edges and faces. A ground truth classification label for each point is extracted by checking whether it is part of any CAD curve flagged as *sharp*. To ensure a meaningful comparison with the work by Himeur et al. (2021), we also only use chunk 0000 and exactly the same 200 models for training and 50 models for validation while also using all 7168 point clouds for evaluation. As *ABC* contains many more points than *Default*, we train our network for 8000 iterations on its training data.

Default++. As the original *Default* dataset published by Himeur et al. (2021) does not include annotated boundary vertices, which prevents its use for training models for boundary detection tasks, we propose to extend it as shown in Fig. 5 to create the *Default*++ dataset. The original *Default* dataset contains two models containing boundary points not annotated as such. Thus, the first modification is to add these boundary annotations accordingly. Furthermore, it is extended by two additional point clouds for training, which were specifically designed to contain clean and noisy curved boundaries of varying radii as these cases are not included in the original *Default* dataset. Finally, to prevent boundary points from being heavily underrepresented in the evaluation set, we additionally add an evaluation model containing a multitude of boundary situations with varying levels of noise. Since the class of boundary points in the training set is still much smaller than the classes of non-edge or sharp-edge points, we only add 100 randomly sampled boundary points to the validation set. The resulting training set contains 279.5k non-edge, 15.7k sharp-edge, and only 0.9k boundary points. As it is similar in size to the *Default* dataset, we use the same 3000 iterations to train on *Default*++.

Table 1
Median scores of edge detection approaches evaluated on the *Default* dataset. The dataset used for parameter tuning or training is mentioned in parentheses.
Source: Data regarding PCEDNet-2c is taken from Himeur et al. (2021).

	Precision(↑)	Recall(↑)	MCC(↑)	F1(↑)	Accuracy(↑)	IoU(↑)
CA (Default) (Bazazian et al., 2015)	0.184	0.891	0.332	0.305	0.753	0.178
CA (ABC) (Bazazian et al., 2015)	0.183	0.357	0.188	0.242	0.863	0.138
FEE (Default) (Alliez et al., 2022; Mérigot et al., 2011)	0.241	0.866	0.400	0.376	0.828	0.232
FEE (ABC) (Alliez et al., 2022; Mérigot et al., 2011)	0.060	**0.961**	−0.021	0.113	0.082	0.060
PCEDNet-2c (*Default*) (Himeur et al., 2021)	0.364	0.611	0.402	0.430	0.908	0.274
BoundED-2c (Ours) (*Default*)	**0.597**	0.566	**0.555**	**0.581**	**0.950**	**0.410**
BoundED-2c (Ours) (*ABC*)	0.426	0.702	0.510	0.530	0.924	0.361

Additional evaluation data. To assess the capabilities of the proposed algorithm more thoroughly, we also use publicly available point clouds of 3D scanned buildings and plants. The *christ_church*[1] point cloud contains 1.9 million points of the Christ Church Cathedral and its surrounding in Dublin. Furthermore, the *pisa_cathedral*[2] point cloud with 2.5 million points scanned by Mellado et al. (2015) is used as well. The *station*[3] point cloud is an even larger point cloud representing a train station as 12.5 million points which we also use for evaluation. Finally, we are using point clouds of three different plants scanned by Conn et al. (2017): An Arabidopsis,[4] a Tobacco,[5] and a Tomato[6] plant with 172k, 1474k, and 226k points respectively.

4.2. Metrics

Similarly to the work by Himeur et al. (2021), we use several metrics for comparison: Precision, Recall, Matthews Correlation Coefficient (MCC), F1 score, Accuracy, and Intersection over Union (IoU, also known as Jaccard index). Precision evaluates the ratio of true classifications as sharp-edge or boundary to the total number of classified points. In contrast, Recall measures the ratio of correctly classified sharp-edge or boundary points compared to the true number of such points existing in the processed model. Precision and Recall are coupled, i.e. Precision increases and Recall decreases if only points exhibiting very high confidence are classified and vice versa. Thus, mainly the other mentioned metrics, which combine Precision and Recall scores in different ways, are used for directly comparing our BoundED technique to related works.

4.3. Comparison to related work

Throughout this section, we compare the performance of our work with the performance of several other recent related works for point cloud edge detection: Covariance Analysis (CA) (Bazazian et al., 2015), Feature Edges Estimation (FEE) (Alliez et al., 2022; Mérigot et al., 2011), ECNet (Yu et al., 2018), PIE-NET (Wang et al., 2020), PCP-Net (Guerrero et al., 2018), and PCEDNet (Himeur et al., 2021). The postfix *-2c* denotes that the respective algorithm has been trained for classification of two classes only, i.e. non-edge and sharp-edge, despite being originally designed to potentially handle more than two classes. For the quantitative evaluation (see Section 4.4), data reported by Himeur et al. (2021) is used for PCEDNet and PCPNet, while we use the numbers published by Wang et al. (2020) for ECNet and PIE-Net. For the two non-learning methods CA and FEE, we use one set of parameters each per dataset finetuned on the dataset's characteristics, i.e. more aggressive thresholding on clean data compared to noisy data: CA finetuned on *Default* uses 0.025 as threshold, while using 0.08 on *ABC*. The parameters for FEE are set to $R = 0.1$, $r = 0.03$ to work well with the *Default* dataset and to $R = 0.02$ and $r = 0.002$ to yield good results on the *ABC* dataset. In both cases, we use 0.16 as threshold. For FEE, we additionally normalize all point clouds to fit inside an axis-aligned unit box as R and r are related to the expected feature size, which varies heavily for the models in the *ABC* dataset. The PCEDNet results shown for the purpose of qualitative evaluation in Section 4.5 are generated using the publicly available precompiled demo application.[7] We assume only the point positions to be given as input for the algorithm. Since PCEDNet relies on point normals, these are generated according to the authors' specification using Meshlab (Cignoni et al., 2008). To be able to report meaningful numbers for the quantitative evaluation in Section 4.4, we have done every experiment five times, evaluated the MCC metric over the test set, and chose the run corresponding to the median MCC score.

To ensure practicality of our algorithm, timings are reported for two different hardware configurations: On the one hand, we use an old consumer-grade Nvidia RTX 2080 Ti GPU with 11 GB memory and an AMD Ryzen 3600X CPU with 32 GB memory. On the other hand, we also used the recent enterprise Nvidia A40 GPU with 48 GB memory and two AMD EPYC 7313 CPUs with 32 threads each and 512 GB memory. Note, however, that we only used 12 worker threads in the data loader during training for both hardware configurations. We exclude the IO and network initialization time from the timings listed in this section and focus on reporting the time required by the actual feature extraction as well as network inference instead.

4.4. Quantitative comparison

Tables 1 and 2 show median scores of various commonly used metrics to allow a quantitative comparison of our approach with others. For all experiments in this section, we are working on the *Default* and *ABC* datasets and aim at distinguishing sharp-edge points from non-edge points.

When training and evaluation are done on the *Default* dataset, our algorithm performs better than all related works in all metrics except for Recall, i.e. BoundED is not able to identify quite as many sharp-edge points as others, but more of those points classified as being a sharp-edge point are actually correctly identified as such. As we are also using a smaller network in comparison to PCEDNet, this suggests, that our multi-scale features are better at describing the geometry of the local neighborhood in terms of sharp edges than their GLS-based features.

Also observe, that BoundED trained on *ABC* performs better than CA and FEE finetuned on *ABC* when evaluating on the *Default* dataset. Both non-learning approaches, i.e. CA and FFE, rely on setting a threshold

[1] Available at: https://sketchfab.com/3d-models/christ-church-and-dublin-city-council-b5f6bcce8ebc44a3b4bbb6b0fef067b3, accessed on 10/14/2022.
[2] Available at: https://www.irit.fr/recherches/STORM/MelladoNicolas/category/datasets/, accessed on 10/22/2022.
[3] Available at: https://sketchfab.com/3d-models/station-rer-6c636ca4793345e8ae12beb97b7d6359, accessed on 10/14/2022.
[4] Available at: http://plant3d.navlakhalab.net/shoots/public/view/plant/40, time point 33, accessed on 10/14/2022.
[5] Available at: http://plant3d.navlakhalab.net/shoots/public/view/plant/20, time point 30, accessed on 10/14/2022.
[6] Available at: http://plant3d.navlakhalab.net/shoots/public/view/plant/15, time point 30, accessed on 10/14/2022.
[7] Available at: https://storm-irit.github.io/pcednet-supp/software.html, accessed on 10/14/2022.

Table 2
Median scores of edge detection approaches evaluated on the *ABC* dataset. The dataset used for parameter tuning or training is mentioned in parentheses. Data regarding PCEDNet-2c and PCPNET-2c is taken from Himeur et al. (2021). Data regarding ECNet and PIE-NET is taken from Wang et al. (2020). Instead of estimating the normals from the noisy point cloud the methods marked with † utilize perfect normals taken from the original CAD model.

	Precision(↑)	Recall(↑)	MCC(↑)	F1(↑)	Accuracy(↑)	IoU(↑)
CA (Default) (Bazazian et al., 2015)	0.312	**0.991**	0.482	0.471	0.845	0.308
CA (ABC) (Bazazian et al., 2015)	0.498	0.820	0.541	0.574	0.929	0.403
FEE (Default) (Alliez et al., 2022; Mérigot et al., 2011)	0.178	0.621	0.213	0.270	0.775	0.156
FEE (ABC) (Alliez et al., 2022; Mérigot et al., 2011)	0.857	0.898	0.821	0.832	0.980	0.712
PCEDNet-2c (*Default*)† (Himeur et al., 2021)	0.662	0.936	0.708	0.730	0.958	0.574
PCEDNet-2c (*ABC*)† (Himeur et al., 2021)	0.735	0.984	0.808	0.822	0.970	0.597
ECNet (*ABC*) (Yu et al., 2018)	0.487	0.573	–	0.526	–	0.356
PIE-NET (*ABC*) (Wang et al., 2020)	0.692	0.858	–	0.766	–	0.622
PCPNet-2c (*ABC*) (Guerrero et al., 2018)	0.954	0.756	0.797	0.807	0.979	0.668
BoundED-2c (Ours) (*Default*)	0.599	0.484	0.518	0.531	0.951	0.361
BoundED-2c (Ours) (*ABC*)	**0.959**	0.857	**0.869**	**0.875**	**0.987**	**0.778**

Table 3
Comparison of time required to calculate the multi-scale features used as network input and training or evaluation time on the training or evaluation data respectively of the dataset in parentheses. Timings of our approach are determined on two different hardware configurations: An older consumer-grade Nvidia RTX 2080 Ti GPU with 11 GB memory and a recent enterprise-grade Nvidia A40 GPU with 48 GB memory. PCEDNet and PCEDNet-2c as well as respective preprocessing were executed on two 10-cores Intel Xeon(R) CPU E5-2640 v4.
Source: Data regarding PCEDNet and PCEDNet-2c is taken from Himeur et al. (2021).

	Training		Evaluation	
	Preprocessing	Training	Preprocessing	Classification
PCEDNet (*Default*) (Himeur et al., 2021)	0:19 m	2:52 m	–	–
BoundED-2c (Ours) (*Default*, RTX 2080 Ti)	0:06 m	1:24 m	1.4 s	0.002 s
BoundED-2c (Ours) (*Default*, A40)	0:05 m	1:08 m	1.4 s	0.005 s
PCEDNet-2c (*ABC*) (Himeur et al., 2021)	2:11 m	20:00 m	2:35:00 h	0:25:30 h
BoundED-2c (Ours) (*ABC*, RTX 2080 Ti)	1:25 m	2:24 m	2:21:09 h	0:00:04 h
BoundED-2c (Ours) (*ABC*, A40)	1:10 m	3:12 m	1:45:35 h	0:00:04 h
BoundED (Ours) (*Default*++, RTX 2080 Ti)	0:07 m	1:25 m	2.0 s	0.002 s
BoundED (Ours) (*Default*++, A40)	0:05 m	1:10 m	1.8 s	0.004 s

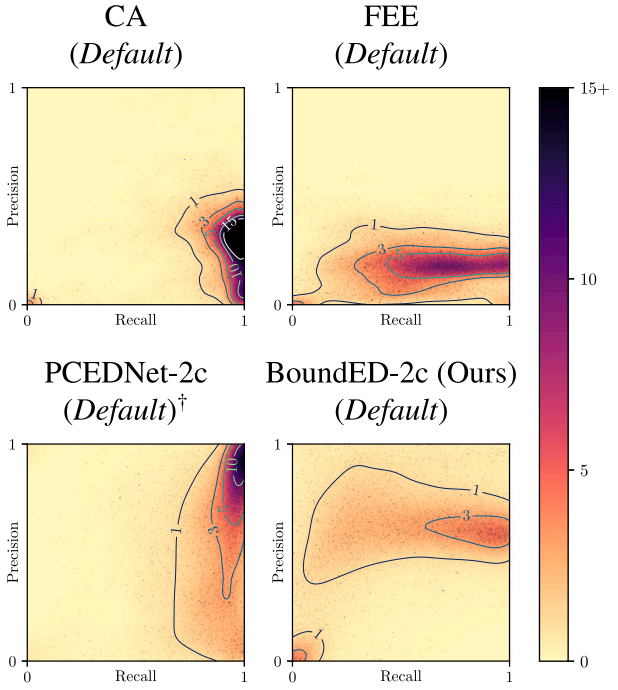

Fig. 6. Precision-Recall-plots of those approaches listed in Table 2 which were trained on the *Default* dataset. Every small semi-transparent black dot corresponds to a single point cloud from the *ABC* dataset and its Precision and Recall scores when being processed by the respective approach. The background depicts the color-coded local density of points. Instead of estimating the normals from the noisy point cloud the methods marked with † utilize perfect normals taken from the original CAD model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to distinguish between sharp-edge and non-edge points. On clean data like the models from the *ABC* dataset, this threshold can be set much more aggressively. In the presence of noise, this, however, leads to the algorithms not detecting all edges in the case of CA and tremendous overclassification of points as sharp-edge points in the case of FEE.

When being evaluated on *ABC*, BoundED trained on *ABC* once again outperforms all other approaches in terms of Precision, MCC, F1, Accuracy, and IoU scores, but PCEDNet loses less effectiveness if being trained on *Default* in comparison to BoundED. While CA trained on *Default* exhibits the highest Recall, it is worse in terms of overall classification performance due to having a much worse Precision score.

The Precision-Recall-plots shown in Figs. 6 and 7 confirm these observations. In these diagrams, every point cloud of the *ABC* dataset is depicted as one small semi-transparent black point according to its Precision and Recall scores. The background color depicts the color-coded local density of points. The plot for BoundED trained on the *ABC* dataset exhibits the highest density in the top right corner suggesting that the classification results on most models are of high quality, while the peak density for approaches trained or finetuned on *Default* is much lower and the individual points are more evenly distributed over a larger area. Since some point clouds of the *ABC* dataset exhibit local neighborhood characteristics that are not represented well in the *Default* dataset, some point clouds classified with BoundED-2c (*Default*) exhibit low precision and low recall. Fig. 8 depicts an example, that is badly classified by BoundED-2c (*Default*), but perfectly classified by BoundED-2c (*ABC*), indicating that the proposed features are capable of handling these cases given enough training data.

Besides yielding better classification scores across the board, the computation of our features is also cheaper compared to PCEDNet and our multi-scale fusion and classification network has roughly 20% fewer parameters. Table 3 lists training and evaluation timings for PCEDNet and our approach. Training in this context consists of the multi-scale feature extraction for the training and validation data of the dataset given in parentheses as well as using this data to train the network. Similarly, evaluation consists of extracting the features on the

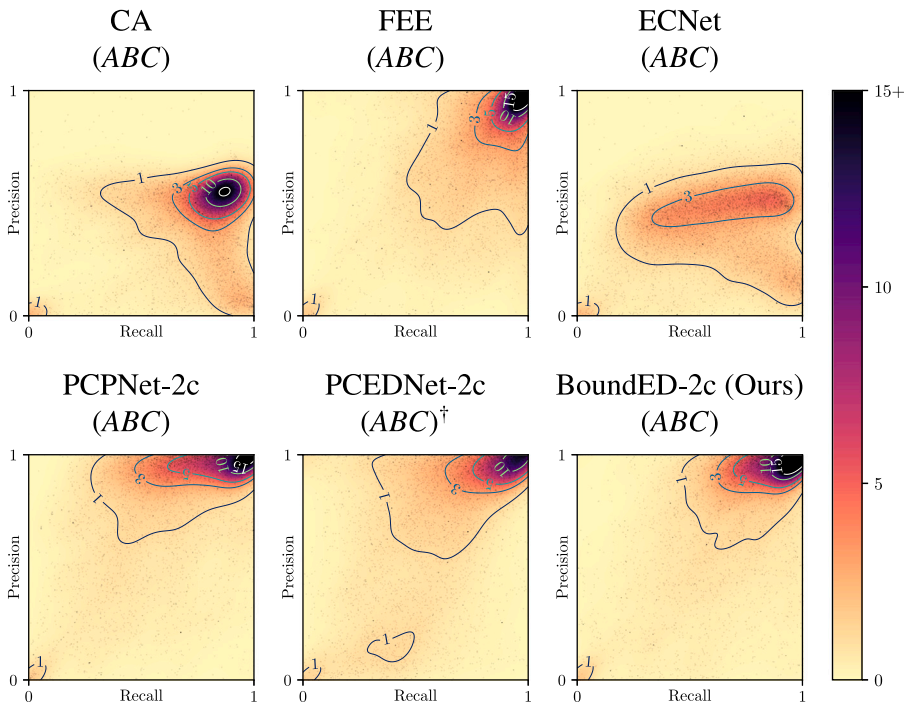

Fig. 7. Precision-Recall-plots of most approaches listed in Table 2 which were trained on the *ABC* dataset. Every small semi-transparent black dot corresponds to a single point cloud from the *ABC* dataset and its Precision and Recall scores when being processed by the respective approach. The background depicts the color-coded local density of points. Instead of estimating the normals from the noisy point cloud the methods marked with † utilize perfect normals taken from the original CAD model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

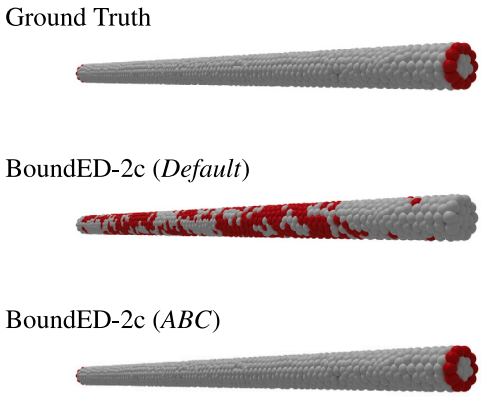

Fig. 8. Model 0251 of the *ABC* dataset evaluated on BoundED-2c trained on *Default* and *ABC*. While BoundED-2c (*Default*) fails to detect the sharp edges of the object, BoundED-2c (*ABC*) achieves perfect results.

evaluation set given in parentheses and classifying all points using the trained network.

While using a powerful GPU accelerates the feature extraction step, the difference for the network training and inference is negligible due to the network's compactness and simplicity.

4.5. Qualitative comparison

If trained on *Default*++, our algorithm learns to identify the sharp edges in the evaluation models well as can be seen in Fig. 9. The edge detection results seem to be even a bit more consistent than the ones of PCEDNet trained on *Default*. Note, that our filtering technique introduced in Section 3.2 is capable of detecting almost all outliers in these objects as depicted in Fig. 10.

Results on some evaluation models of the *ABC* dataset are depicted in Fig. 11. PCEDNet exhibits mixed performance on the models 0027 and 0059. Depending on the dataset used for training, the algorithm either tends to have more severe problems with the thin wall of model 0059 or produces less consistent results on some parts of model 0027. The most consistent results, however, are produced by our approach trained on the *Default*++ dataset. It is the only configuration that produces an inner circular edge on model 0027 without holes while not massively overclassifying the walls of model 0059 as sharp edge. The classification results of points which are part of the screw thread in model 0117 are not as consistent for BoundED trained on *Default* as the detected sharp edges contain many holes. The screw is classified best by our algorithm trained on *ABC* with BoundED trained on *Default*++ being a close second place.

BoundED also works well on actual 3D scanned real-world data as shown in Figs. 12 and 13. On the *christ_church* point cloud, it outperforms PCEDNet in classifying the sharp-edges of roofs (see green zoom-in) and also gives good results for the fine stone structures of the church (see blue zoom-in). The results on the *station* point cloud are similar. Especially for the third row, our algorithm gives much more consistent results in the area of the escalator.

4.6. Boundary detection

As already mentioned in Section 1, the processing of point clouds often requires the detection of boundaries in addition to sharp edges due to potentially very fine structures as well as finite resolution. This is especially important if the scanned object has many fine structures like leaves on plants or fine fins on buildings. Due to the GLS (Mellado et al., 2012) features used in PCEDNet (Himeur et al., 2021), which rely on point normals estimated using a small neighborhood of points, PCEDNet is by design not able to detect boundaries in point clouds. In contrast, using our proposed set of features and the extended *Default*++

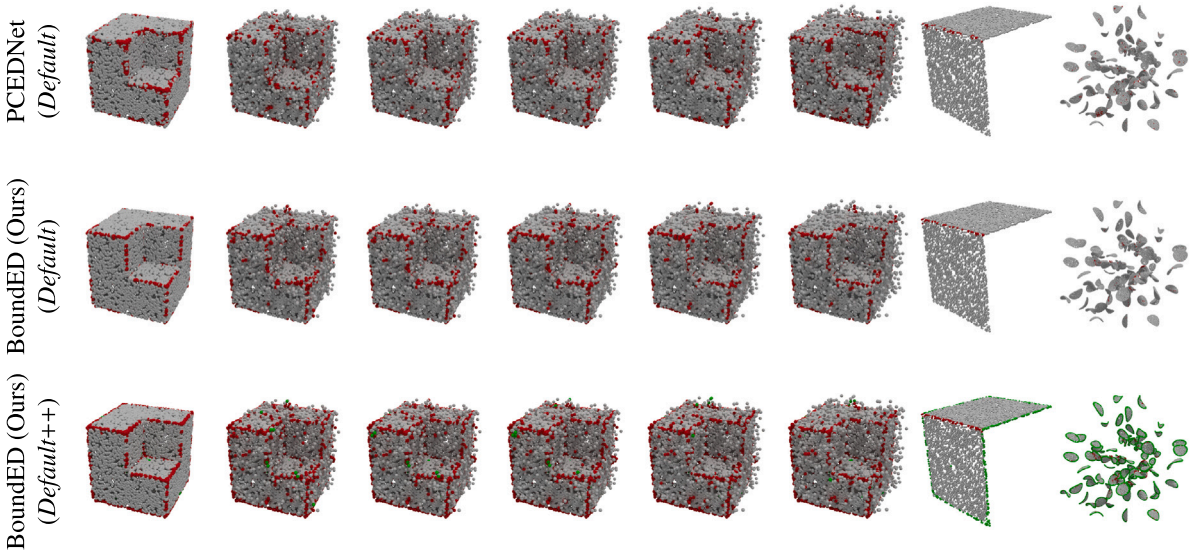

Fig. 9. Comparison of the results on the *Default*++ evaluation set. The dataset used for training is reported in parentheses. As first and second row were trained on the *Default* dataset, the respective approaches are by design not able to detect boundaries.

Table 4
Ablation study regarding the choice of features used as input for the network as well as the neighborhood filtering scheme. The table lists median scores for various classification metrics. *Default*++ dataset is used for training as well as evaluation. The individual features are defined in Sections 3.1 and 3.3.

		Precision(↑)	Recall(↑)	MCC(↑)	F1(↑)	Accuracy(↑)	IoU(↑)
With Filtering	$\boldsymbol{x}_{i,k} = (\boldsymbol{d}_{i,k}, \boldsymbol{s}_{i,k}, \boldsymbol{c}_{i,k}, \boldsymbol{r}_{i,k})^T$	0.383	0.834	0.525	0.523	0.908	0.355
	$\boldsymbol{x}_{i,k} = (\boldsymbol{\sigma}_{i,k,\text{upper}}, \boldsymbol{\sigma}_{i,k,\text{lower}}, \boldsymbol{s}_{i,k}, \boldsymbol{c}_{i,k}, \boldsymbol{r}_{i,k})^T$	0.263	0.458	0.291	0.334	0.888	0.200
	$\boldsymbol{x}_{i,k} = (\boldsymbol{\sigma}_{i,k,\text{upper}}, \boldsymbol{\sigma}_{i,k,\text{lower}}, \boldsymbol{d}_{i,k}, \boldsymbol{c}_{i,k}, \boldsymbol{r}_{i,k})^T$	0.240	0.166	0.157	0.196	0.917	0.109
	$\boldsymbol{x}_{i,k} = (\boldsymbol{\sigma}_{i,k,\text{upper}}, \boldsymbol{\sigma}_{i,k,\text{lower}}, \boldsymbol{d}_{i,k}, \boldsymbol{s}_{i,k}, \boldsymbol{r}_{i,k})^T$	0.225	0.400	0.243	0.291	0.874	0.170
	$\boldsymbol{x}_{i,k} = (\boldsymbol{\sigma}_{i,k,\text{upper}}, \boldsymbol{\sigma}_{i,k,\text{lower}}, \boldsymbol{d}_{i,k}, \boldsymbol{s}_{i,k}, \boldsymbol{c}_{i,k})^T$	**0.521**	0.618	0.548	0.576	**0.942**	0.404
	$\boldsymbol{x}_{i,k} = (\boldsymbol{\sigma}_{i,k,\text{upper}}, \boldsymbol{\sigma}_{i,k,\text{lower}}, \boldsymbol{d}_{i,k}, \boldsymbol{s}_{i,k}, \boldsymbol{c}_{i,k}, \boldsymbol{r}_{i,k})^T$	0.442	0.826	**0.574**	**0.578**	0.926	**0.406**
	$\boldsymbol{x}_{i,k} = (\boldsymbol{\sigma}_{i,k}, \boldsymbol{\sigma}_{i,k,\text{upper}}, \boldsymbol{\sigma}_{i,k,\text{lower}}, \boldsymbol{d}_{i,k}, \boldsymbol{s}_{i,k}, \boldsymbol{c}_{i,k}, \boldsymbol{r}_{i,k})^T$	0.393	**0.934**	0.567	0.549	0.907	0.378
Without Filtering	$\boldsymbol{x}_{i,k} = (\boldsymbol{\sigma}_{i,k,\text{upper}}, \boldsymbol{\sigma}_{i,k,\text{lower}}, \boldsymbol{d}_{i,k}, \boldsymbol{s}_{i,k}, \boldsymbol{c}_{i,k})^T$	0.332	0.886	0.499	0.483	0.884	0.318
	$\boldsymbol{x}_{i,k} = (\boldsymbol{\sigma}_{i,k,\text{upper}}, \boldsymbol{\sigma}_{i,k,\text{lower}}, \boldsymbol{d}_{i,k}, \boldsymbol{s}_{i,k}, \boldsymbol{c}_{i,k}, \boldsymbol{r}_{i,k})^T$	0.352	0.909	0.524	0.507	0.892	0.339

Table 5
Ablation study regarding the number of scales used by our network. The table lists median scores for various classification metrics. The *Default*++ dataset is used for training as well as evaluation.

	Precision(↑)	Recall(↑)	MCC(↑)	F1(↑)	Accuracy(↑)	IoU(↑)
2 scales (128, 32)	0.403	0.875	0.559	0.552	0.913	0.381
4 scales (128, 64, 32, 16)	**0.442**	0.826	**0.574**	**0.578**	**0.926**	**0.406**
8 scales (128, 91, 64, 45, 32, 23, 16, 11)	0.356	**0.934**	0.538	0.516	0.893	0.348
16 scales (128, 108, 91, 76, 64, 54, 45, 38, 32, 27, 23, 19, 16, 13, 11, 10)	0.347	0.882	0.506	0.500	0.887	0.333

Table 6
Median scores of our edge detection approach BoundED trained and evaluated on the *Default* dataset with (second row) and without (first row) smooth-edge points of the original dataset being treated as sharp-edge points.

	Precision(↑)	Recall(↑)	MCC(↑)	F1(↑)	Accuracy(↑)	IoU(↑)
BoundED-2c (Ours) (*Default*)	**0.597**	0.566	0.555	0.581	**0.950**	0.410
BoundED-2c (Ours) (*Default-Soft*)	0.521	**0.675**	**0.566**	**0.591**	0.943	**0.419**

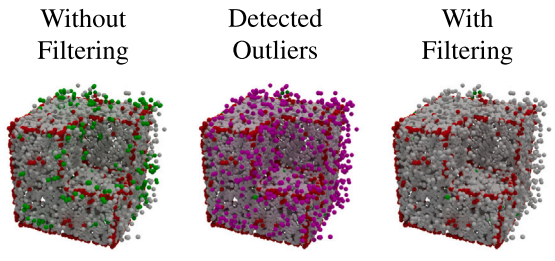

Fig. 10. Impact of outlier detection. Detected outliers are colored purple in the middle column. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

dataset makes our approach capable of detecting boundaries in addition to sharp edges.

Fig. 9 shows successfully detected boundaries for the two rightmost models, i.e. the only ones containing actual boundary points. For model 1222 of the *ABC* datasets evaluation data (see Fig. 11), the boundary is found almost perfectly as well. Despite being actually 3D structures and therefore not boundaries in the strict sense, the top of the walls of model 0059 are detected as a boundary as well. Due to the low thickness of the walls, this is a reasonable behavior depending on the exact use-case for the extracted boundary data.

Very thin structures being identified correctly as boundary can also be seen in the red zoom-in of Fig. 12. In the *station* point cloud (see Fig. 13), mostly points of thin signs and humans are identified as boundary points. Note, that humans in this point cloud are

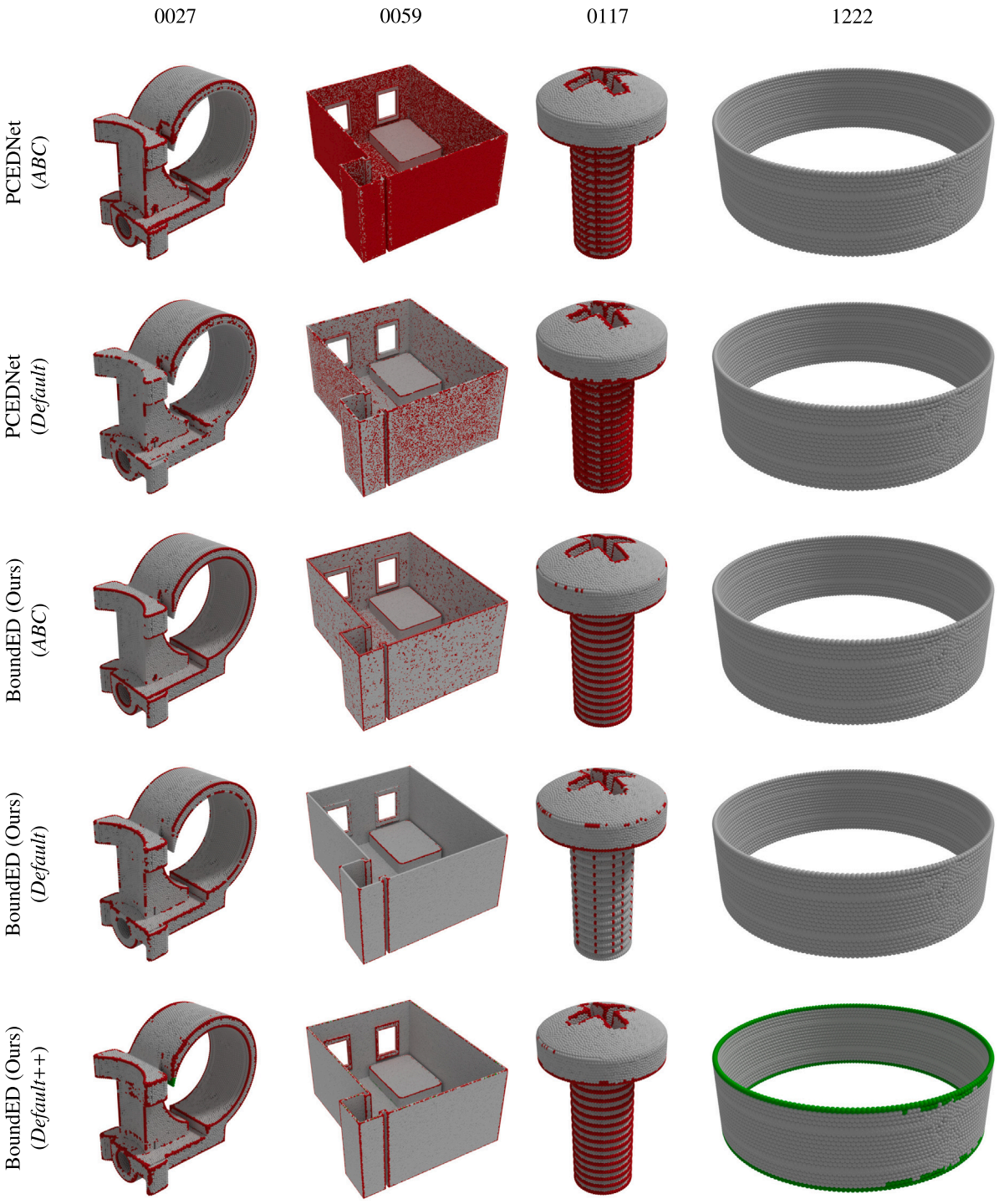

Fig. 11. Comparison of PCEDNet and BoundED trained on three different datasets and evaluated on four different models from the *ABC* evaluation dataset. The dataset used for training the respective approach is given in parentheses. Algorithms trained on *Default* or *ABC* are not able to detect boundaries by design.

mostly two-dimensional due to the scanning procedure and rather low resolution.

Finally, results on plants are depicted in Fig. 14. All leaves are nicely separated by boundaries. Some stems contain sharp-edge points due to scanning artifacts.

4.7. Behavior on corrupted data

In addition to the results on clean point clouds in Fig. 11, Fig. 15 shows a direct comparison on clean as well as noisy data taken from the *ABC* dataset of our algorithm BoundED and PCEDNet (Himeur et al., 2021). The respective noisy models are taken from Himeur et al. (2021).[8] Note, that we assume only the point positions to be given. Thus, the normals needed by PCEDNet were calculated according to the author's instructions via meshlab (Cignoni et al., 2008). BoundED outperforms PCEDNet on noisy data if both are trained on *Default* as it is significantly less prone to predict false positives in originally flat regions. The difference is particularly noticeable in the eighth row on

[8] Available at: https://storm-irit.github.io/pcednet-supp/abc_noise_0.04.html, accessed on 10/19/2022.

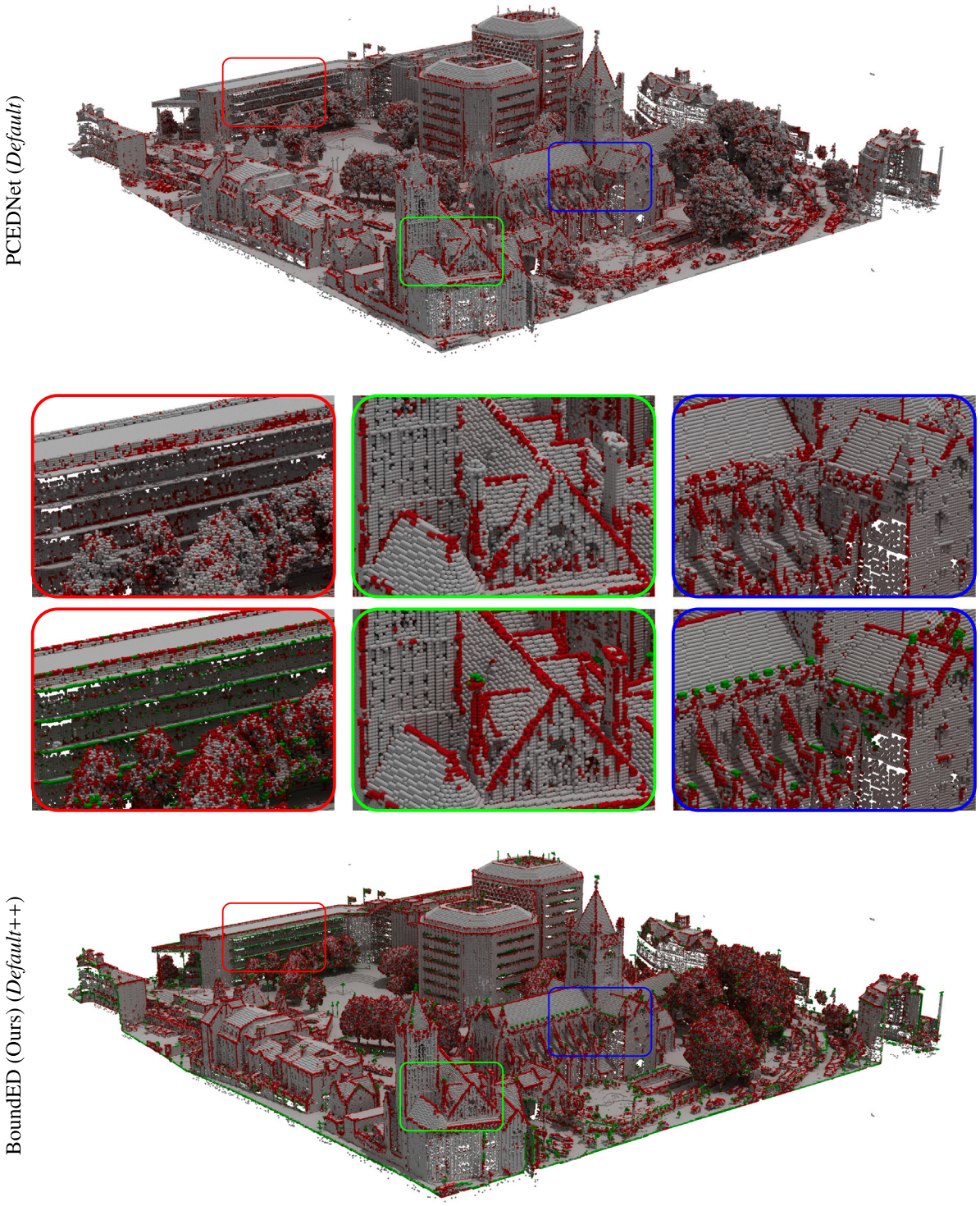

Fig. 12. Classification result of PCEDNet trained on *Default* (top) and BoundED trained on *Default++* (bottom) for the mid-sized (1.9 million points) scanned *christ_church* point cloud. Three different zoomed parts are depicted for direct comparison (middle). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

model 7487. Furthermore, in the sixth row on model 4986, PCEDNet has difficulties in detecting the prominent sharp edges at the top and bottom of the object. As the network architectures of both approaches are very similar, we expect that the main reason for our approach to perform better in the presence of noise is the additional robustness of our features due to the underlying statistics.

Further experiments were conducted on a point cloud containing typical laser scanning characteristics like points being distributed very anisotropically and with varying density. The results depicted in Fig. 16 demonstrate, that BoundED is mostly robust against such artifacts and still yields reasonable results. The robustness concerning varying point density is further displayed in Fig. 17 showing classification results on simple synthetic point clouds. Whereas we focus a simple approach to achieve efficiency in terms of processing time for our current approach, consistency in resolving ambiguities, e.g. points belonging to thin structures or slight bumps in otherwise planar regions, could still be

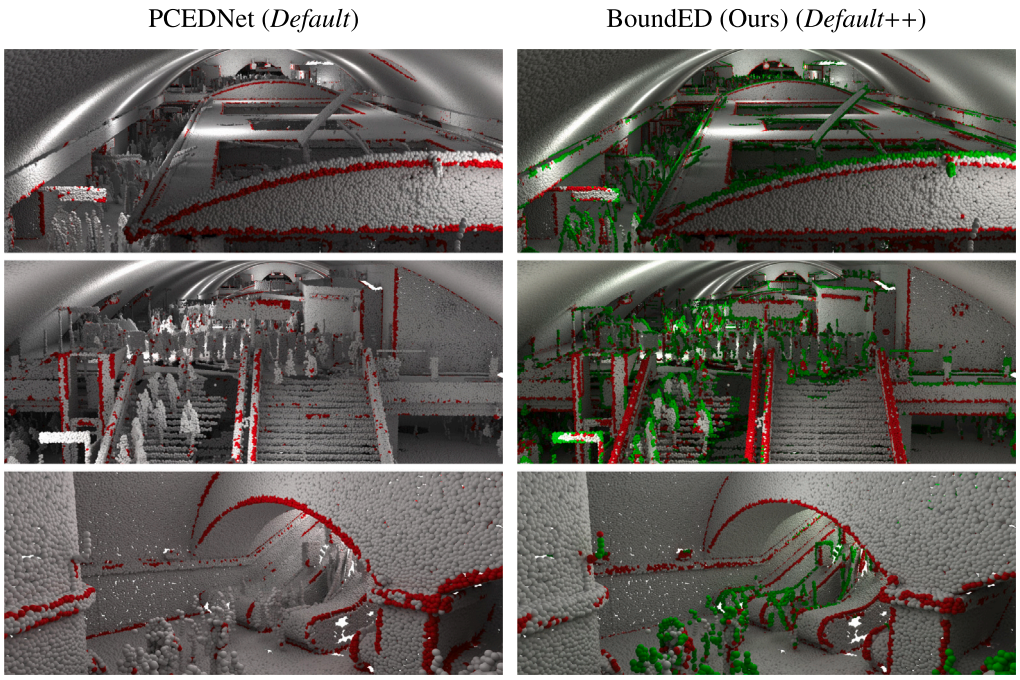

Fig. 13. Classification result of PCEDNet trained on *Default* (left) and BoundED trained on *Default*++ (right) for the large (12.5 million points) scanned *train_station* point cloud.

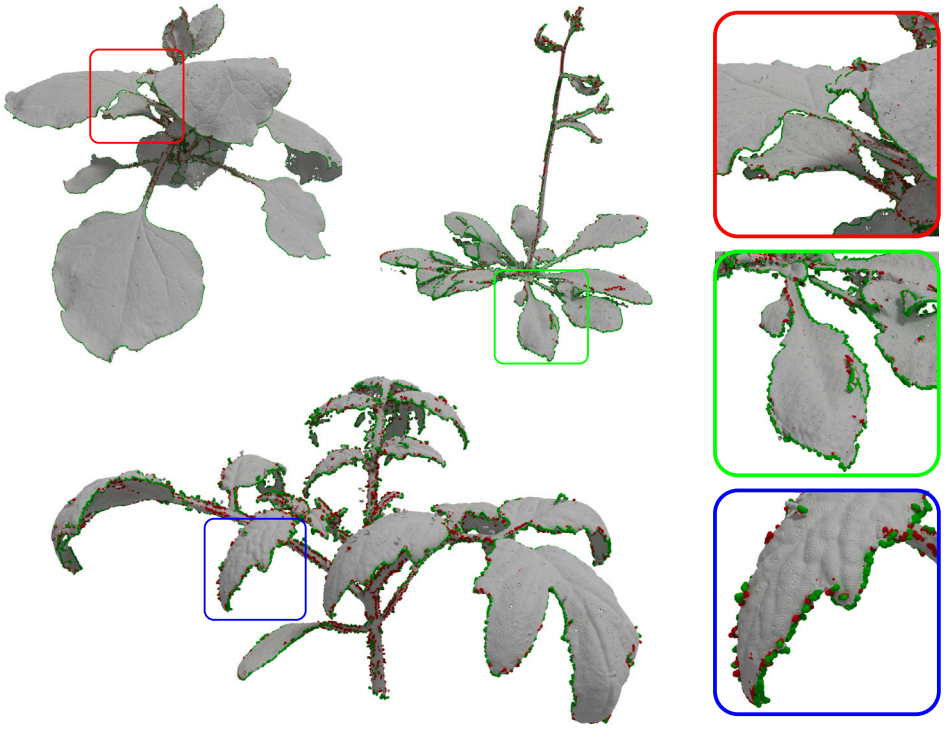

Fig. 14. 3D scanned plant point clouds classified using our approach BoundED trained on *Default*++.

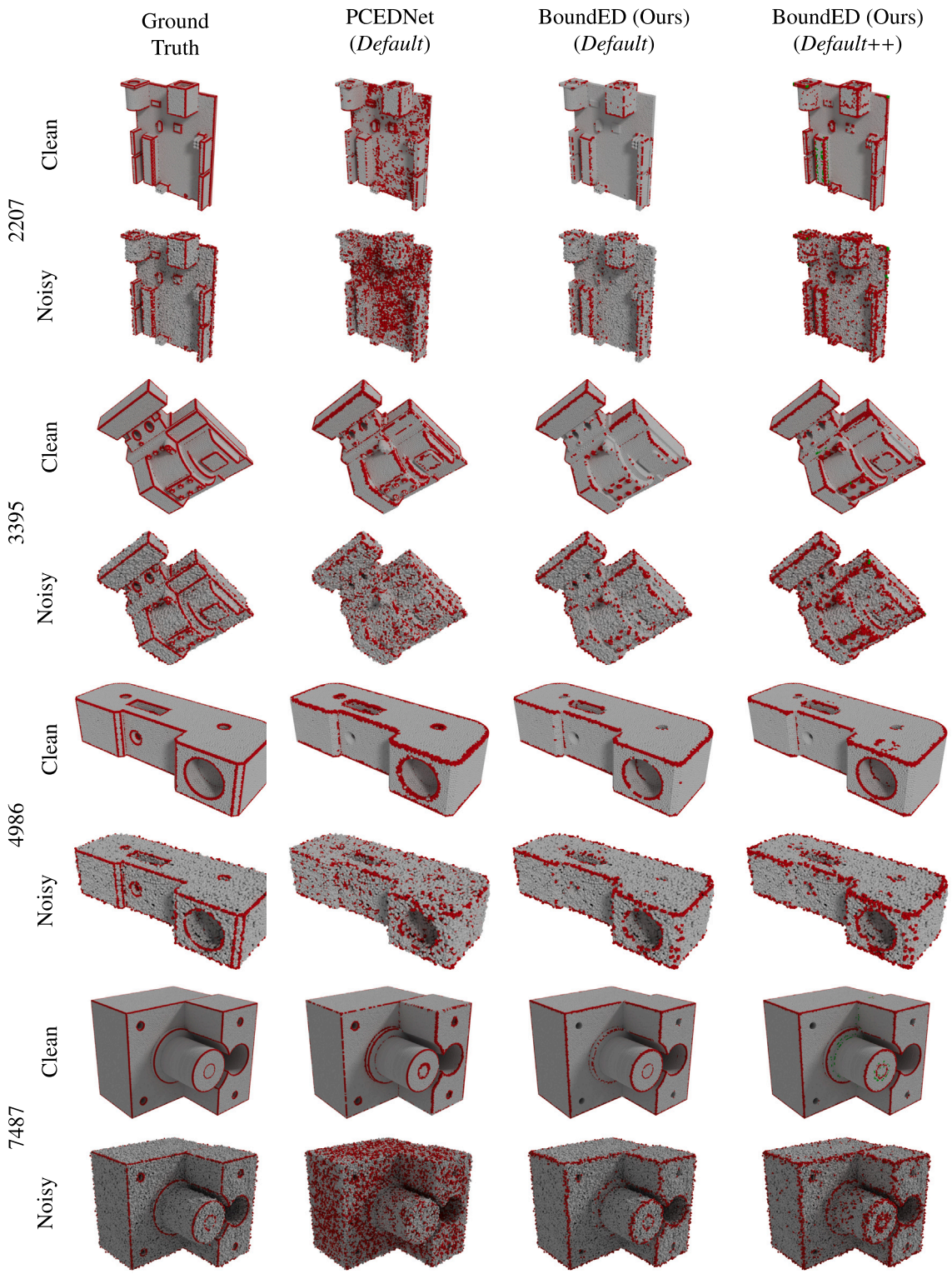

Fig. 15. Comparison of PCEDNet and BoundED regarding behavior on noisy data. The dataset used for training the respective approach is given in parentheses.

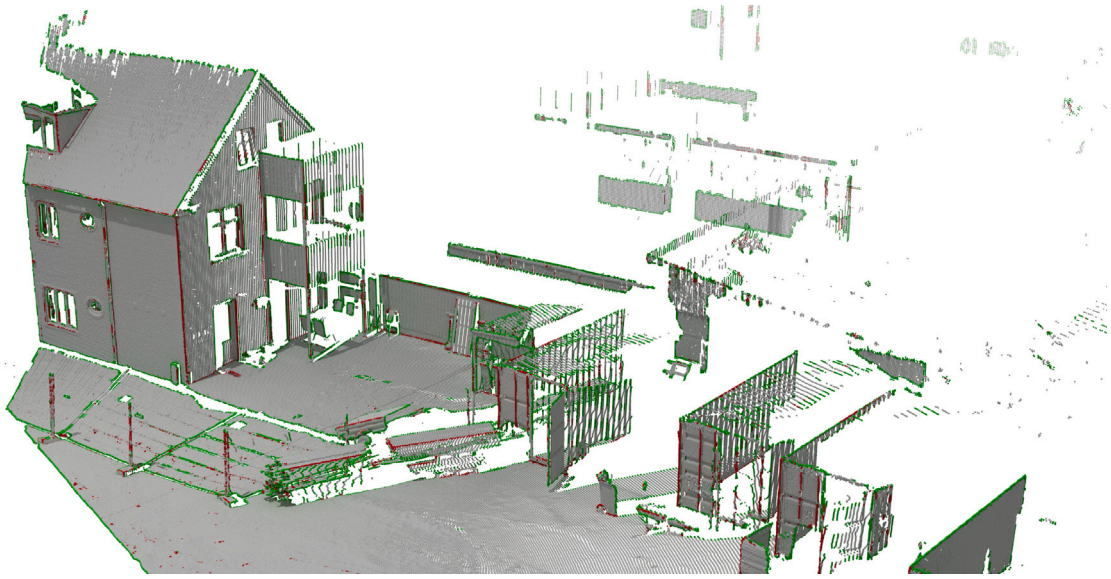

Fig. 16. Classification result for a point cloud exhibiting typical laser scanning artifacts using our BoundED approach trained on the *Default*++ dataset.

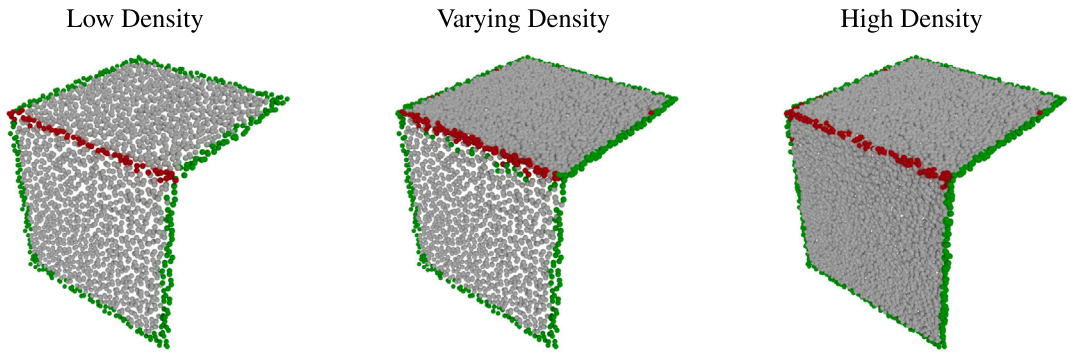

Fig. 17. Comparison of classification results using BoundED trained on the *Default*++ dataset for point clouds of various densities.

improved. Furthermore, when the point density varies extremely across a sharp edge, artifacts in the form of points being wrongly classified as boundary can be observed as depicted in the middle image in Fig. 17. A feature designed to capture local density variation could improve the results as part of future work.

4.8. Ablation study

In the scope of an additional ablation study, we validate the chosen features, the selected number of scales, as well as the sharp-edge definition.

Table 4 shows median scores of various classification metrics for results of our approach trained and evaluated on the *Default*++ dataset. While all chosen features seem to contribute positively to the classification result, experiments suggest, that $\boldsymbol{s}_{i,k}$ is the most important feature. We suspect the reason for this to be the high importance of its tangential component for the detection of boundaries while additionally the normal component can be utilized by the network to classify sharp-edge points. Passing the singular values $\boldsymbol{\sigma}_{i,k}$ of the unpartitioned neighborhood's covariance matrix to the network in addition to $\boldsymbol{\sigma}_{i,k,\text{upper}}$ and $\boldsymbol{\sigma}_{i,k,\text{lower}}$ yields worse results. We suspect that the additional information gained for including this feature is not able to offset the additional complexity in the network input data. Increasing network capacity and training time could be used to mitigate this effect, if training speed and compactness of the network are not of concern for a particular application. Note, that, while the neighborhood filtering scheme described in Section 3.2 drastically increases classification performance, even just including $\boldsymbol{r}_{i,k}$ as an input feature for the network without actually filtering out outliers is beneficial.

The impact of the number of scales is shown in Table 5. For the experiments, we chose to use 2^i neighbors per scale where i is distributed evenly spaced over the interval $(3, 7]$. Similarly to adding $\boldsymbol{\sigma}_{i,k}$ to the network input, our experiments show, that extracting the features over more than four scales is not beneficial regarding the classification performance of our method due to the increase in complexity. Hence, we use four scales.

Since the original *Default* dataset (Himeur et al., 2021) includes an additional smooth-edge class, which contains points being close to sharp edges, we investigate the impact of treating these points as sharp-edge or as non-edge points for our approach in Table 6. If smooth-edge points are treated as sharp-edge points during training and evaluation, the Precision decreases as smooth-edge points exhibit similar characteristics to non-edge points and, hence, the algorithm is more liberal in classifying points as sharp-edge. This more liberal classification on the other hand leads to higher Recall scores at the same time. Note, that all other metrics are very similar. Hence, if a specific application relies on this less strict edge point definition, the overall results presented in this work are expected to hold in this case as well.

Finally, we investigate the benefits of our chosen filtering approach with regard to alternatives. For this purpose, we compare our outlier detection component with a network based outlier detection proposed in the PointCleanNet approach (Rakotosaona et al., 2020). Note that

Table 7
Comparison of processing time for outlier detection. Our approach allows a significant speed-up in comparison to the neural network based outlier detection component used in PointCleanNet (Rakotosaona et al., 2020).

	Cube (7.2k points)	Tomato (226.4k points)
PointCleanNet (Rakotosaona et al., 2020)	4.88 s	125.3 s
BoundED-2c (Ours) (*Default*)	**0.12 s**	**4.2 s**

PointCleanNet's outlier detection also relies on a user-defined threshold, and we followed the authors' selection of the threshold value of 0.5. In the timings reported in Table 7, we observe a significantly faster outlier detection with our simple approach which shows the benefits of following a light-weight approach that does not involve a complex network architecture for outlier detection.

4.9. Limitations

Despite yielding great results in most cases, the feature extraction step can fail in various scenarios. If e.g. the smallest singular value of a neighborhood's covariance matrix does not correspond to the true surface normal, the points are partitioned in an unexpected way leading to very unpredictable results. Note, that, by estimating the normal per-scale and passing all respective singular values to the network, it is able to extract additional information about the neighborhood. Using only a single-scale normal e.g. estimated during the scanning process might therefore lead to less accurate classifications. To some degree, this can be compensated by the proposed filtering technique and by providing enough training data to the classification network. Nonetheless, the results would surely improve if the feature extraction step can already tackle such edge cases on its own by e.g. using a per-scale global normal smoothing step.

Furthermore, the *Default* dataset was designed to yield good results if GLS features are used for classification, but it does not cover all relevant edge cases for our features. Designing a new point cloud dataset with our features in mind or even generating a dataset based on the feature values directly instead of going the detour over generating point clouds could solve this problem and improve results e.g. in the case of varying point density as mentioned in Section 4.8.

Finally, depending on the point cloud size, a large part of the time needed for the feature extraction step is spent on finding the k neighbors of each point. A custom-tailored solution for this neighborhood search could probably improve the performance of the feature extraction significantly. Due to the simplicity and compactness of the network, the same holds for the implementation of the classification network as general frameworks like PyTorch introduce significant overhead in this situation.

5. Conclusion and future work

In this work, we introduced a novel set of per-point features extracted from filtered local point neighborhoods to facilitate the detection of sharp edges and boundaries via a simple and compact neural classification network. Due to the small network and an efficient GPU implementation for the feature extraction, the algorithm is faster than previous state-of-the-art methods while at the same time achieving more consistent classification results. This could make the proposed BoundED algorithm a good choice for situations in which interactive classification is required.

The two-level covariance analysis conducted on the neighborhood of a point has, even in the simple form deployed in this work, proven to be a valuable tool to describe the local geometry. In the future, our novel features could be utilized to estimate the curvature of curved surfaces as well. We expect the inclusion of higher-order moments to further improve the results and enable us to also learn the estimation of distances to edges and boundaries.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The point clouds shown in Fig. 16 were provided by The Royal Danish Academy of Fine Arts Schools of Architecture, Design and Conservation (CITA).

Funding

This work was supported by the DFG-Project KL 1142/9-2.

References

Ahmed, S.M., Tan, Y.Z., Chew, C.M., Al Mamun, A., Wong, F.S., 2018. Edge and corner detection for unorganized 3d point clouds with application to robotic welding. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 7350–7355.

Alliez, P., Giraudot, S., Jamin, C., Lafarge, F., Mérigot, Q., Meyron, J., Saboret, L., Salman, N., Wu, S., Yildiran, N.F., 2022. Point set processing. In: CGAL User and Reference Manual, 5.5 ed. CGAL Editorial Board, URL https://doc.cgal.org/5.5/Manual/packages.html\char"0023PkgPointSetProcessing3.

Bazazian, D., Casas, J.R., Ruiz-Hidalgo, J., 2015. Fast and robust edge extraction in unorganized point clouds. In: 2015 International Conference on Digital Image Computing: Techniques and Applications. DICTA, IEEE, pp. 1–8.

Bendels, G.H., Schnabel, R., Klein, R., 2006. Detecting Holes in Point Set Surfaces. Václav Skala-Union Agency.

Blomley, R., Weinmann, M., 2017. Using multi-scale features for the 3d semantic labeling of airborne laser scanning data. ISPRS Ann. Photogram. Remote Sens. Spat. Inf. Sci. 4.

Bode, L., Merzbach, S., Kaltheuner, J., Weinmann, M., Klein, R., 2022. Locally-guided neural denoising. Graph. Vis. Comput. 200058. http://dx.doi.org/10.1016/j.gvc.2022.200058, URL https://www.sciencedirect.com/science/article/pii/S2666629422000110.

Boulch, A., Le Saux, B., Audebert, N., 2017. Unstructured point cloud semantic labeling using deep segmentation networks. 3DOR@ Eurographics 3, 1–8.

Brodu, N., Lague, D., 2012. 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. ISPRS J. Photogram. Remote Sens. 68, 121–134.

Che, E., Olsen, M.J., 2018. Multi-scan segmentation of terrestrial laser scanning data based on normal variation analysis. ISPRS J. Photogram. Remote Sens. 143, 233–248.

Chen, J., Guennebaud, G., Barla, P., Granier, X., 2013. Non-oriented MLS gradient fields. In: Computer Graphics Forum. Vol. 32. No. 8. Wiley Online Library, pp. 98–109.

Chen, H., Liang, M., Liu, W., Wang, W., Liu, P.X., 2022. An approach to boundary detection for 3D point clouds based on DBSCAN clustering. Pattern Recognit. 124, 108431. http://dx.doi.org/10.1016/j.patcog.2021.108431, URL https://www.sciencedirect.com/science/article/pii/S0031320321006075.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G., 2008. MeshLab: An open-source mesh processing tool. In: Scarano, V., Chiara, R.D., Erra, U. (Eds.), Eurographics Italian Chapter Conference. The Eurographics Association, http://dx.doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

Conn, A., Pedmale, U.V., Chory, J., Stevens, C.F., Navlakha, S., 2017. A statistical description of plant shoot architecture. Curr. Biol. 27 (14), 2078–2088.

Daniels, J., Ochotta, T., Ha, L.K., Silva, C.T., et al., 2008. Spline-based feature curves from point-sampled geometry. Vis. Comput. 24 (6), 449–462.

Demantké, J., Mallet, C., David, N., Vallet, B., 2011. Dimensionality based scale selection in 3D lidar point clouds. In: Laserscanning.

Demarsin, K., Vanderstraeten, D., Volodine, T., Roose, D., 2007. Detection of closed sharp edges in point clouds using normal estimation and graph theory. Comput. Aided Des. 39 (4), 276–283.

Edelsbrunner, H., Mücke, E.P., 1994. Three-dimensional alpha shapes. ACM Trans. Graph. 13 (1), 43–72.

Fan, T.-J., Medioni, G., Nevatia, R., 1987. Segmented descriptions of 3-D surfaces. IEEE J. Robot. Autom. 3 (6), 527–538.

Fernandes, L.A., Oliveira, M.M., 2012. A general framework for subspace detection in unordered multidimensional data. Pattern Recogn. 45 (9), 3566–3579.

Fleishman, S., Cohen-Or, D., Silva, C.T., 2005. Robust moving least-squares fitting with sharp features. ACM Trans. Graph. (TOG) 24 (3), 544–552.

Gelfand, N., Guibas, L.J., 2004. Shape segmentation using local slippage analysis. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. pp. 214–223.

Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A., 2012. A kernel two-sample test. J. Mach. Learn. Res. 13 (1), 723–773.

Guerrero, P., Kleiman, Y., Ovsjanikov, M., Mitra, N.J., 2018. PCPNet: Learning local shape properties from raw point clouds. In: Computer Graphics Forum. Vol. 37. No. 2. Wiley Online Library, pp. 75–85.

Gumhold, S., Wang, X., MacLeod, R.S., et al., 2001. Feature extraction from point clouds. In: IMR. pp. 293–305.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M., 2020. Deep learning for 3d point clouds: A survey. IEEE Trans. Pattern Anal. Mach. Intell. 43 (12), 4338–4364.

Guo, B., Zhang, Y., Gao, J., Li, C., Hu, Y., 2022. SGLBP: Subgraph-based local binary patterns for feature extraction on point clouds. In: Computer Graphics Forum. Wiley Online Library.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J.D., Schindler, K., Pollefeys, M., 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. pp. 91–98.

Hackel, T., Wegner, J.D., Schindler, K., 2016a. Contour detection in unstructured 3D point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1610–1618.

Hackel, T., Wegner, J.D., Schindler, K., 2016b. Fast semantic segmentation of 3D point clouds with strongly varying density. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. 3, 177–184.

Himeur, C.-E., Lejemble, T., Pellegrini, T., Paulin, M., Barthe, L., Mellado, N., 2021. PCEDNet: A lightweight neural network for fast and interactive edge detection in 3D point clouds. ACM Trans. Graph. 41 (1), 1–21.

Hu, Z., Zhen, M., Bai, X., Fu, H., Tai, C.-l., 2020. Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds. In: European Conference on Computer Vision. Springer, pp. 222–239.

Huang, J., You, S., 2016. Point cloud labeling using 3d convolutional neural network. In: 2016 23rd International Conference on Pattern Recognition. ICPR, IEEE, pp. 2670–2675.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., Panozzo, D., 2019. ABC: A big CAD model dataset for geometric deep learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9601–9611.

Landrieu, L., Raguet, H., Vallet, B., Mallet, C., Weinmann, M., 2017. A structured regularization framework for spatially smoothing semantic labelings of 3D point clouds. ISPRS J. Photogramm. Remote Sens. 132, 102–118.

Landrieu, L., Simonovsky, M., 2018. Large-scale point cloud semantic segmentation with superpoint graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4558–4567.

Lawin, F.J., Danelljan, M., Tosteberg, P., Bhat, G., Khan, F.S., Felsberg, M., 2017. Deep projective 3D semantic segmentation. In: International Conference on Computer Analysis of Images and Patterns. Springer, pp. 95–107.

Li, M., Hashimoto, K., 2017. Curve set feature-based robust and fast pose estimation algorithm. Sensors 17 (8), 1782.

Li, J., Weinmann, M., Sun, X., Diao, W., Feng, Y., Hinz, S., Fu, K., 2022. VD-LAB: A view-decoupled network with local-global aggregation bridge for airborne laser scanning point cloud classification. ISPRS J. Photogramm. Remote Sens. 186, 19–33.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2980–2988.

Lin, Y., Wang, C., Cheng, J., Chen, B., Jia, F., Chen, Z., Li, J., 2015. Line segment extraction for large scale unorganized point clouds. ISPRS J. Photogramm. Remote Sens. 102, 172–183.

Loizou, M., Averkiou, M., Kalogerakis, E., 2020. Learning part boundaries from 3d point clouds. In: Computer Graphics Forum. Vol. 39. No. 5. Wiley Online Library, pp. 183–195.

Lu, X., Liu, Y., Li, K., 2019. Fast 3D line segment detection from unorganized point cloud. arXiv preprint arXiv:1901.02532.

Mao, Y., Chen, K., Diao, W., Sun, X., Lu, X., Fu, K., Weinmann, M., 2022. Beyond single receptive field: A receptive field fusion-and-stratification network for airborne laser scanning point cloud classification. ISPRS J. Photogramm. Remote Sens. 188, 45–61.

Matveev, A., Rakhimov, R., Artemov, A., Bobrovskikh, G., Egiazarian, V., Bogomolov, E., Panozzo, D., Zorin, D., Burnaev, E., 2022. DEF: Deep estimation of sharp geometric features in 3D shapes. ACM Trans. Graph. 41 (4), 1–22.

Mellado, N., Dellepiane, M., Scopigno, R., 2015. Relative scale estimation and 3D registration of multi-modal geometry using Growing Least Squares. IEEE Trans. Vis. Comput. Graph. PP (99), 1. http://dx.doi.org/10.1109/TVCG.2015.2505287.

Mellado, N., Guennebaud, G., Barla, P., Reuter, P., Schlick, C., 2012. Growing least squares for the analysis of manifolds in scale-space. In: Computer Graphics Forum. Vol. 31. No. 5. Wiley Online Library, pp. 1691–1701.

Mérigot, Q., Ovsjanikov, M., Guibas, L.J., 2011. Voronoi-based curvature and feature estimation from point clouds. IEEE Trans. Vis. Comput. Graphics 17 (6), 743–756.

Mineo, C., Pierce, S.G., Summan, R., 2019. Novel algorithms for 3D surface point cloud boundary detection and edge reconstruction. J. Comput. Des. Eng. 6 (1), 81–91.

Mitropoulou, A., Georgopoulos, A., 2019. An automated process to detect edges in unorganized point clouds. ISPRS Ann. Photogram. Remote Sens. Spat. Inf. Sci. 4.

Monga, O., Deriche, R., Rocchisani, J.-M., 1991. 3D edge detection using recursive filtering: Application to scanner images. CVGIP: Image Understand. 53 (1), 76–87.

Nguyen, K.W.L., Aprilia, A., Khairyanto, A., Pang, W.C., Seet, G.G.L., Tor, S.B., 2018. Edge detection from point cloud of worn parts. In: Proceedings of the 3rd International Conference on Progress in Additive Manufacturing. Pro-AM 2018, pp. 595–600.

Ni, H., Lin, X., Ning, X., Zhang, J., 2016. Edge detection and feature line tracing in 3D-point clouds by analyzing geometric properties of neighborhoods. Remote Sens. 8 (9), 710.

Niemeyer, J., Rottensteiner, F., Soergel, U., 2014. Contextual classification of lidar data and building object detection in urban areas. ISPRS J. Photogram. Remote Sens. 87, 152–165.

Öztireli, A.C., Guennebaud, G., Gross, M., 2009. Feature preserving point set surfaces based on non-linear kernel regression. In: Computer Graphics Forum. Vol. 28. No. 2. Wiley Online Library, pp. 493–501.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Vol. 32. Curran Associates, Inc, pp. 8024–8035, URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pauly, M., Keiser, R., Gross, M., 2003. Multi-scale feature extraction on point-sampled surfaces. In: Computer Graphics Forum. Vol. 22. No. 3. Wiley Online Library, pp. 281–289.

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017a. PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660.

Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017b. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems. Vol. 30.

Raina, P., Mudur, S.P., Popa, T., 2018. MLS2: Sharpness field extraction using CNN for surface reconstruction. In: Graphics Interface. pp. 66–75.

Raina, P., Mudur, S., Popa, T., 2019. Sharpness fields in point clouds using deep learning. Comput. Graph. 78, 37–53. http://dx.doi.org/10.1016/j.cag.2018.11.003, URL https://www.sciencedirect.com/science/article/pii/S009784931830181X.

Rakotosaona, M.-J., La Barbera, V., Guerrero, P., Mitra, N.J., Ovsjanikov, M., 2020. PointCleanNet: Learning to denoise and remove outliers from dense point clouds. In: Computer Graphics Forum. Vol. 39. No. 1. Wiley Online Library, pp. 185–203.

Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.-Y., Johnson, J., Gkioxari, G., 2020. Accelerating 3D deep learning with PyTorch3D. arXiv:2007.08501.

Rusu, R.B., Marton, Z.C., Blodow, N., Dolha, M., Beetz, M., 2008. Towards 3D point cloud based object maps for household environments. Robot. Auton. Syst. 56 (11), 927–941.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15 (1), 1929–1958.

Steinsiek, M., Polewski, P., Yao, W., Krzystek, P., 2017. Semantische analyse von ALS- und MLS-daten in urbanen gebieten mittels conditional random fields. Tagungsband 37, 521–531.

Tabib, R.A., Jadhav, Y.V., Tegginkeri, S., Gani, K., Desai, C., Patil, U., Mudenagudi, U., 2020. Learning-based hole detection in 3D point cloud towards hole filling. Procedia Comput. Sci. 171, 475–482.

Tang, P., Huber, D., Akinci, B., 2007. A comparative analysis of depth-discontinuity and mixed-pixel detection algorithms. In: Sixth International Conference on 3-D Digital Imaging and Modeling. 3DIM 2007, IEEE, pp. 29–38.

Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S., 2017. Segcloud: Semantic segmentation of 3d point clouds. In: 2017 International Conference on 3D Vision. 3DV, IEEE, pp. 537–547.

Thomas, H., Qi, C.R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L.J., 2019. KPCONV: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6411–6420.

Trinh, T.H., Tran, M.H., et al., 2015. Hole boundary detection of a surface of 3D point clouds. In: 2015 International Conference on Advanced Computing and Applications. ACOMP, IEEE, pp. 124–129.

Von Gioi, R.G., Jakubowicz, J., Morel, J.-M., Randall, G., 2008. LSD: A fast line segment detector with a false detection control. IEEE Trans. Pattern Anal. Mach. Intell. 32 (4), 722–732.

Wang, Q., Sohn, H., Cheng, J.C., 2019. Development of high-accuracy edge line estimation algorithms using terrestrial laser scanning. Autom. Constr. 101, 59–71.

Wang, X., Xu, Y., Xu, K., Tagliasacchi, A., Zhou, B., Mahdavi-Amiri, A., Zhang, H., 2020. PIE-NET: Parametric inference of point cloud edges. In: Advances in Neural Information Processing Systems. Vol. 33. pp. 20167–20178.

Weber, C., Hahmann, S., Hagen, H., 2010. Sharp feature detection in point clouds. In: 2010 Shape Modeling International Conference. IEEE, pp. 175–186.

Weber, C., Hahmann, S., Hagen, H., Bonneau, G.-P., 2012. Sharp feature preserving MLS surface reconstruction based on local feature line approximations. Graph. Models 74 (6), 335–345.

Weinmann, M., Jutzi, B., Hinz, S., Mallet, C., 2015a. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS J. Photogramm. Remote Sens. 105, 286–304.

Weinmann, M., Schmidt, A., Mallet, C., Hinz, S., Rottensteiner, F., Jutzi, B., 2015b. Contextual classification of point cloud data by exploiting individual 3D neigbourhoods. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. II-3 (2015), Nr. W4 2 (W4), 271–278.

Weinmann, M., Urban, S., Hinz, S., Jutzi, B., Mallet, C., 2015c. Distinctive 2D and 3D features for automated large-scale scene analysis in urban areas. Comput. Graph. 49, 47–57.

Xia, S., Wang, R., 2017. A fast edge extraction method for mobile LiDAR point clouds. IEEE Geosci. Remote Sens. Lett. 14 (8), 1288–1292.

Xiao, R., Xu, Y., Hou, Z., Chen, C., Chen, S., 2019. An adaptive feature extraction algorithm for multiple typical seam tracking based on vision sensor in robotic arc welding. Sensors Actuators A 297, 111533. http://dx.doi.org/10.1016/j.sna.2019.111533, URL https://www.sciencedirect.com/science/article/pii/S0924424719307605.

Xie, Y., Tian, J., Zhu, X.X., 2020. Linking points with labels in 3D: A review of point cloud semantic segmentation. IEEE Geosci. Remote Sens. Mag. 8 (4), 38–59.

Yagüe-Fabra, J., Ontiveros, S., Jiménez, R., Chitchian, S., Tosello, G., Carmignato, S., 2013. A 3D edge detection technique for surface extraction in computed tomography for dimensional metrology applications. CIRP Ann. 62 (1), 531–534.

Yu, L., Li, X., Fu, C.-W., Cohen-Or, D., Heng, P.-A., 2018. EC-Net: An edge-aware point set consolidation network. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 386–402.

Yu, M., Zhao, T., Alliez, P., Lafarge, F., 2022. Sharp feature consolidation from raw 3D point clouds via displacement learning.