Eurographics Symposium on Geometry Processing 2020
Q. Huang and A. Jacobson
(Guest Editors)

Volume 39 (2020), Number 5

Learning Part Boundaries from 3D Point Clouds

Marios Loizou [1,2], Melinos Averkiou [1,2], Evangelos Kalogerakis[3]

[1] University of Cyprus, [2] RISE CoE, Nicosia, Cyprus, [3] University of Massachusetts Amherst

Abstract

We present a method that detects boundaries of parts in 3D shapes represented as point clouds. Our method is based on a graph convolutional network architecture that outputs a probability for a point to lie in an area that separates two or more parts in a 3D shape. Our boundary detector is quite generic: it can be trained to localize boundaries of semantic parts or geometric primitives commonly used in 3D modeling. Our experiments demonstrate that our method can extract more accurate boundaries that are closer to ground-truth ones compared to alternatives. We also demonstrate an application of our network to fine-grained semantic shape segmentation, where we also show improvements in terms of part labeling performance.

CCS Concepts

• ***Computing methodologies*** → ***Neural networks; Point-based models;*** *Shape analysis;*

1. Introduction

Segmenting 3D objects into their constituent parts with accurate boundaries is a fundamental problem in computer graphics and vision. Although there has been significant amount of research in detecting contours and object boundaries in natural images with neural networks [BST15a, BST15b, BST16, CKUF17, HLC*18, Kok15, LLD19, LKV*18, LCF*18, LCH*17, MPTAG17, SWW*15, WZLH18, XT17], detecting boundaries in 3D point clouds is largely an unexplored area. Despite the significant advances in the area of 3D deep learning for processing unstructured point clouds, most research has focused so far on assigning part tags to individual points. The resulting segmentations often suffer from artifacts at areas that lie near the boundaries of parts, since the point assignments become highly uncertain at these areas (see also Figure 1).

In this paper, we present a neural network approach that learns to detect part boundaries in point clouds of 3D shapes. There is a number of technical challenges to overcome in developing an approach that addresses this problem. First, the notion of an object part is often ambiguous and usually depends on the task. For example, in semantic segmentation, parts follow label definitions (e.g., leg, back, seat for chairs), while for 3D modeling tasks, shapes are often modeled as collections of geometric primitives (e.g., spheres, cylinders, surfaces of extrusion, NURBS, and so on). We show that an effective boundary detector can be trained from semantic segmentation datasets to accurately localize boundaries of labeled parts, and also from shape datasets with segmented geometric patches. Second, boundaries are usually sparse; only a small percentage of points in a point cloud lie near boundaries. During training, we employ a sampling procedure to gather a sufficient amount of boundary points for training, and use a classification loss function robust to the imbalance of the number of boundary versus non-boundary points. Furthermore, in contrast to semantic segmentation networks that often rely on points expressed in global coordinate frames, we found that learning features from points expressed in local frames aligned with surface normals are better suited for boundary extraction. The output of our method is probabilistic: it assigns a probability for each point belonging to a part boundary or not. We demonstrate pairwise terms that can easily adopt these probabilities within graph cuts formulations.

We conducted a number of experiments to validate our method. First, we compare our extracted boundaries with annotated ones in geometric and semantic segmentation tasks. We found that the boundaries produced by our architecture are much closer to ground-truth ones compared to alternatives. For example, we observed that the error was reduced by 61.2% compared to the best alternative edge detector we adapted for our task (EC-Net, [YLF*18]), measured based on Chamfer distance between detected and ground-truth boundaries in the ABC dataset [KMJ*19]. We also show that our boundary detector, when combined with graph cuts, offers a small, but noticeable boost in terms the semantic segmentation performance: an increase of +2.6% in shape Intersection over Union (IoU), and +0.5% in part IoU on average in PartNet [MZC*19] compared to using a neural network (DGCNN [WSL*19]) that assigns tags to points without explicitly considering boundaries. Our contributions can be summarized as follows:

- a neural network module, called LocalEdgeConv (inspired by DGCNN), that operates on point cloud neighborhoods expressed in local frames (in constrast to global frames used in [WSL*19]). We found that this adaptation is more suitable for the task of 3D part boundary detection.
- a network training procedure that robustly samples and weights boundary data of either semantic parts or geometric primitives.
- a graph cuts formulation that uses our probabilistic boundary detector to improve semantic shape segmentation, especially near part boundaries.

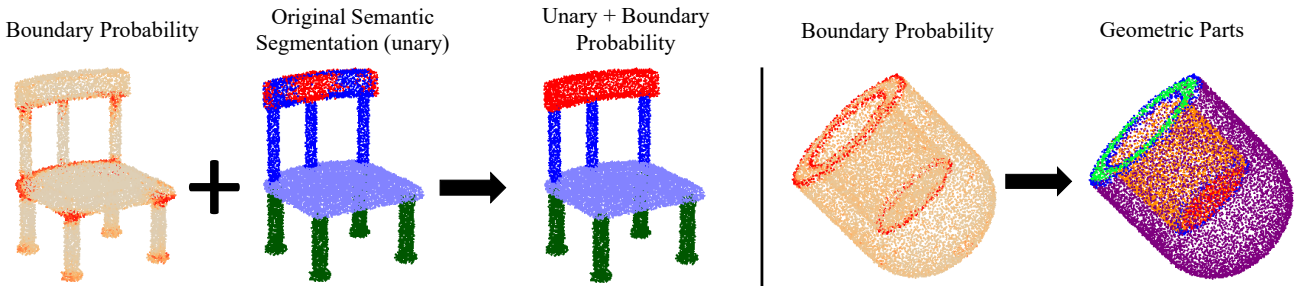

Figure 1: *Our method predicts part boundaries in 3D point clouds using a graph convolutional network which outputs a probability per point to lie on a boundary between parts in a 3D shape. The output probability per point can be used in pairwise terms to improve graph-based semantic segmentation methods (left) by localizing boundaries between semantic parts. It can also be used in the geometric decomposition of point clouds into regions enclosed by sharp boundaries detected by our method (right).*

2. Related Work

Below we briefly overview prior work on 3D point cloud segmentation as well as contour detection in 2D natural images and 3D shapes.

3D point cloud segmentation. A large number of works have been proposed to segment point clouds by assigning a part label for each point. Most recent semantic segmentation approaches train deep neural networks that learn representations from point clouds based on point set abstraction and aggregation functions [QYSG17, LCL18, SGS19, KZH19, LKM19, SKM19], point convolution operators [LBS*18, HTY18, LFM*19, GWL18, AML18, HRV*18, XFX*18, WQL19, TQD*19], graph convolution [WSL*19, LFXP19, LMTG19, JZL*19, XSyW*19, WHH*19, PK20], convolution on hierarchical grids [RUG17, KL17, WLG*17, WSLT18, SJS*18], point-to-voxel mappings [RWS*18, SWL19, LTLH19], view-based projections [KAMC17, HKC*17], and spectral approaches [YSGG17, BBL*17]. Apart from semantic segmentation, a number of approaches have also been proposed to perform geometric decompositions of point clouds based on convexity analysis [vKFK*14, SWS*14, DGY*19], primitive fitting [LSD*19, SWK07, LWC*11, ZYY*17, SLK*20], graph cuts [GF09, KMFF13, KT18], and clustering [RBM*07, BS16, ZZ19]. In both semantic segmentation and geometric decomposition scenarios, part boundaries often tend to become fuzzy and noisy (Figure 1 - left). To improve the quality of segmentation and align boundaries with underlying surface feature curves, such as creases, some methods employ simple geometric criteria, most commonly normal differences [PNH*19, KMFF13, SWS*14], within pairwise terms modeling the probability of boundaries between points. Similar pairwise terms have also been used in mesh segmentation approaches [KT03, GF08, KHS10, vKFK*14].

In contrast, our method learns a pairwise term indicating the existence of part boundaries directly on 3D point clouds. As we show in our experiments, our learned boundaries are more accurate and are able to improve the quality of semantic segmentation and geometric decomposition to a larger degree compared to other alternatives.

Feature curve and edge detection on 3D shapes. Our work is also related to learning methods for detecting edges, or feature curves on 3D shapes. This is because part boundaries often coincide with surface feature curves, such as creases, ridges and valleys. Detecting such feature curves relies on geometric features, such as curvature extrema or normal discontinuities that can be detected on meshes, RGB-D data or point clouds [OBS04, BAM*05, CTC13, CJXH17, KST09, KNS*09, SJW*11, HWG*13], however, their extraction often depends on hand-tuned procedures. Most similar to our approach, EC-Net [YLF*18] attempts to learn edges on point clouds using a deep architecture operating on isolated point cloud regions (patches). Our method instead trains a neural network that operates directly on the whole point cloud encoding both local and global structure without any patch extraction or pre-processing. In our experiments, we show that our approach is much more accurate for part boundary detection compared to EC-Net even when the latter is trained on the same dataset as our method.

Contour detection for natural images. Our approach is inspired by contour detection methods for object segmentation in natural images. Most recent approaches achieve state-of-the-art performance on contour detection by training deep convolutional or recurrent neural networks [Kok15, BST15a, BST15b, BST16, SWW*15, LCH*17, MPTAG17, CKUF17, XT17, WZLH18, HLC*18, LCF*18, LKV*18, LLD19]. In contrast to the regular 2D grid structure of images, point clouds are unorganized and non-uniformly sampled. Our method adapts neural networks for point cloud processing and is trained on datasets for 3D semantic or geometric segmentation of shapes.

3. Method

Our architecture takes as input a point cloud $\mathbf{P} = \{\mathbf{p}_i, \mathbf{n}_i\}_{i=1}^{N}$, where $\mathbf{p}_i$ are 3D point coordinates and $\mathbf{n}_i$ are 3D normals, and outputs a scalar $b_i \in [0,1]$ for each point. The output b_i represents the probability for a part boundary to lie on the point i. Our architecture is shown in Figure 2. We first describe it at test time (Section 3.1), then we discuss the datasets used for training it, along with the training procedure (Sections 3.2 and 3.3). We present an application of our probabilistic boundary detector to semantic shape segmentation in Section 4.3.

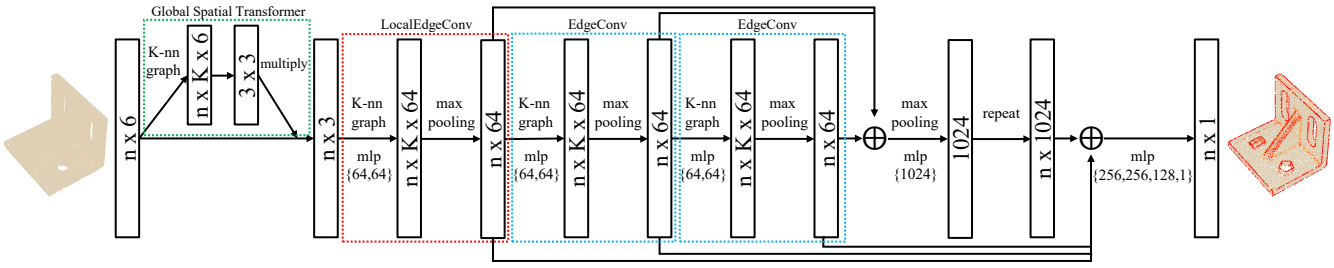

Figure 2: *Architecture of our network (PB-DGCNN) for probabilistic boundary detection. It consists of three main blocks: the LocalEdgeConv layer, EdgeConv layers [WSL*19], and a Global Spatial Transformer. The LocalEdgeConv layer constructs a K-NN graph for each input point i in Euclidean space and expresses the coordinates of its K nearest neighbors in the local coordinate frame at point i. Then a feature transformation is applied to the edge features of the graph through a 2-layer MLP, and output representations are aggregated through max-pooling. These representations are further processed by the two EdgeConv layers that operate on the K-nn graphs constructed in the feature space of the previous layer. Finally, a global descriptor is aggregated and concatenated with the point-wise descriptors of the three previous layers, which are transformed through a 4-layer MLP, and the boundary probability is produced by a sigmoid function. The Global Spatial Transformer [WSL*19] operates on the Euclidean space of the input points and helps to align the point cloud to a canonical space.*

3.1. Architecture

Our architecture, called PB-DGCNN, follows the concept of graph edge convolution (EdgeConv) introduced in the DGCNN network [WSL*19]. To implement edge convolution, a graph first needs to be formed over the point cloud. In the first EdgeConv layer, each point i is connected to its K neighbors in Euclidean space, where K is a hyper-parameter of the network. In the original DGCNN formulation, the first EdgeConv layer processes the input point coordinates $\mathbf{p}_i$ of each point i along with the coordinates of its neighbors $\{\mathbf{p}_j\}_{j\in\mathcal{N}_e(i)}$, where $\mathcal{N}_e(i)$ is the Euclidean neighborhood of the point i. The output representation for each point is computed as follows:

$$\mathbf{y}_i = \max_{j\in\mathcal{N}_e(i)} MLP(\mathbf{p}_i, \mathbf{p}_j - \mathbf{p}_i) \tag{1}$$

where *MLP* represents a learned Multi-Layer Perceptron operating on the above input feature vector of point coordinates and differences (concatenated and flattened). In this manner, each edge encodes the input coordinates at a point i along with the coordinates of its neighbors expressed relative to it. The max operator (max pooling) is used to aggregate all edge representations per point and guarantees invariance to point and edge permutations.

LocalEdgeConv layer. The relative point coordinate differences $(\mathbf{p}_j - \mathbf{p}_i)$ help capturing local neighborhood structure in the original DGCNN. However, these differences are still expressed with respect to the global coordinate frame axes. As a result, if the local neighborhood is rotated, the input edge features to the MLP will change. In turn, the output of the MLP may also change. This effect might be desirable in the case of semantic segmentation, since changing the orientation of a part in a shape may change its functionality and its semantic label, especially for man-made objects (e.g. rotating a horizontal tailplane 90 degrees in an airplane would make it look like a vertical stabilizer). However, the part boundaries are more likely to remain unaffected by such local rotations e.g., one would still want to label points between the fuselage and the tailplane, or the stabilizer, as boundaries. Thus, in our architecture, we make the following modification to the first edge convolution layer:

$$\mathbf{y}_i = \max_{j\in\mathcal{N}_e(i)} MLP\Big(\mathbf{p}_i, \mathbf{R}_i^T(\mathbf{p}_j - \mathbf{p}_i)\Big) \tag{2}$$

where $\mathbf{R}_i$ is a rotation matrix responsible for expressing the relative point coordinate differences in a local coordinate frame at point i (instead of a global one). Note that the transpose of the rotation is used to perform the coordinate transformation. The local frame is formed from the point normal $\mathbf{n}_i$ and two tangent vectors $\mathbf{u}, \mathbf{v}$ randomly selected on the tangent plane of the point i: $\mathbf{R}_i = [\mathbf{u}_i\ \mathbf{v}_i\ \mathbf{n}_i]$. Since the tangent vectors are chosen randomly, rotational invariance is not guaranteed, however, the use of the point normal decreases the variance of inputs that the network needs to handle. We call the above edge convolution of Eq. 2 as LocalEdgeConv in the rest of the paper. Experimentally, we observed a significant improvement in boundary detection due to LocalEdgeConv (see our evaluation in Section 4). We note that we also experimented with using principal curvature directions as tangent directions, and also treating their sign ambiguity through max pooling, yet the gain was still smaller than LocalEdgeConv (see results section).

LocalEdgeConv layer with normals as features. Another variant of LocalEdgeConv we experimented with was to include point normals as additional input features to this layer. Specifically, we horizontally concatenate point positions and normals per point $(\mathbf{x}_i = [\mathbf{p}_i, \mathbf{n}_i])$, then transform them through a MLP in a local coordinate system:

$$\mathbf{y}_i = \max_{j\in\mathcal{N}_e(i)} MLP\Big(\mathbf{x}_i, \mathbf{R}_i^T(\mathbf{x}_j - \mathbf{x}_i)\Big) \tag{3}$$

In this manner, the network also considers differences of normal coordinates in a neighborhood around each point transformed in a local coordinate frame. We note that we also experimented with processing points together with normals as input to the original EdgeConv as first layer (instead of LocalEdgeConv). However, the gain was smaller compared to using LocalEdgeConv with normals as input features.

Architecture. After using a LocalEdgeConv layer (with or without normals as additional features), our architecture stacks two

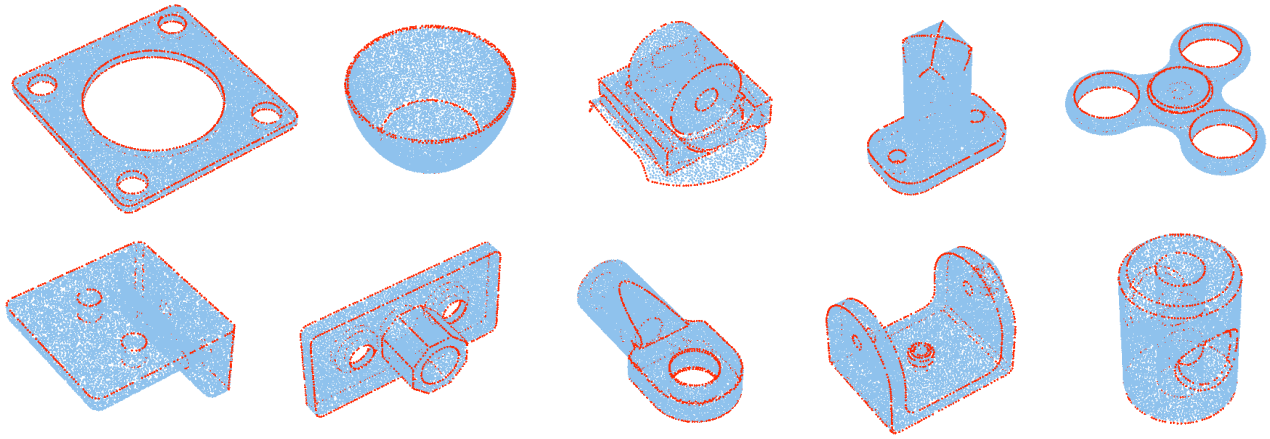

Figure 3: *Marked (with red) boundaries on ABC point clouds for training.*

EdgeConv layers (Figure 2) that sequentially process the representations extracted based on our local coordinate frames. At each EdgeConv layer, each point is connected its K nearest neighbors dynamically updated from the input feature space of the layer, as done in DGCNN [WSL*19]. The point-wise representations extracted from the LocalEdgeConv and the two EdgeConv layers are concatenated, then processed through a max pooling layer, which produces a global shape descriptor. The global descriptor is tiled and horizontally concatenated with the point-wise representations (Figure 2), so that the resulting point representations encode both local and global shape information. These are passed into a MLP, followed by a sigmoid transformation that outputs a boundary probability b_i for each point i.

3.2. Datasets

To train our network, we make use of datasets that provide shape segmentations. We made use of two datasets for training and evaluation: a geometric segmentation dataset and a semantic segmentation one, described below. We train and test our architecture on each dataset separately.

Geometric segmentation dataset. The ABC dataset [KMJ*19] recently introduced a large repository of 3D geometric models, each defined by parametric surfaces and ground truth information on their decomposition into individual patches. This dataset is a good source to learn segmentation boundaries between geometric primitives and patches. Another advantage of this dataset is that the patch boundaries are provided in parametric curve format, which allows us to extract very accurate boundaries for training and evaluation.

Since our goal is to detect boundaries for input point clouds, we first convert the geometric models into point-sampled surfaces for training. Specifically, we first sample the surface of 3D models with $10K$ points based on Poisson-Disk sampling [EDP*11], to create an initial point cloud. Since it is rather unlikely to sample points lying exactly on boundary curves with this sampling procedure, we perform a second pass where we randomly sample another $10K$ points along boundary curves, specifically based on their underlying parametric representation. Concatenating the surface point samples of the first pass with the boundary point samples of the second pass tends to create higher point cloud density near the boundary regions. To avoid this higher density bias during training, we perform a third pass where for each boundary point, we remove any surface samples within a distance equal to ε, which is computed by measuring the distance of each point sample of the first pass to its nearest neighbor, then setting it to the maximum distance over the point cloud. Finally, we observed that some ABC shapes sometimes contain adjacent patches of same local geometry (e.g., two adjacent planes forming a flat boundary), where the boundary between them can be ignored. We filtered out such boundaries. All shapes are centered in the origin and scaled so they lie inside the unit sphere.

The result of this procedure is the generation of a point cloud for each ABC shape with surface points carrying a binary label: boundary or non-boundary point. Figure 3 shows examples of such point clouds colored according to the binary label. We created a training set of $16,291$ labeled point clouds from ABC based on the above procedure.

To monitor the training procedure, we also need a hold-out validation set. In addition, for evaluation, we need a test set. We gathered an additional set of $2,327$ shapes for hold-out validation and $4,655$ shapes for testing i.e., in total we had $23,273$ point clouds from ABC, and a 70%-10%-20% proportion for training, validation and testing respectively. It is also important to note that the hold-out validation and testing point clouds are generated with Poisson point sampling from the original surfaces without adding boundary points (i.e., without the second and third pass used in training shapes). In this manner, we avoid biasing our testing procedure with point samples that are exactly at the geometric boundaries, and which may exhibit particular regular patterns due to their sampling from the underlying parametric representation of boundary curves. As discussed in our results section, the goal of our evaluation metrics is to detect boundaries up to a certain distance tolerance i.e., find points whose distance to the ground-truth parametric curve

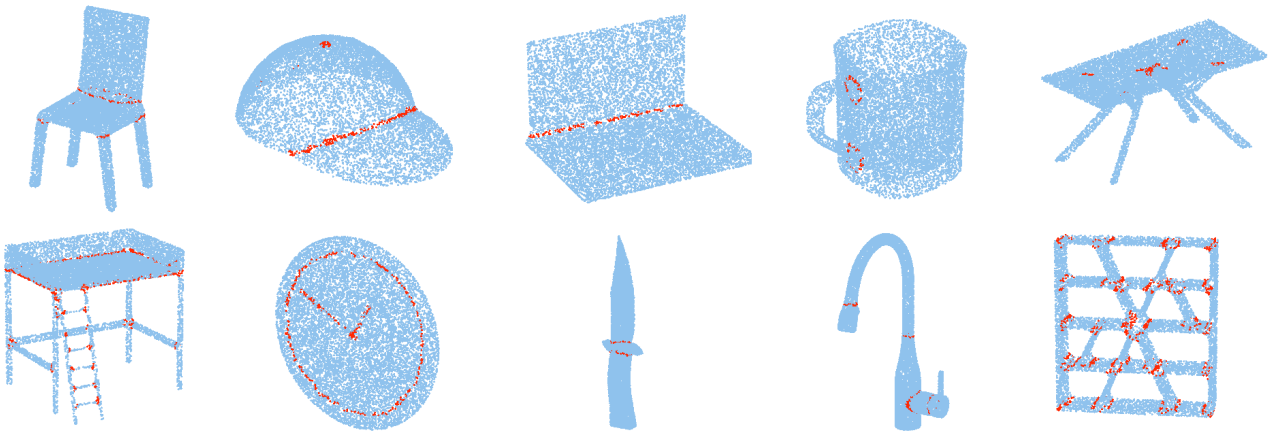

Figure 4: *Marked boundaries on PartNet point clouds for training.*

boundaries is up to distance equal to the maximum point sampling distance ε. Finally, to simulate noisy point clouds for validation and testing, we add isotropic Gaussian noise to point coordinates with $\mu = 0$ and $\sigma = 0.005$. Normals are also perturbed from their original direction, by an angle sampled from a normal distribution trimmed within an interval $[-3, 3]$ degrees.

Semantic segmentation dataset. To learn boundaries for semantic segmentation, we use the recent PartNet dataset [MZC*19]. The dataset provides hierarchical segmentations of 26,671 shapes into labeled parts in 24 categories. The shapes are provided in the form of polygon meshes split into parts according to their label. We use the segmentations from the last hierarchy level in each category (i.e., the "fine-grained" segmentations). To generate the training point clouds with boundaries, we follow the following procedure. First, we sample $10K$ points based on Poisson-Disk sampling. Partnet does not provide boundary curves. Furthermore, neighboring parts in PartNet meshes are topologically disconnected from each other in their mesh representation, often inter-penetrate each other, or even do not touch each other. We instead mark as boundaries all points of triangles that have neighboring points labeled with a different part label within a radius equal to ε set to the largest distance between all-pairs of nearest neighboring point samples. We add the same noise profile in the validation and test shapes, as in our ABC dataset. Figure 4 shows examples of the resulting point clouds, colored according to the binary label. The boundaries are more fuzzy and spread compared to the ABC dataset, yet are still clearly indicating zones separating semantic parts. PartNet provides training, hold-out validation, and test splits, thus we follow the same splits in our case.

3.3. Training procedure

To train our architecture, we use the marked boundary and non-boundary points from either of the above training datasets as supervisory signal. We treat the problem as binary classification, and we use binary cross-entropy as our loss function. However, since the number of boundary points is extremely small compared to the number of non-boundary ones (i.e., they represent less than 1% of the total points on average), we use a weighted cross entropy loss that emphasizes the error on boundary points more:

$$L = \sum_{s \in \mathcal{D}} \sum_{i=1}^{N_s} w_b \cdot \hat{t}_i \cdot \log(b_i) + (1 - \hat{t}_i) \cdot \log(1 - b_i) \qquad (4)$$

where $\hat{t}_i = 1$ for marked boundary points and $\hat{t}_i = 0$ for non-boundary points, and w_b weights the cross-entropy terms for boundary points. Specifically, we set the weight according to the ratio of the number of non-boundary points and the number of boundary points: $w_b = (\sum_i [\hat{t}_i == 0])/(\sum_i [\hat{t}_i == 1])$. In this manner, we penalize more misclassifications of boundary points. During training, as a form of data augmentation, and to also increase robustness, we add random noise in the points and normals (same noise distributions used in the validation and test sets of our datasets).

Implementation details. Training is done through the Adam optimizer [KB14] with learning rate 0.001, beta coefficients set to $(0.9, 0.999)$, batch normalization with momentum set to 0.5 and batch normalization decay set to 0.5 every 10 epochs. The batch size is set to 8 point clouds. Our implementation is in Tensorflow and is publicly available at: https://github.com/marios2019/learning_part_boundaries.

4. Evaluation

We now discuss experimental evaluation of our method. First, we introduce evaluation metrics for part boundary detection, and present results on the ABC dataset for geometric boundary detection. Then we present an application of our method to semantic segmentation, and present evaluation on the PartNet dataset.

4.1. Evaluation metrics

Our evaluation metrics are inspired by the literature on line drawing and segmentation for 3D meshes. Cole et al. [CGL*08] introduced metrics that evaluate similarity of human-annotated line drawings with computer-generated ones based on precision and recall. Liu

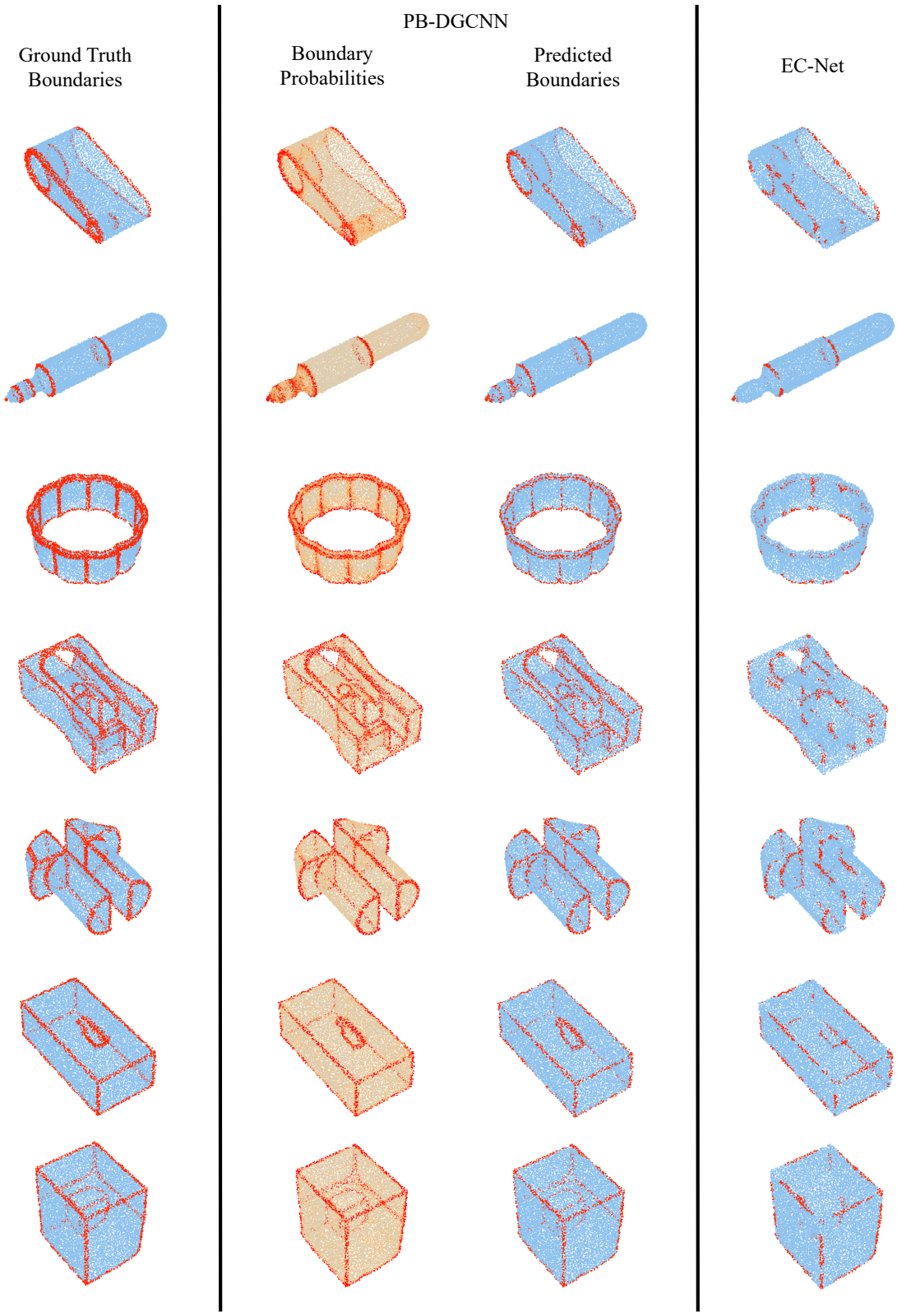

Figure 5: *Visual comparison of the boundaries detected by our method PB-DGCNN, and EC-Net on some example ABC point clouds. The first column on the left shows the ground truth boundaries. The second column shows boundary probabilities produced by PB-DGCNN, and the third column shows boundaries predicted by PB-DGCNN after thresholding. The last column shows the boundaries predicted by EC-Net.*

Model	Input Features		Metrics				
	position	normal	CD	bIoU	F_1	P	R
EC-Net	✓		4.9	52.7	64.6	88.5	50.9
	✓	✓	7.5	56.9	67.2	85.7	55.3
PB-DGCNN w/ EdgeConv	✓		3.0	81.6	85.2	89.8	81.1
	✓	✓	2.1	89.8	90.3	90.9	89.7
PB-DGCNN w/ LocalEdgeConv-curv	✓		2.6	85.2	88.0	91.3	85.8
	✓	✓	2.0	89.2	90.5	**91.8**	89.1
PB-DGCNN w/ LocalEdgeConv	✓		2.4	90.0	89.7	89.2	90.1
	✓	✓	**1.9**	**92.0**	**91.9**	**91.8**	**92.1**

Table 1: *Boundary classification results on the ABC dataset (CD: Chamfer Distance - %, bIoU: Boundary IoU - %, F_1: F_1 score - %, P: Precision - %, R: Recall - %)*

et al. [LNHK20] extended these metrics to include Intersection over Union (IoU). Our part boundaries can be thought of as point-sampled lines in 3D, thus we also use precision, recall, and IoU inspired by these works. Chen et al [CGF09] introduced various metrics for evaluating segmentation for 3D meshes. In the case of boundaries, they propose cut discrepancy that measures distances of annotated and predicted boundaries on the surface. Following the above works, we introduce the following metrics for evaluation boundaries:

Precision is defined as the fraction of predicted boundary points in a point cloud that are "near" any annotated boundary. The proximity is computed by measuring Euclidean distance of points to boundary curves in ABC dataset, or boundary point samples in PartNet. The definition of "near" requires a distance threshold indicating tolerance to small errors. We define this tolerance as the maximum point sampling distance ε (largest distance between all-pairs of nearest neighboring point samples per point cloud). We also examine performance under varying levels of tolerance (multiples of ε).

Recall is defined as the fraction of annotated boundary points that are "near" any predicted boundary point. We follow the same definition of nearness as above. In the ABC dataset, we densely sample the parametric boundary curves to evaluate recall.

F1-score is the harmonic mean of precision and recall, often used to combine them both in one metric.

Boundary IoU (bIoU) is the Intersection over Union that measures "overlap" between annotated boundaries and predicted ones. A boundary and predicted point "overlap" if they are near to each other, based on the same definition of nearness as above.

Chamfer distance (CD) measures Euclidean distance from annotated boundary samples to nearest predicted boundary points, and vice versa (i.e., we use the symmetric Chamfer distance).

It is important to note that in order to evaluate the above metrics, the probabilistic boundaries must be binarized first. In the ABC dataset, we use thresholding (i.e., a point becomes boundary if its probability is above a threshold). To select the threshold, we perform dense grid search in our hold-out validation dataset, and select the value that minimizes the Chamfer Distance. In the PartNet dataset, the probabilistic boundaries are used in a pairwise term in graph cuts - the points crossed by the cut are marked as boundaries. Finally, we note that the metrics are computed for each test point cloud shape, then averaged over the test shapes.

4.2. Geometric part boundary detection

We now discuss evaluation for detection of part boundaries between geometric primitives on ABC [KMJ*19] based on the dataset described in Section 3.2. The primitives in ABC include plane, cone, cylinder, sphere, torus, surface of revolution or extrusion or NURBS patch. We compare our method with the edge detection network called EC-Net from [YLF*18]. The method was introduced for detecting edges on point clouds for 3D reconstruction. It upsamples the original point cloud, while we also producing a value per point corresponding to its distance to the nearest edge. By thresholding the value, the method detects edges. We adapted their method for our task. We trained their method on our dataset, tuned their hyper-parameters (weights of losses) in our hold-out validation set, tuned the threshold for edge detection using hold-out validation to optimize Chamfer distance, and used the same augmentation as in our method. Since their method is based on sampling individual patches from the point cloud, we experimentally verified that the sampled patches fully cover the ABC shapes by setting their number to 50.

Table 1 reports our five evaluation metrics for EC-Net and our method. We evaluated two version of EC-Net: one with points only as input features (EC-Net w/o normals), and another with points and normals as input features (EC-Net w/ normals). As indicated by all metrics, our method produces boundaries that are much closer to the annotated ones compared to EC-Net. For example, the EC-Net without normals has 2.04 times higher error than our method without normals (see PB-DGCNN w/ LocalEdgeConv w/o normals) in terms of Chamfer Distance, and 2.58 times higher error than our method with normals (see PB-DGCNN w/ LocalEdgeConv w/ normals). The EC-Net with normals seems to have even higher error in Chamfer Distance, yet better Recall and IoU profile than EC-net without normals. It seems that the EC-Net with normals makes better predictions near ground-truth boundaries, but also produces additional boundaries away from ground-truth ones, which results in higher Chamfer Distance. In any case, our PB-DGCNN with

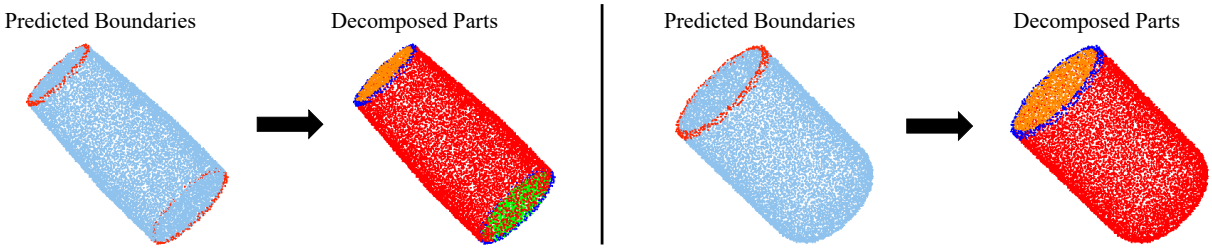

Figure 6: *Examples of watershed (flood-filling) segmentation. In these cases well-defined predicted boundaries between geometric parts, enable their decomposition to individual segments through simple BFS-based flood-filling.*

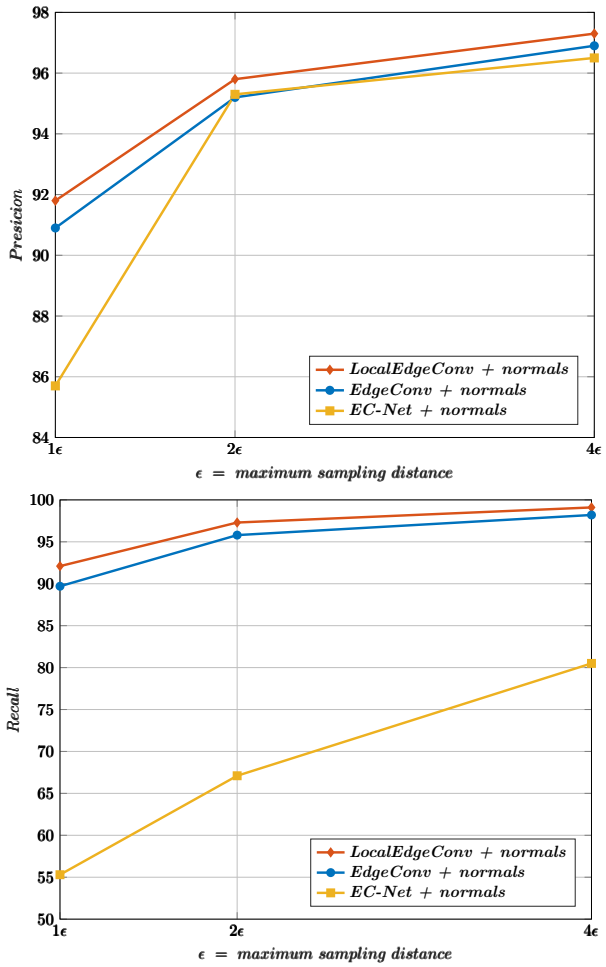

Figure 7: *Boundary detection evaluation wrt. precision and recall for different tolerance levels.*

LocalEdgeConv offers much better performance compared to both versions of EC-Net according to all our evaluation metrics.

Ablation study. In Table 1, we also report the performance of our method under the following variants: (a) "PB-DGCNN w/ EdgeConv" where we use the original EdgeConv layer of DGCNN [WSL*19] as first layer instead of LocalEdgeConv. We include the performance of this variant with and without using normals as input features (b) "PB-DGCNN w/ LocalEdgeConv-curv" where we use the principal curvature directions as tangent vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ to define the local coordinate frame $\mathbf{R}_i$ per point (we note that since the curvature directions are defined up to a sign, the max operator in Eq. 3 is applied to MLPs that also include coordinate transformations based on the opposite principal directions). Curvature is estimated based on the method proposed in [KSNS07]. Finally, we include the performance of of our method "PB-DGCNN w/LocalEdgeConv" with and without normals as input features. Based on the numerical results, we observe that "PB-DGCNN w/LocalEdgeConv w/ normals" has the best performance on average. Its achieved Chamfer Distance is lowest compared to all variants, and the bIoU and F1 score are the highest. Using principal directions did not seem to help the performance of LocalEdgeConv. The LocalEdgeConv w/ normals has consistently better performance compared to using EdgeConv w/ normals according to all metrics. Similarly LocalEdgeConv w/o normals is better than using EdgeConv w/o normals on average. The precision of EdgeConv w/o normals is a bit higher than LocalEdgeConv w/o normals, yet note that its recall is much lower.

Figure 7 shows precision and recall for LocalEdgeConv, EdgeConv and EC-Net (here, we use points and normals as input to all methods). Increasing the tolerance results in increasing the precision for all methods, since more predicted boundaries are classified as correct by increasing the boundary distance tolerance threshold, as expected. Similarly, recall is also increased. Most importantly, our method based on LocalEdgeConv has better behavior than the rest, since it demonstrates both higher precision and recall for all tolerance levels we examined.

Figure 5 provides a visual demonstration for some example point clouds from ABC. We find that the EC-Net boundaries are highly noisy and inconsistent, while ours tend to agree with the ground-truth more.

Application to geometric decomposition. We found that our boundaries can be used for segmentation of several ABC shapes using a simple flood-filling, watershed segmentation approach (Figure 1 and 6). We first construct a K-NN graph ($K = 4$) over the point cloud, then we perform a BFS starting from a random seed point and stopping at predicted boundary points. All visited points result in a segment. Then we start the same procedure by using a random seed point from the rest of the non-visited points. We note, however, that this simple flood-filling approach can fail in cases where small gaps exist in boundaries (Figure 9).

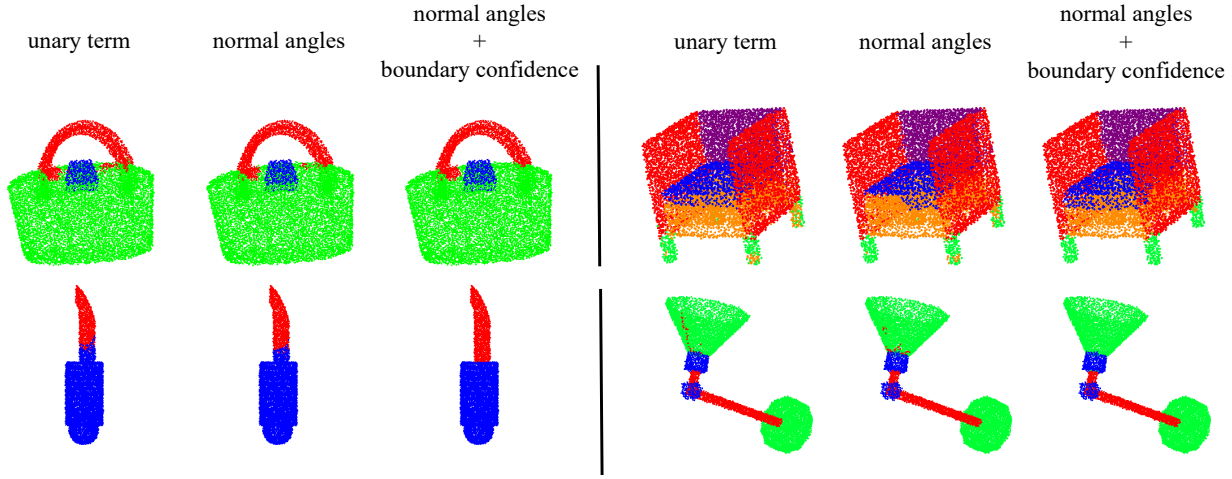

Figure 8: *Visual comparison of semantic segmentation for example PartNet point clouds, using DGCNN alone (unary), a graph-cut formulation with normal angles in the pairwise term, and a graph-cut formulation with a combination of normal angles and boundary probabilities produced by PB-DGCNN in the pairwise term. Boundary confidence can help to diminish small semantic segments, which are falsely predicted from the semantic segmentation model (unary). These are the cases of Bag (top row, left) and Lamp (bottom row, right), where wrong annotated parts are present even after applying the graph cuts method with normal angles as a pairwise term. Moreover, it can further smooth out semantic parts as it is been illustrated in the case of Chair (top row, right) and Knife (bottom row, left).*

Category	Bag	Bed	Bott	Bowl	Chai	Cloc	Dish	Disp	Door	Ear	Fauc	Hat	Key	Knif	Lamp	Lap	Micr	Mug	Frid	Scis	Stor	Tabl	Tras	Vase	**Avg**
Shape IoU																									
Unary only	75.9	25.7	59.1	75.5	50.8	43.9	53.9	84.0	44.0	52.9	55.8	64.3	**62.4**	43.3	43.5	95.9	59.9	88.4	51.6	76.8	52.1	52.9	52.4	80.8	60.2
GC normal diff	76.1	25.9	60.6	81.3	55.0	**44.1**	54.1	85.2	45.4	**53.0**	58.0	65.2	**62.4**	45.6	47.2	**96.0**	60.7	89.5	**52.9**	76.9	55.2	55.7	54.0	82.5	61.8
GC PB-DGCNN	76.1	26.1	61.4	83.2	54.6	43.7	54.0	**86.1**	48.9	52.9	57.5	70.3	**62.4**	47.4	48.5	94.6	**60.9**	**90.4**	51.6	77.2	54.1	56.3	**55.0**	83.5	62.4
GC both	**76.2**	**26.2**	**61.6**	**84.6**	**56.3**	43.7	**54.4**	85.9	**49.3**	52.9	**58.4**	**72.2**	**62.4**	**48.0**	**49.9**	94.6	60.8	88.7	52.6	**78.0**	**55.3**	**57.0**	54.7	**83.8**	**62.8**
Part IoU																									
Unary only	49.9	26.8	39.9	64	40.6	**24.6**	46.2	84.3	32.2	**42.4**	46.1	62.7	**61.1**	39.5	24.1	95.6	**54.4**	81.5	37.7	76.5	43.1	33.4	**45.5**	55.9	50.3
GC normal diff	50.0	**27.0**	39.9	64.2	41.5	**24.6**	46.5	**84.6**	32.7	**42.4**	47.1	63.0	**61.1**	38.5	24.3	**95.7**	52.3	80.4	**38.1**	76.6	**43.3**	**33.6**	42.8	**56.8**	50.3
GC PB-DGCNN	**50.2**	**27.0**	44.3	66.7	41.2	24.0	46.7	**84.6**	32.6	**42.4**	46.7	63.6	**61.1**	**39.7**	**24.4**	93.8	53.9	**82.7**	37.7	76.8	43.2	33.4	43.8	56.6	50.7
GC both	**50.2**	**27.0**	**45.0**	**69.4**	**41.6**	24.0	**47.0**	84.4	**33.1**	**42.4**	**47.3**	**65.7**	**61.1**	38.4	**24.4**	93.9	52.2	80.2	**38.1**	**77.6**	43.2	33.4	42.0	56.7	**50.8**

Table 2: *Point labeling evaluation of fine-grained semantic segmentation on the PartNet dataset (Part IoU, Shape IoU - %). "Unary alone" represents using the per point part probabilities produced by DGCNN, "GC normal diff" represents graph cuts using normal angles as pairwise term, "GC PB-DGCNN" represents graph cuts using our predicted boundary confidences as pairwise term, and "GC both" represents graph cuts using the weighted combination of pairwise terms based on both normal angles and PB-DGCNN.*

4.3. Semantic shape segmentation

We now discuss evaluation on semantic shape segmentation based on the PartNet dataset. Here, we train our network on the PartNet training split for each of its categories, as described in Section 3.2. To take advantage of semantic part labels, here we first use a network that predicts a probability for each part per point. Specifically, we use the DGCNN network for this task [WSL*19], operating on 10K number of point samples per shape. We then incorporate a graph cuts formulation, where the above per-point part probability is used as a unary term, and the output boundary probabilities from our method ("PB-DGCNN w/ LocalEdgeConv w/normals") are used as a pairwise term:

$$E(\mathbf{c}) = \sum_{i \in \mathcal{P}} \psi(c_i) + \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}(i)} \phi(c_i, c_j) \qquad (5)$$

where $\mathbf{c} = \{c_i\}$ are the label assignments we wish to compute by minimizing the above energy, $\mathcal{P}$ is the set of points in a test point cloud, and $\mathcal{N}(i)$ is the neighborhood of each point i formed by its $K = 4$ nearest Euclidean neighbors. The unary term is expressed as follows $\psi(c_i) = -logP(c_i)$, where $P(c_i)$ is the probability distribution over part labels associated with the point i produced by the DGCNN point labeling network. The pairwise term uses the maximum of our PB-DGCNN boundary probabilities for the two points: $\phi(c_i, c_j) = -\lambda \cdot log(\max(b_i, b_j))$ for $c_i \neq c_j$, and 0 otherwise. The weighting parameter λ is adjusted through grid search in the hold-out validation set per shape category. To avoid infinite costs, we add a small $\varepsilon = 10^{-3}$ to the above log expressions.

We also experimented with another pairwise term variant as baseline that considers angles between between point normals: $\phi'(c_i, c_j) = -\lambda' \cdot log(min(\omega_{i,j}/90^o, 1))$, for $c_i \neq c_j$, where $\omega_{i,j}$ is the angle between the point normals. The term results in zero cost for right angles between normals indicating a strong edge. The weighting parameter λ' is adjusted through grid search in the hold-out validation set per shape category. We finally experimented with a combination of using both the above pairwise terms in Eq. 5: $\phi(c_i, c_j) + \phi'(c_i, c_j)$.

We first report point labeling performance in Table 2 based on the standard part IoU and shape IoU metrics in PartNet [MZC*19]. We examine the performance of using DGCNN alone as unary term ("unary alone"), then using graph cuts based on the normal angle baseline described above ("GC normal diff"), graph cuts based on the predicted boundary probabilities of PB-DGCNN ("GC PB-DGCNN"), and finally graph cuts using the summation of pairwise terms from normal angles and PB-DGCNN ("GC both"). We observe small but noticeable average performance increases for both shape IoU and part IoU when using all variants of graph cuts. The best performance is achieved on average (see last row, last column) when combining both pairwise terms. Specifically, we observe an increase of average shape IoU by 2.6% and part IoU 0.5% compared to using the unary term alone. We believe that using both pairwise terms offers the best performance because our boundary probabilities are more fuzzy in PartNet - we note that the training boundary data were also slightly fuzzy in PartNet (see Figure 4) in the first place. Using normal angles further sharpens our probabilistic boundaries. Nevertheless, the metrics seem improved with the use of our probabilistic boundaries in the pairwise term alone. We note that graph cuts is executed in a deterministic manner (i.e., there is no variance in the above increases given a fixed unary term). The performance increase is not dramatic: this is not surprising, since refining boundaries changes relatively few point labels near boundaries.

Table 3 reports our evaluation metrics in terms of boundary quality. We note that compared to the ABC dataset, the evaluation of boundaries here is less reliable. In contrast to ABC, where the ground-truth boundaries were parametric curves and were highly accurate, the ground-truth boundaries in PartNet are marked approximately using the heuristic search described in Section 3.2. We report the performance of our best variant of graph cuts (using both terms), and also the unary term alone. We observe that graph cuts result in boundaries that are more consistent with ground-truth. In particular, we observe an improvement of 2.8% for bIoU, and 2.9% for F1-score on average. We note that although the recall is lower, the precision is significantly much higher. Figures 1 and 8 show semantic segmentation results for a few examples from PartNet.

5. Limitations and Conclusion

We presented a method for detecting probabilistic boundaries in point clouds based on a neural network. Our evaluation showed that our boundaries are closer to ground-truth in geometric decomposition tasks, and also improve the quality of cuts in semantic segmentation tasks. Our method also has limitations that could inspire future research. First, our method currently extracts probabilities of part boundaries over points. Sometimes these probabilities seem too low (Figure 10), resulting in sparsely labeled boundary points, which makes it harder to extract a continuous boundary curve. It would be interesting to investigate robust fitting of parametric curves or lines to probabilistic boundaries to localize them more accurately. This could in turn be combined with neural patch fitting [SLK*20], and also result in geometric decomposition of point clouds to primitives with more accurately trimmed boundaries. For semantic segmentation, jointly optimizing the unary and pairwise term with the rest of the network in an end-to-end manner could further improve results. Finally, it would be interesting to extend our method to handle polygon mesh segmentations based on our detected boundaries.

6. Acknowledgements.

This research is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 739578 (RISE – Call: H2020-WIDESPREAD-01-2016-2017-TeamingPhase2), the Government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development, and NSF (CHS-1617333). Our experiments were performed in the UMass GPU cluster obtained under the Collaborative Fund managed by the Massachusetts Technology Collaborative. We especially thank Gopal Sharma for assisting with the filtering and preparation of the ABC training data. We thank the anonymous reviewers for their feedback.

References

[AML18] ATZMON M., MARON H., LIPMAN Y.: Point convolutional neural networks by extension operators. *ACM Trans. Graph. 37*, 4 (2018). 2

[BAM*05] BELYAEV A., ANOSHKINA E., MARTIN R., BEZ H., SABIN M.: Detection of surface creases in range data. *Mathematics of surfaces XI : 11th IMA International Conference, Springer, 50-61 (2005)* (10 2005). 2

[BBL*17] BRONSTEIN M. M., BRUNA J., LECUN Y., SZLAM A., VANDERGHEYNST P.: Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine 34* (2017), 18–42. 2

[BS16] BOGOSLAVSKYI I., STACHNISS C.: Fast range image-based segmentation of sparse 3d laser scans for online operation. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)* (2016). 2

[BST15a] BERTASIUS G., SHI J., TORRESANI L.: Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR* (2015). 1, 2

[BST15b] BERTASIUS G., SHI J., TORRESANI L.: High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. *IEEE International Conference on Computer Vision (ICCV)* (2015), 504–512. 1, 2

[BST16] BERTASIUS G., SHI J., TORRESANI L.: Semantic segmentation with boundary neural fields. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). 1, 2

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (2009). 7

[CGL*08] COLE F., GOLOVINSKIY A., LIMPAECHER A., BARROS H. S., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S.: Where do people draw lines? *ACM Trans. Graph. 27*, 3 (2008). 5

[CJXH17] CAO Y.-P., JU T., XU J., HU S.-M.: Extracting sharp features from rgb-d images. *Computer Graphics Forum 36*, 8 (2017). 2

[CKUF17] CASTREJON L., KUNDU K., URTASUN R., FIDLER S.: Annotating object instances with a polygon-rnn. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 4485–4493. 1, 2

[CTC13] CHOI C., TREVOR A. J. B., CHRISTENSEN H. I.: Rgb-d edge detection and edge-based registration. In *IROS* (2013). 2

[DGY*19] DENG B., GENOVA K., YAZDANI S., BOUAZIZ S., HINTON G., TAGLIASACCHI A.: Cvxnet: Learnable convex decomposition, 2019. arXiv:1909.05736. 2

Category	Bag	Bed	Bott	Bowl	Chai	Cloc	Dish	Disp	Door	Ear	Fauc	Hat	Key	Knif	Lamp	Lap	Micr	Mug	Frid	Scis	Stor	Tabl	Tras	Vase	**Avg**
Chamfer Distance																									
Unary alone	**6.3**	**3.1**	**6.9**	38.8	4.0	**5.4**	6.6	7.8	37.3	**9.9**	6.0	4.1	**2.3**	9.4	7.2	**1.0**	**3.3**	3.6	**3.9**	9.4	1.5	3.2	**3.8**	**10.9**	8.2
GC both	6.4	**3.1**	7.1	**37.0**	**3.7**	**5.4**	**6.2**	**7.7**	**36.4**	**9.9**	**5.4**	**3.1**	**2.3**	**7.3**	**5.9**	1.3	3.5	**3.2**	**3.9**	**8.7**	**1.3**	**2.7**	5.9	11.0	**7.9**
Boundary IoU																									
Unary alone	**61.5**	72.3	48.0	56.0	67.0	59.0	64.0	73.5	54.4	**49.1**	54.4	77.3	**76.5**	38.7	57.1	90.5	**79.1**	66.7	69.1	44.7	88.4	75.9	**77.6**	73.4	65.6
GC both	60.9	**73.0**	**48.4**	**60.7**	**70.2**	**63.9**	**66.0**	**74.9**	**57.9**	49.0	**63.2**	**82.1**	76.1	**47.0**	**67.0**	**90.9**	76.4	**75.3**	**69.7**	**50.4**	**90.0**	**79.2**	74.8	**75.4**	**68.4**
Precision																									
Unary alone	74.3	70.5	52.3	58.7	65.3	57.8	67.1	72.1	56.9	56.6	51.8	78.4	81.8	34.8	56.2	91.2	81.4	75.0	76.4	46.2	88.6	78.0	78.4	72.8	67.6
GC both	**77.6**	**73.7**	**60.4**	**68.0**	**80.2**	**73.9**	**75.7**	**81.4**	**65.7**	**57.6**	**71.2**	**89.2**	**82.1**	**54.7**	**81.9**	**93.8**	**81.7**	**87.3**	**80.3**	**54.8**	**94.4**	**90.1**	**89.6**	**78.8**	**76.8**
Recall																									
Unary alone	**57.9**	**79.5**	**48.9**	**62.8**	**74.7**	**68.0**	**68.0**	**79.1**	**56.1**	**47.8**	**61.8**	**78.9**	**75.9**	**51.6**	**72.5**	**91.0**	**78.0**	65.1	**66.9**	45.3	**90.0**	**81.1**	**79.9**	**79.9**	**69.2**
GC both	55.9	76.8	44.8	62.1	67.1	63.4	63.2	72.7	55.9	46.8	60.2	78.0	75.3	46.2	63.9	90.0	73.3	**70.6**	65.4	**48.9**	87.5	76.2	68.9	76.4	66.2
F_1-score																									
Unary alone	**65.1**	74.7	50.5	60.7	69.7	62.5	67.5	75.4	56.5	**51.8**	56.4	78.7	**78.8**	41.5	63.3	91.1	**79.7**	69.7	71.3	45.7	89.3	79.5	**79.1**	76.2	68.1
GC both	65.0	**75.2**	**51.4**	**64.9**	**73.1**	**68.3**	**68.9**	**76.8**	**60.4**	51.6	**65.2**	**83.2**	78.5	**50.1**	**71.8**	**91.9**	77.2	**78.0**	**72.1**	**51.7**	**90.8**	**82.5**	77.9	**77.6**	**71.0**

Table 3: *Evaluation of fine-grained semantic segmentation boundaries (Chamfer distance, Boundary IoU, Precision, Recall, F_1 score - %) on the PartNet dataset. "Unary alone" represents using the per point part probabilities produced by DGCNN, "GC both" represents a weighted combination of pairwise terms based on normal angles and PB-DGCNN.*

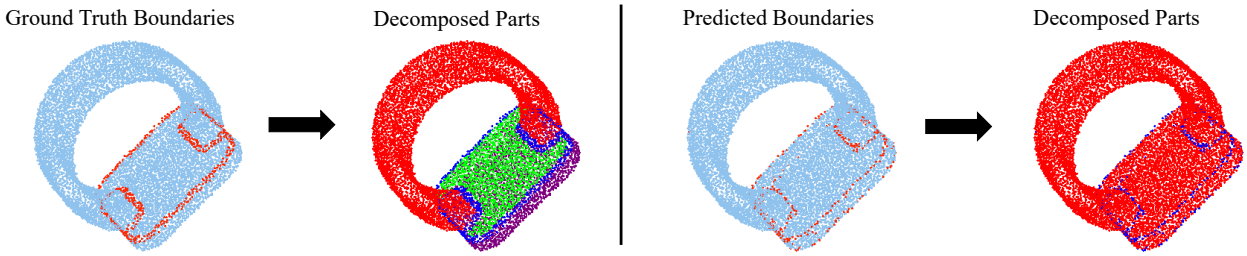

Figure 9: *The figure on the left depicts part decomposition of the point cloud from ground truth boundaries, with BFS flood-filling. It successfully segments the point cloud into three parts. This is not the case on the right figure, where the predicted boundaries fail to enclose points into separate segments, which results to only one part after the watershed segmentation procedure.*

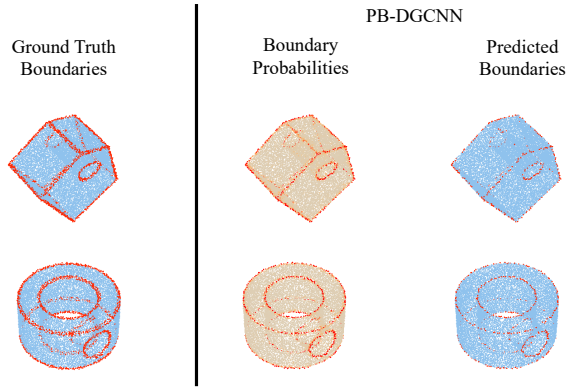

Figure 10: *Predicted boundary confidences (middle column) is sometimes low resulting in sparsely labeled boundary points (right column).*

[EDP*11] EBEIDA M. S., DAVIDSON A. A., PATNEY A., KNUPP P. M., MITCHELL S. A., OWENS J. D.: Efficient maximal poisson-disk sampling. *ACM Trans. Graph.* (July 2011). 4

[GF08] GOLOVINSKIY A., FUNKHOUSER T.: Randomized Cuts for 3D Mesh Analysis. *ACM Trans. on Graph. 27*, 5 (2008). 2

[GF09] GOLOVINSKIY A., FUNKHOUSER T.: Min-cut based segmentation of point clouds. In *IEEE Workshop on Search in 3D and Video (S3DV) at ICCV* (Sept. 2009). 2

[GWL18] GROH F., WIESCHOLLEK P., LENSCH H. P. A.: Flex-convolution (million-scale point-cloud learning beyond grid-worlds). In *Asian Conference on Computer Vision (ACCV)* (2018). 2

[HKC*17] HUANG H., KALOGERAKIS E., CHAUDHURI S., CEYLAN D., KIM V. G., YUMER E.: Learning local shape descriptors from part correspondences with multiview convolutional networks. *ACM Transactions on Graphics 37*, 1 (2017). 2

[HLC*18] HOU Q., LIU J.-J., CHENG M.-M., BORJI A., TORR P. H. S.: Three birds one stone: A general architecture for salient object segmentation, edge detection and skeleton extraction, 2018. arXiv:1803.09860. 1, 2

[HRV*18] HERMOSILLA P., RITSCHEL T., VÁZQUEZ P.-P., VINACUA A., ROPINSKI T.: Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Trans. Graph. 37*, 6 (2018). 2

[HTY18] HUA B., TRAN M., YEUNG S.: Point-wise convolutional neural network. *Computer Vision and Pattern Recognition (CVPR)* (2018). 2

[HWG*13] HUANG H., WU S., GONG M., COHEN-OR D., ASCHER U., ZHANG H. R.: Edge-aware point set resampling. *ACM Trans. Graph.* (2013). 2

[JZL*19] JIANG L., ZHAO H., LIU S., SHEN X., FU C.-W., JIA J.: Hierarchical point-edge interaction network for point cloud semantic segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). 2

[KAMC17] KALOGERAKIS E., AVERKIOU M., MAJI S., CHAUDHURI S.: 3D shape segmentation with projective convolutional networks. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)* (2017). 2

[KB14] KINGMA D., BA J.: Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014). 5

[KHS10] KALOGERAKIS E., HERTZMANN A., SINGH K.: Learning 3D Mesh Segmentation and Labeling. *ACM Transactions on Graphics 29*, 3 (2010). 2

[KL17] KLOKOV R., LEMPITSKY V.: Escape from cells: Deep Kd-Networks for the recognition of 3D point cloud models. *ICCV* (2017). 2

[KMFF13] KARPATHY A., MILLER S., FEI-FEI L.: Object discovery in 3d scenes via shape analysis. In *International Conference on Robotics and Automation (ICRA)* (2013). 2

[KMJ*19] KOCH S., MATVEEV A., JIANG Z., WILLIAMS F., ARTEMOV A., BURNAEV E., ALEXA M., ZORIN D., PANOZZO D.: Abc: A big cad model dataset for geometric deep learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (2019). 1, 4, 7

[KNS*09] KALOGERAKIS E., NOWROUZEZAHRAI D., SIMARI P., MCCRAE J., HERTZMANN A., SINGH K.: Data-driven curvature for real-time line drawing of dynamic scene. *ACM Transactions on Graphics 28*, 1 (2009). 2

[Kok15] KOKKINOS I.: Pushing the boundaries of boundary detection using deep learning. In *ICLR 2016* (2015). 1, 2

[KSNS07] KALOGERAKIS E., SIMARI P., NOWROUZEZAHRAI D., SINGH K.: Robust statistical estimation of curvature on discretized surfaces. In *Proc. SGP* (2007). 8

[KST09] KOLOMENKIN M., SHIMSHONI I., TAL A.: On edge detection on surfaces. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009). 2

[KT03] KATZ S., TAL A.: Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts. *ACM Trans. Graphics* (2003). 2

[KT18] KISNER H., THOMAS U.: Segmentation of 3d point clouds using a new spectral clustering algorithm without a-priori knowledge. In *VISIGRAPP* (2018). 2

[KZH19] KOMARICHEV A., ZHONG Z., HUA J.: A-CNN: annularly convolutional neural networks on point clouds. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 7413–7422. 2

[LBS*18] LI Y., BU R., SUN M., WU W., DI X., CHEN B.: Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems 31* (2018). 2

[LCF*18] LIU Y., CHENG M.-M., FAN D.-P., ZHANG L., BIAN J., TAO D.: Semantic edge detection with diverse deep supervision, 2018. arXiv:1804.02864. 1, 2

[LCH*17] LIU Y., CHENG M., HU X., WANG K., BAI X.: Richer convolutional features for edge detection. *IEEE conference on computer vision and pattern recognition (CVPR)* (2017), 3000–3009. 1, 2

[LCL18] LI J., CHEN B., LEE G.: So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (06 2018). 2

[LFM*19] LIU Y., FAN B., MENG G., LU J., XIANG S., PAN C.: Densepoint: Learning densely contextual representation for efficient point cloud processing. In *IEEE International Conference on Computer Vision (ICCV)* (2019), pp. 5239–5248. 2

[LFXP19] LIU Y., FAN B., XIANG S., PAN C.: Relation-shape convolutional neural network for point cloud analysis. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 8895–8904. 2

[LKM19] LE E., KOKKINOS I., MITRA N. J.: Going deeper with point networks. *CoRR abs/1907.00960* (2019). 2

[LKV*18] LINSLEY D., KIM J., VEERABADRAN V., WINDOLF C., SERRE T.: Learning long-range spatial dependencies with horizontal gated recurrent units. In *Advances in Neural Information Processing Systems*. 2018. 1, 2

[LLD19] LE T., LI Y., DUAN Y.: Red-net: A recursive encoder-decoder network for edge detection, 2019. arXiv:1912.02914. 1, 2

[LMTG19] LI G., MÜLLER M., THABET A. K., GHANEM B.: Deepgcns: Can gcns go as deep as cnns? *The IEEE International Conference on Computer Vision (ICCV)* (2019). 2

[LNHK20] LIU D., NABAIL M., HERTZMANN A., KALOGERAKIS E.: Neural contours: Learning to draw lines from 3d shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020). 7

[LSD*19] LI L., SUNG M., DUBROVINA A., YI L., GUIBAS L.: Supervised fitting of geometric primitives to 3d point clouds. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 2647–2655. 2

[LTLH19] LIU Z., TANG H., LIN Y., HAN S.: Point-voxel CNN for efficient 3d deep learning. *Annual Conference on Neural Information Processing Systems (NeurIPS)* (2019). 2

[LWC*11] LI Y., WU X., CHRYSATHOU Y., SHARF A., COHEN-OR D., MITRA N. J.: Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph. 30*, 4 (2011). 2

[MPTAG17] MANINIS K., PONT-TUSET J., ARBELÁEZ P., GOOL L. V.: Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2017). 1, 2

[MZC*19] MO K., ZHU S., CHANG A. X., YI L., TRIPATHI S., GUIBAS L. J., SU H.: PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 1, 5, 10

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph. 23*, 3 (2004). 2

[PK20] PETROV D., KALOGERAKIS E.: Cross-shape graph convolutional networks, 2020. arXiv:2003.09053. 2

[PNH*19] PHAM Q.-H., NGUYEN D. T., HUA B.-S., ROIG G., YEUNG S.-K.: JSIS3D: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019). 2

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2017), NIPS'17, Curran Associates Inc., p. 5105–5114. 2

[RBM*07] RADU BOGDAN RUSU, BLODOW N., MARTON Z., SOOS A., BEETZ M.: Towards 3d object maps for autonomous household robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2007). 2

[RUG17] RIEGLER G., ULUSOY A. O., GEIGER A.: Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017). 2

[RWS*18] RETHAGE D., WALD J., STURM J., NAVAB N., TOMBARI F.: Fully-convolutional point networks for large-scale point clouds. In *European Conference on Computer Vision (ECCV)* (2018). 2

[SGS19] SRIVASTAVA N., GOH H., SALAKHUTDINOV R.: Geometric capsule autoencoders for 3d point clouds, 2019. arXiv:1912.03310. 2

[SJS*18] SU H., JAMPANI V., SUN D., MAJI S., KALOGERAKIS E., YANG M.-H., KAUTZ J.: SPLATNet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018). 2

[SJW*11] SUNKEL M., JANSEN S., WAND M., EISEMANN E., SEIDEL H.-P.: Learning line features in 3d geometry. *Computer Graphics Forum 30*, 2 (2011). 2

[SKM19] SHARMA G., KALOGERAKIS E., MAJI S.: Learning point embeddings from shape repositories for few-shot segmentation, 2019. arXiv:1910.01269. 2

[SLK*20] SHARMA G., LIU D., KALOGERAKIS E., MAJI S., CHAUDHURI S., MĚCH R.: Parsenet: A parametric surface fitting network for 3d point clouds, 2020. arXiv:2003.12181. 2, 10

[SWK07] SCHNABEL R., WAHL R., KLEIN R.: Efficient ransac for point-cloud shape detection. *Computer Graphics Forum 26*, 2 (June 2007), 214–226. 2

[SWL19] SHI S., WANG X., LI H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 2

[SWS*14] STEIN S. C., WÖRGÖTTER F., SCHOELER M., PAPON J., KULVICIUS T.: Convexity based object partitioning for robot applications. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014). 2

[SWW*15] SHEN W., WANG X., WANG Y., BAI X., ZHANG Z.: Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR* (2015). 1, 2

[TQD*19] THOMAS H., QI C. R., DESCHAUD J., MARCOTEGUI B., GOULETTE F., GUIBAS L. J.: Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision* (2019), 6411–6420. 2

[vKFK*14] VAN KAICK O., FISH N., KLEIMAN Y., ASAFI S., COHEN-OR D.: Shape segmentation by approximate convexity analysis. *ACM Trans. on Graphics to appear* (2014). 2

[WHH*19] WANG L., HUANG Y., HOU Y., ZHANG S., SHAN J.: Graph attention convolution for point cloud semantic segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019). 2

[WLG*17] WANG P.-S., LIU Y., GUO Y.-X., SUN C.-Y., TONG X.: O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM TOG 36*, 4 (2017). 2

[WQL19] WU W., QI Z., LI F.: Pointconv: Deep convolutional networks on 3d point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 9621–9630. 2

[WSL*19] WANG Y., SUN Y., LIU Z., SARMA S. E., BRONSTEIN M. M., SOLOMON J. M.: Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph. 38*, 5 (2019). 1, 2, 3, 4, 8, 9

[WSLT18] WANG P.-S., SUN C.-Y., LIU Y., TONG X.: Adaptive o-cnn: A patch-based deep representation of 3d shapes. *ACM Trans. Graph. 37*, 6 (2018). 2

[WZLH18] WANG Y., ZHAO X., LI Y., HUANG K.: Deep crisp boundaries: From boundaries to higher-level tasks. *IEEE Transactions on Image Processing 28*, 3 (2018), 1285–1298. 1, 2

[XFX*18] XU Y., FAN T., XU M., ZENG L., QIAO Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 87–102. 2

[XSyW*19] XU Q., SUN X., YING WU C., WANG P., NEUMANN U.: Grid-gcn for fast and scalable point cloud learning, 2019. arXiv:1912.02984. 2

[XT17] XIE S., TU Z.: Holistically-nested edge detection. *Int. J. Comput. Vision 125*, 1–3 (2017), 3–18. 1, 2

[YLF*18] YU L., LI X., FU C.-W., COHEN-OR D., HENG P.-A.: Ec-net: an edge-aware point set consolidation network. In *ECCV* (2018). 1, 2, 7

[YSGG17] YI L., SU H., GUO X., GUIBAS L.: Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). 2

[ZYY*17] ZOU C., YUMER E., YANG J., CEYLAN D., HOIEM D.: 3d-prnn: Generating shape primitives with recurrent neural networks. *Proceedings of the IEEE International Conference on Computer Vision* (2017), 900–909. 2

[ZZ19] ZHANG L., ZHU Z.: Unsupervised feature learning for point cloud by contrasting and clustering with graph convolutional neural network. 2