Computers & Graphics 97 (2021) 208–216

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Section on CAD & Graphics 2021

LGCPNet : Local-global combined point-based network for shape segmentation

Boliang Guan[a], Hanhui Li[b], Fan Zhou[a], Shujin Lin[c], Ruomei Wang[a,*]

[a] School of Computer Science and Engineering, National Engineering Research Center of Digital Life, Sun Yat-sen University, Guangzhou 510006, China
[b] School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore
[c] School of Communication and Design, Sun Yat-sen University, Guangzhou 510006, China

ARTICLE INFO

Article history:
Received 7 February 2021
Revised 5 April 2021
Accepted 13 April 2021
Available online 24 April 2021

Keywords:
Shape segmentation
Pointwise labelling
Mesh processing
Deep learning

ABSTRACT

Segmenting 3D shapes represented by meshes remains a challenging problem, due to the irregularity and complexity of meshes. Point cloud, on the other hand, can be considered as the simplest no-frills approximation for meshes. Therefore, in this paper, we regard the shape segmentation problem as a point labeling task: Given a shape, we first transform it into points encoding barycenters and normal vectors of faces. Then we construct a Barycentric Dual Graph (BDG) on the transformed points, and propose a Barycentric Dual Graph Edge Convolution (BDGEC) to extract features from the graph. Based on the BDGEC, we further propose a novel point-based deep neural network (DNN) named local-global combined point-based network (LGCPNet). Our LGCPNet consists of three modules, of which the Local Module and Global Module capture local and global features respectively, while the Fusion Module uses a gate mechanism to aggregate local features and global features, and obtain the point labeling result. Comprehensive experimental results on various datasets demonstrate that the proposed network inherits the merits of point-based DNNs and achieves the state-of-the-art performance.

1. Introduction

Shape segmentation, aiming to divide a 3D shape into meaningful parts, is one of the important steps for not only capturing shape properties and structural characteristics, but also model analysis and understanding [1,2]. Earlier techniques for shape segmentation are unsupervised, which focus on manually engineering geometric features [2]. These approaches always achieve partitioned results by employing optimization strategies on predefined hand-crafted geometric features such as curvature tensor [3], normalized cut [4], surface fitting [5], spectral clustering [6], and convexity [7]. Unsupervised methods are simple and able to achieve good performance in the case that the number of shapes are small. Nevertheless, with the scale of 3D data enlarging, these traditional unsupervised approaches can not satisfy the needs of shape analysis in universality and efficiency.

Due to the above limitation of traditional unsupervised approaches, data-driven approaches become more and more applicable and useful to shape analysis with the availability of 3D data increasing [1,2]. In contrast to traditional unsupervised approaches, data-driven methods mainly analyze sets of shapes in a supervised manner by learning from given shapes, which effectively reasons about their properties and relationships without relying on predefined hand-crafted rules [1]. As a result, these supervised learning methods can discover geometric and structural patterns among shapes, and facilitate high-level shape understanding.

With the outstanding performance of deep neural networks (DNNs) in image segmentation, data-driven methods recently concentrate on how to improve the performance of shape segmentation by DNNs. Researchers leveraged DNNs as an optimization strategy to cluster hand-crafted geometric features, which enhance the performance and functionality in shape segmentation [11–14]. To better preserve original geometric information of shapes, some approaches [8,9,15] utilize DNNs as an end-to-end strategy to extract features of shapes and cluster them straightly. Thereinto, the edge-based method [8], shown in Fig. 1 (a), partitions shapes via a convolutional neural network (CNN) designed for triangular meshes. Tailored to the mesh structure, it performances well in segmenting boundaries, but it needs to generate extra edge-level semantic labels. The view-based methods [9,15], shown in Fig. 1 (b), project 3d shapes into multiple 2d views to employ CNNs, and outputs of each view are mapped onto the 3D object surfaces. These methods fit datasets with face labels

* Corresponding author.
E-mail addresses: uanbl3@mail2.sysu.edu.cn (B. Guan), hanhui.li@ntu.edu.sg (H. Li), isszf@mail.sysu.edu.cn (F. Zhou), linshjin@mail.sysu.edu.cn (S. Lin), isswrm@mail.sysu.edu.cn (R. Wang).

https://doi.org/10.1016/j.cag.2021.04.028
0097-8493/

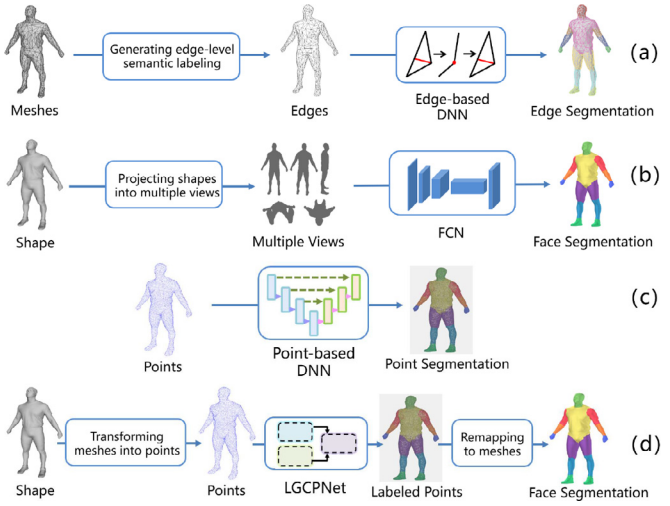

Fig. 1. The illustration of various methods for shape segmentation. (a) MeshCNN is tailored to the mesh structure via the edge-based pretreating and convolution but achieves edge-level semantic labeling. [8], (b) ShapePFCN renders the shape to shaded images in order to employ the CNN [9], (c) PointNet++ is exploited to analyze discrete point clouds [10], and (d) LGCPNet(ours). Compared with other methods, LGCPNet can attain face-level semantic labeling with an easy transformation of shapes.

well and achieve outstanding segmentation results, but vast views make view-based methods consume more memory and time. As a commonly-used 3D data format, point cloud based representation defines a no-frills approximation for an underlying three-dimensional shape [8]. Point-based DNNs directly take discrete points as input and learn robust features, can obtain results efficiently and effectively [10,16], as shown in Fig. 1 (c).

In this paper, we design a novel local-global combined point-based network (LGCPNet) shown in Fig. 1 (d), for mesh segmentation as point labeling. To convert the shape segmentation problem into point labeling, the given shape is firstly processed into points consisting of the barycenter and the normal vector of each face of the shape. For feature extraction, we propose the Barycentric Dual Graph Edge Convolution (BDGEC) to extract edge features by operating over the nodes of Barycentric Dual Graph (BDG) intrinsic for the given shape. Based on BDG, our LGCPNet contains Local Module and Global Module to capture local and global features respectively, and the Fusion Module is designed to merge local features and global features and obtain the point labeling result. Finally, the shape segmentation result will be achieved through the point labeling result.

By employing the proposed BDGEC and architecture with Local, Global and Fusion Module, the proposed mesh segmentation method inherits the merits of point-based DNNs and achieves comparable mesh segmentation performance with state-of-the-art techniques on various challenging datasets. The key contributions of our work are as follows:

- A simple and effective shape segmentation method which treats mesh segmentation as point labeling by converting each face into its barycenter and normal vector per shape with the spatial information of given shape preserving.
- An intuitive and efficient operation, called Barycentric Dual Graph Edge Convolution (BDGEC), extracts edge features by operating over the nodes of Barycentric Dual Graph (BDG) intrinsic for the given shape.
- A novel LGCPNet is proposed for mesh segmentation, which can capture local and global features of shape efficiently and obtain satisfying segmentation performance.

The rest of the paper is organized as follows: Section 2 discusses related work, and we describe the proposed framework in Section 3. Experimental results are presented in Section 4 and the last section concludes the paper.

2. Related work

Aiming to improve the precision of segmentation efficiently, shape segmentation has experienced the development. In this section, unsupervised methods for shape segmentation are first briefly introduced for literature comprehensiveness. Then we mainly focus on DNNs methods for shape segmentation and point-based DNNs methods for point cloud segmentation.

2.1. Unsupervised segmentation

In some early studies, shape segmentation is implemented by easy, yet effective ideas in an unsupervised manner. Most approaches are based on simple geometric features including curvature tensor [3], spectral clustering [6], Student-t mixture model [17], convexity [7], surface fitting [5], normalized cut [4]. Then, to tackle large variations in terms of shape and topology, a large body of co-segmentation approaches [18–22] are proposed to investigate consistent segmentation of a collection of shapes. Both singe- and co-segmentation methods rely on hand-crafted geometric features in an unsupervised manner, so that their performance in the final segmentation may be worsened by the large variations between different shapes and the generality of the segmentation method is also weak.

2.2. Supervised segmentation

Supervised methods introduced here mainly focus on DNNs as their outstanding performance in various fields. In the SFCN architecture [11], similar to convolution and pooling operations used on image segmentation, three-dimensional shapes were firstly represented as three low-level geometric features as input. Wang et al. [12] trained the convolutional neural networks (CNNs) by using a large pool of classical geometric features. Similarly, Zhou et al. [14] employed a large variety of shape-based mesh features, which are calculated with face areas and different scales in consideration. In SyncSpecCNN [13], a shape graph equipped with vertex functions by kernels is used to predict a per-vertex label. Nevertheless, the features of these methods were still hand-crafted but not intrinsic. Instead of inputting hand-crafted features, some approaches previously projected three-dimensional shapes to different two dimensional views. Kalogerakis et al. [9] proposed a multi-view image-based Fully Convolutional Networks (FCNs) and surface-based Conditional Random Fields (CRFs) to yield coherent segmentations. Le et al. [15] reported the method to combine the multi-view convolutional neural networks (CNNs) and a two-layer long short term memory (LSTM). Whereas, these view-based DNNs need more viewpoints to maximally cover the shape surface. Later, by leveraging their intrinsic geodesic connections, Hanocka et al. [8] proposed MeshCNN to combine specialized convolution and pooling layers that operate on the mesh edges. Such convolutions are applied on edges, so specific pretreatment for edges is necessary.

2.3. Point-based DNNs

Recently, using neural networks for point cloud shape analysis has been focused. PointNet [23] was presented as the first deep neural network that directly takes the discrete points as input for many tasks of point clouds. In its follow-up work, PointNet++ [10],

an improved version of PointNet, exploited a hierarchical architecture to capture local structures better. Similar to the PointNet architecture, Kd-networks [24] also worked directly with points, while performs and shares parameters of multiplicative transformations according to the subdivisions of the point clouds imposed onto them by kd-trees. Atzmon et al. [25] defined an efficient convolution operator to makes the method invariant to the order of points by mapping the point clouds functions into volumetric functions. PointCNN [26] learned an x-transformation from the input points to permute unordered points into a latent potentially canonical order for feature learning. Recently, Wang et al. [27] took into account local point neighborhood information by dense sampling, and carried out dynamic updates driven by forecasting the similarity distance between points in feature space. Based on the architecture of PointNet++, PointWeb [28] applied an impact map carrying element-wise impact between point pairs in each local region to enhance local neighborhood features. Later, PointConv [29] learned the weight functions with multi-layer perceptron networks and density functions through kernel density estimation. Point-based DNNs significantly reduce the model space and computational complexity, but the point cloud structure sampled from the origin meshes may lack the intrinsic structure of the meshes.

3. Proposed method

From the literature review we can see the researchers applied DNNs technologies to the shape segmentation in different way to obtain perfect performance. Specially, segmentation of point-based DNNs, which directly take the discrete points as input, learn spatial information from point sets more efficiently to achieve the shape segmentation. In this paper, we design a novel local-global combined point-based network (LGCPNet) for mesh segmentation based on point labeling, shown in Fig. 1. The first step is to represent the mesh as a set of points that can be input into the network for processing. We call this is the conversion of mesh to point. In LGCPNet, we design Local Module and Global Module to capture local and global features respectively, and the Fusion Module is designed to concatenate two types of features to obtain the point labeling result. In this section, we will describe the proposed method in details.

3.1. Conversion of mesh to point

The main work in this part is the conversion of mesh to point. We need to input the mesh into the network for processing by the form of points. Both the barycenter and vertices of a mesh are the potential representations for shape transformation. Referred in [30], barycentric coordinates, which encode the location of a point with respect to a planar polygon (or polyhedron in 3D), are intrinsic(insensitive to isometric deformations) and fast to compute. As illustrated in Fig. 3, barycenters are more well-distributed and have a greater number than vertices for a shape, which are desirable to generalize original shapes to improve the performance of neural network. Moreover, each barycenter has the only one label since datasets provide ground truth segmentation per face, while each vertex is related with some edges, such as v_2 is related with five edges and it will has more than one label. As a result, we represent a face by the barycenter of each face. Besides, the normal vector of each surface not only uniquely defines the grid in Euclidean space, but also greatly expresses the original geometric information of the shape. Therefore, the normal vector is also used as the characteristic data of mesh.

As a result, given a mesh $\mathcal{M}$, we donate the barycenter and the normal vector of each face to represent it as 3D coordinates in the Euclidean space. Namely, a given shape can be defined as $\mathcal{M} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\} \subset \mathbb{R}^6$, where $\mathbf{p}_i, i \in \{1, \ldots, N\}$ is a 6-dimensional coordinate representing a barycenter point and a surface normal, N is the number of faces.

According to the mentioned method, we can transform mesh data into point set. Ideally, the number of faces from the given shape is in an appropriate range and all faces are involved in the transformed point set. But, in some cases, the transformed point set is too massive to handle by networks straightly. Therefore, we address a strategy to prepare the input data by slicing them into segments. To cluster points from neighbor faces as far as possible, the transformed point set will be regarded as the Barycentric Dual Graph (BDG) akin to [31] and sorted based on the Breadth-first search(BFS) [32]. Our data slicing strategy not only obviously augments the input data, but also effectively reduces the computational expense and the memory usage of GPU.

Given an 6-dimensional point set with N points, denoted by $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\} \subset \mathbb{R}^6$, it is sliced by Eq. (1):

$$\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\} \Rightarrow \begin{cases} S_1 = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_j\} \\ S_2 = \{\mathbf{p}_{j+1}, \mathbf{p}_{j+2}, \ldots, \mathbf{p}_{j\times 2}\} \\ \vdots \\ S_{\lfloor N/j \rfloor} = \{\mathbf{p}_{j\times(\lfloor N/j \rfloor - 1)+1}, \mathbf{p}_{j\times(\lfloor N/j \rfloor - 1)+2}, \ldots, \mathbf{p}_{j\times \lfloor N/j \rfloor}\} \\ S_{\lceil N/j \rceil} = \{\mathbf{p}_{j\times \lfloor N/j \rfloor + 1}, \mathbf{p}_{j\times \lfloor N/j \rfloor + 2}, \ldots, \mathbf{p}_N, \ldots, \mathbf{p}_m, \mathbf{p}_n\} \end{cases}, \tag{1}$$

where j is the number of elements in each input data slice. Referred to PointNet [23], we set the value of j 2048. Slice $S_{\lceil N/j \rceil}$ should be complemented by randomly sampling once $\lceil N/j \rceil \neq \lfloor N/j \rfloor$. After data slicing, for each point $\mathbf{p}_i, i \in \{1, \ldots, N\}$ in one slice, its neighbor points are firstly fed into edge set $\mathcal{E}$ in order based on the Barycentric Dual Graph (BDG). If the number of neighbor points is less than three yet at least one, it will be added up to three by randomly chose the existing neighbor points.

3.2. Barycentric dual graph edge convolution

After transforming meshes into points, it is a key to construct the graph of which nodes represent points. Most of classic point-based DNNs [10,27–29] use k-nearest neighbor (KNN) algorithm to compute a graph $\mathcal{G} = (V, E)$ representing local point cloud structure, where $V = \{1, \ldots, n\}$ and $E \subseteq V \times V$ are the nodes and edges, respectively. For each point, it is Based on the constructed graph, edge features are extracted via a nonlinear function with a set of learnable parameters and the aggregation operation (e.g., *mean* or *max*) is applied on the edge features associated with all the edges emanating from each node, as shown in Fig. 4 (a). This operation has a good coverage of the entire point set, but its computational complexity is $\mathcal{O}(N^2)$, which make it inappropriate for large scale point sets.

Refered in [31], for a watertight shape, barycentric duals guarantee that the dual of each simplex lies within the simplex. To keep the simplex of original shapes, the Barycentric Dual Graph (BDG), reflecting the relationship of faces based on their barycenters, is extracted to construct the graph of our network. The BDG of a mesh $\mathcal{M}$ is an undirected graph of which each node is the barycenter for each face of $\mathcal{M}$ and an edge between two nodes if the corresponding faces are adjacent in $\mathcal{M}$. Since each face of the mesh in our paper is represented by its barycenter, we can construct our graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ by generating the BDG of the mesh. Moreover, the computational complexity of constructing the BDG is only $\mathcal{O}(N)$, which is less time-consuming than KNN Graph. We denote our BDG as $\mathcal{B}(\mathcal{M})$, and define edge features by utilizing a edge function to operate over the nodes of BDG as Eq. (2)

$$\mathbf{e}_{ij} = g_\Theta\left(\mathbf{x}_i, \mathbf{x}_j\right), \tag{2}$$

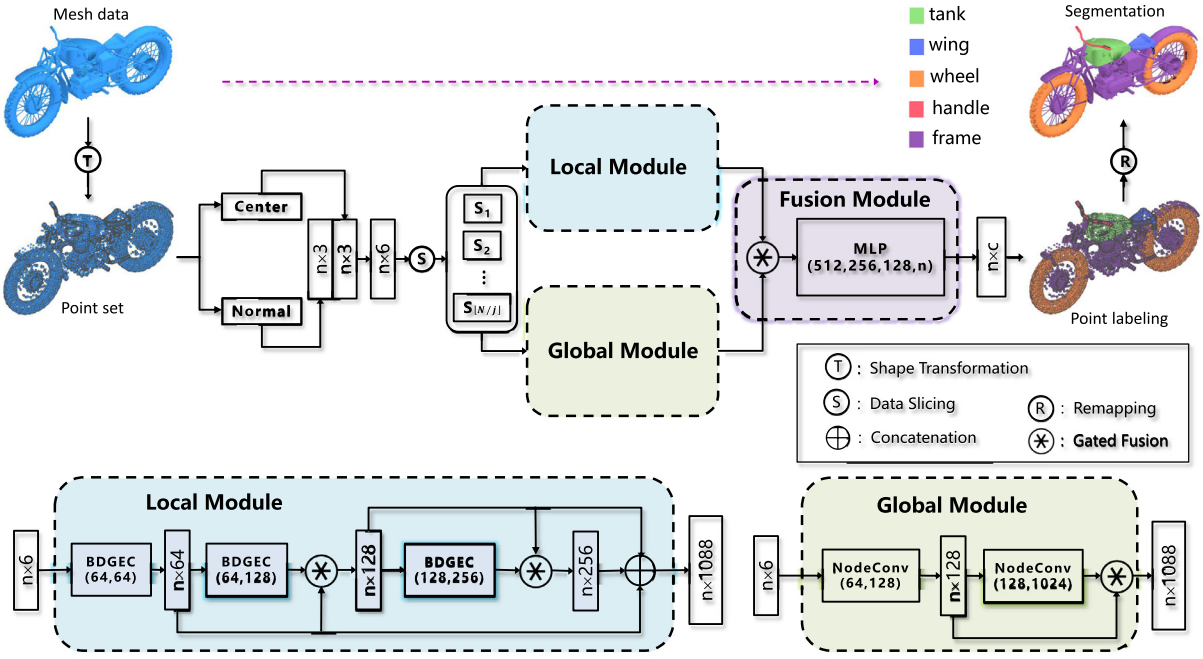

Fig. 2. The whole framework of the local-global combined point-based network. The input shape is firstly transformed into a point set, then fed into the Local Module and Global Module to generate local and global features of shape data in both space and structure, and lastly partitioned into different parts with corresponding point labels.

where $\mathbf{x}_i$ and $\mathbf{x}_j, j \in \{1,2,3\}$ are the current node and its connected nodes respectively, and where $g_\Theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$ is a nonlinear function with a set of learnable parameters $\boldsymbol{\Theta}$. Then we define our Barycentric Dual Graph Edge Convolution (BDGEC) by applying a *max pooling* operation on the edge features associated with only connected edges emanating from each node, as shown in Fig. 4 (b). Consequently, the output of BDGEC at the $i-$th node is obtained by Eq. (3)

$$\mathbf{f}_i = \max g_\Theta\left(\mathbf{x}_i, \mathbf{x}_j\right). \tag{3}$$

Our BDGEC is not only a simple and effective operation faithful to the original mesh in terms of clustering of the mesh faces, but also a lighter convolution than those using KNN to construct the graphs.

3.3. Local-global combined point-based network

LGCPNet is designed for mesh segmentation based on point labeling. A mesh of shape has been transformed into point set as the input data for the net. For three-dimensional shapes, their geometric features are not only manifested locally, but also on the global level. In the segmentation task, the local feature is an important criterion to separate given data into several subsets, and global feature is used to improve segmentation results. Combining two types of features, the points can be classified into one of few predefined part category labels. So as is shown in Fig. 2, our LGCPNet consists of three modules. The Local and Global Module are adopted to capture local and global features respectively, while the Fusion Module combines local and global features to label transformed points.

Local module. For segmentation task, local features are the important features extracted from the local areas of the data. The local areas are always several subsets separated based on the graph constructed from given data. In our Local Module, we use our Barycentric Dual Graph Edge Convolution (BDGEC) as the basic operation to capture local features by taking into account from local point neighborhood information to construct a local graph capturing semantic characteristics. Though the graph in the feature space

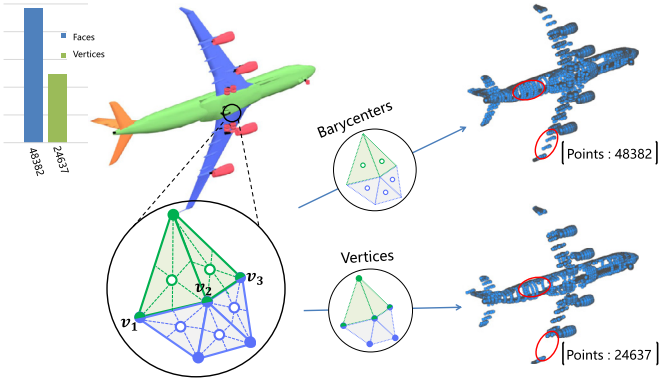

Fig. 3. Illustration of an airplane with four parts transformed into centers(top) and vertices(bottom). Since all datasets provide ground truth segmentation per face, each center can match its only corresponding label, but not for vertices. Besides, the number of centers, equaling to that of faces, is larger than that of vertices.

is fixed on each layer, the weights of edges for each node is dynamically changing for each layer, so the features of each node will keep enlarging. To deactivate the activations of node features that are not deemed relevant by the higher-level information contained, we exploit the gate mechanism instead of classic concatenation to aggregate features from current layer and the last layer as akin to the proposed strategy referred in [33]. As shown in Fig. 2, our Local Module is implemented by three BDGEC layers (64, 64), (64, 128), (128, 256), where two output features from current layer and the last layer are aggregated via a gated fusion, and all output features on each layer are concatenated finally in Local Module. We denote the output features of layer l by $\mathbf{f}_i^l, l = 1,2,3$, and then the feature from two layers can be obtained by Eq. (4)

$$\begin{aligned}\hat{\mathbf{f}}_i^l &= \left(\mathbf{f}_i^l \circledast \mathbf{f}_i^{l-1}\right) \\ &= \left(\left(\mathbf{f}_i^l \odot \alpha_l\right) + \mathbf{f}_i^l\right),\end{aligned} \tag{4}$$

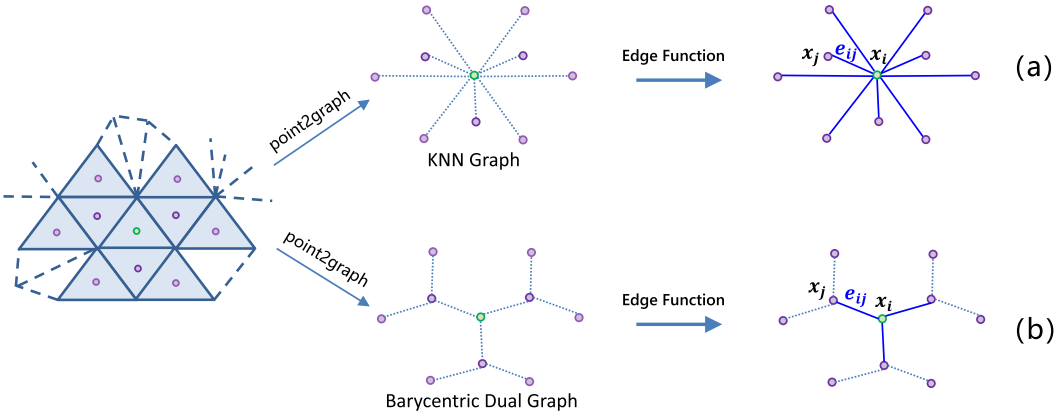

Fig. 4. Comparison of the edge functions operating over the nodes of two kinds of graphs to extract edge features. (a) The graph is computed by KNN and the edge function is calculated by aggregating the edge features associated with all the edges emanating from all the nodes. (b) The Barycentric Dual Graph (BDG) is intrinsic for the given shape and the edge function only take three neighbor nodes into account to extract the edge features. Note that computational complexity of KNN is $\mathcal{O}(N^2)$, while that of BDG is only $\mathcal{O}(N)$.

where α_l is an attention map $\alpha \in \mathbb{R}^N$ by concatenating $\mathbf{f}_i^l$ and $\mathbf{f}_i^{l-1}$ followed by a combo consisting of $Relu + Conv + Sigmoid$:

$$\alpha = Sigmoid\left(C_{1\times1}\left(Relu\left(\mathbf{f}_i^l \| \mathbf{f}_i^{l-1}\right)\right)\right), \tag{5}$$

where $\|$ denotes concatenation of feature maps, and $C_{1\times1}$ is a normalized 1×1 convolutional layer. As a result, the activations of node features on current layer can be deactivate which are not deemed relevant by the higher-level information contained.

Global module. Global feature refers to the overall property of the data, global features are also beneficial to improve segmentation results. To extract the global feature of given data, it is necessary to take all nodes into account, namely, the edge features associated with all the edges emanating from each node. Inspired by PointNet [23], we define NodeConv as the basic operation in our Global Module, and apply *max pooling* as the symmetric aggregation to accumulate features on all nodes. Consequently, our Global Module is implemented by two NodeConv layers (64, 128), (128, 1024) and two outputs of the PointConv are finally gatedly fused as a global feature map of the Global Module. The output features of layer l $\mathbf{v}_i^l, l = 1, 2$ in Global Module can be obtained by Eq. (6)

$$\mathbf{v}_i^l = \max_{j\in N} h_\Theta\left(\mathbf{x}_i, \mathbf{x}_j\right), \tag{6}$$

where h_Θ denote MLPs mapping features of each node. And then the global feature map is attained by fusing outputs of two layers via gate mechanism akin to that in Local Module.

Fusion module. Although both local and global features are extracted, local features are more important than global features for segmentation task. Thus we consider the local feature as regular while the global feature as gated. As a result, after computing local feature map $\mathbf{F}^L$ and global feature map $\mathbf{F}^G$, they two are aggregated by a gated function to acquire the final score matrix, formulated as Eq. (7)

$$\begin{aligned}\hat{\mathbf{F}} &= \left(\mathbf{F}^L \circledast \mathbf{F}^G\right) \\ &= \left(\left(\mathbf{F}^L \odot \alpha\right) + \mathbf{F}^L\right),\end{aligned} \tag{7}$$

where α is computed as Eq. (5). Finally, a MLP with two shared fully connected layers (512, 256), (256, 128) and a softmax layer. The prediction label of each face is the same as its corresponding point determined by the semantic component with the highest probability. At test time, the point label confidences are sequentially mapped to corresponding faces, which produces the same results at once.

4. Experiments

In this section, we will present extensive experimental evaluation in different ways and verify the effectiveness of our method by comparing with state-of-the-art methods. Moreover, we conduct detailed ablation experiments to analyze the contribution of modules in our LGCPNet framework.

4.1. Experimental settings

Datasets. We evaluate our method on the ShapeNetCore [34] and Labeled-PSB (L-PSB) [35,36] datasets reported on the project page.[1] These two datasets are trained and tested separately on each shape category using the same training and test splits per category. Thereinto, the original ShapeNetCore labelings were provided in a point cloud format, which transferred the point labels to mesh polygon labels in ShapePFCN [9] via a nearest neighbors approach combined with graph cuts. While the L-PSB, we used the provided labeled meshes as-is. We also estimate our method on the COSEG [37] datasets provided by MeshCNN [8]. For fair comparison, all datasets we used are based on the ground truth segmentation per face.

Implementation. Our LGCPNet is implemented in PyTorch. All our experiments are performed on a PC with Intel i7 CPU, 32 GB RAM and GTX 1080ti GPU. Our network is trained for 400 epochs with the Adam algorithm with momentum, where the size of mini-batch, initial learning rate, momentum and weight decay are set to 10, 0.01, 0.9 and 0 respectively. All these parameters are fixed throughout this paper.

4.2. Experimental results and comparison

At first, we test the proposed method on the ShapeNetCore dataset. It contains 17,773 shape models across 16 categories, which is a large-scale and challenging dataset and suitable for learning and evaluating deep learning-based methods. The performance of state-of-the-art methods including ShapeBoost [36], Guo et al. [11] and ShapePFCN [9] is reported in Table 1 for the ShapeNetCore dataset. Moreover, EdgeConv [27] where k set 3 is utilized to replace our Barycentric Dual Graph Edge Convolution (BDGEC) in Local Module as the baseline, which is used to demonstrate the validity of the point-based network architecture

[1] http://people.cs.umass.edu/kalo/papers/shapepfcn/

Table 1
Comparison of labeling accuracy on the ShapeNetCore dataset.

	Train/Test	Part Labels	ShapeBoost	Guo et al.	ShapePFCN [2]	Ours(baseline)	Ours
Bag	38/38	2	93.1	91.0	**94.9**	90.8	91.1
Cap	27/28	2	85.9	85.7	**93.6**	92.8	93.3
Knife	196/196	2	81.2	77.1	83.8	90.4	**91.8**
Laptop	222/223	2	86.1	82.7	95.1	**97.2**	**97.2**
Mug	92/92	2	94.9	95.1	95.9	**97.9**	**97.9**
Earphone	34/35	3	81.4	79.8	**85.6**	79.7	81.8
Guitar	250/250	3	89.0	89.9	92.5	97.9	**98.1**
Pistol	137/138	3	88.2	84.1	91.4	94.4	**96.1**
Rocket	33/33	3	79.2	76.9	83.9	83.6	**84.6**
Skateboard	76/76	3	91.0	89.6	**92.1**	88.2	89.7
Table	250/250	3	74.5	77.8	87.8	90.2	**90.3**
Airplane	250/250	4	85.8	87.4	91.2	92.1	**93.6**
Car	250/250	4	79.5	80.1	87.5	92.8	**93.0**
Chair	250/250	4	70.1	66.8	85.5	90.7	**91.8**
Lamp	250/250	4	71.7	71.6	**81.3**	72.3	74.3
Motorbike	101/101	6	77.2	80.1	87.4	93.5	**94.7**
Category Avg.			83.0	82.2	89.4	90.3	**91.2**
Category Avg.(>3 labels)			76.9	77.2	86.6	88.7	**89.8**
Dataset Avg.			81.2	80.6	88.8	90.7	**91.6**
Dataset Avg.(>3 labels)			76.8	76.8	86.5	89.6	**90.6**

on shape segmentation task. In terms of the per-category accuracy, our method achieves the best results in most categories on the ShapeNetCore dataset. Those cases, where our method has the poor performance, have a small size of either scale or number of shapes. For example, Bag, Cap and Earphone have less than 50 shapes for training, while Skateboard and Lamp have the small scale with less than 5k faces per shape. Faced with these two cases, mesh subdivision [38] can be employed before the conversion of mesh to point. For category average(Category Avg.) reported in Table 1, the baseline performs 0.9% better than the best-performing ShapePFCN [9][2] by computing a simple average of the per-category accuracies. This implies that regarding shape segmentation as point labelling is an effective strategy to improve final segmentation results. Moreover, our BDGEC further improves the performance of the baseline by 0.9 % increasing on accuracy. For dataset average(Dataset Avg.), another more objective measure that weights each category by the number of test shapes, our method improves upon the state-of-the-art by 3.2%. Most importantly, in categories with complex objects whose labels are more than 3, our method has significantly higher performance, improves upon the state-of-the-art by 2.8% and 4.1% in category and dataset average respectively. Fig. 5 demonstrates human-labeled ground-truth segmentations of ShapeNet shapes, along with segmentations produced by ShapePFCN [9] and our method for test shapes originating from the ShapeNetCore dataset. We found that ShapePFCN [9], which relies on view-based DNN, may generate significant artifacts(including no, wrong or over-segmentation) highlighted by black ellipses in Fig. 5. In contrast, our LGCPNet, which is point-based DNN dependent on spatial features, has advantages on shapes with complex structure and topology, especially on edges.

Then we analyze the effectiveness of the data slicing though different pretreating strategies of input data with data slicing or not. As reported in Table 2, we set both the number of points for each slice and the number of downsampling points are 1024 and 2048 respectively. By comparison, pretreating with data slicing can achieve the better final segmentation results than that with downsampling. While the number in each slice has little impact to final segmentation result, which implies the robustness of this strategy.

Table 2
Results of different pretreating strategies on the ShapeNetCore dataset.

Number of Points	Data Slicing		Downsampling	
	1024	2048	1024	2048
Category Avg.	91.1	**91.2**	87.5	88.0
Dataset Avg.	91.3	**91.6**	88.3	89.1

Table 3
Comparison of labeling accuracy on the PSB dataset.

	Train/Test	ShapeBoost	ShapePFCN [2]	Ours
psbCup	2	94.0	93.8	**99.1**
psbOctopus	2	98.2	98.3	**99.1**
psbTable	2	99.4	**99.5**	91.1
psbFish	3	95.7	**96.0**	88.6
psbGlasses	3	**96.9**	96.6	92.6
psbPlier	3	95.2	**95.5**	87.0
psbChair	4	98.1	**98.5**	88.9
psbAirplane	5	**96.1**	93.0	93.1
psbAnt	5	**98.7**	98.6	90.4
psbBearing	5	92.2	**92.3**	91.7
psbBird	5	**89.6**	88.5	86.5
psbMech	5	**99.5**	98.7	**99.5**
psbTeddy	5	**98.7**	97.7	91.0
psbVase	5	81.7	**86.8**	83.5
psbFourLeg	6	83.3	85.0	**86.1**
psbHand	6	94.4	84.8	**95.5**
psbBust	8	63.4	68.4	**88.3**
psbHuman	8	86.8	94.5	**95.7**
psbArmadillo	11	92.6	92.8	**95.3**
Category Avg.		92.3	**92.6**	91.2
Category Avg.(>5 labels)		89.8	90.1	**90.5**

We also evaluated labeling accuracy on the PSB dataset reported in Table 3. The train/test ratio of every catagory in the PSB dataset is 12/8. Since the PSB dataset is much smaller than the ShapeNetCore dataset in terms of scale, view-based methods can augment data by increasing the number of views to achieve better performance. But as mentioned in ShapePFCN [9], even a shallow classifier may handle with high accuracy faced with a small number of shapes with limited variability. As a result, even ShapeBoost [36] can keep pace with ShapePFCN [9] among the shapes with labels less than 5. In contrast, our method has outstanding

[2] The labeling accuracy per category cited here is better results for most classes and on average in the case of consistent upright shape orientation and additional input channel in rendered images for encoding the upright axis coordinate values(height from the ground plane).

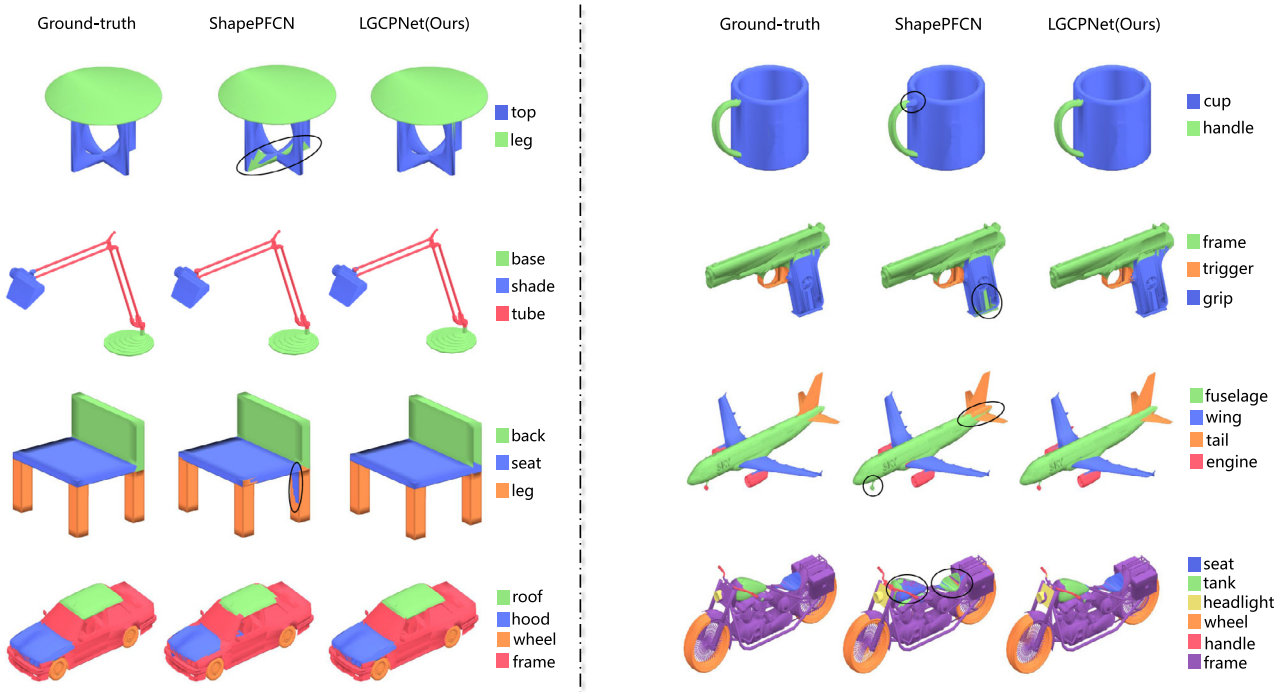

Fig. 5. Human-labeled ground-truth segmentations of ShapeNet shapes, along with segmentations produced by ShapePFCN [9] and our method for test shapes originating from the ShapeNetCore dataset. Significant artifacts(including no, wrong or over-segmentation) are highlighted by black ellipses.

Table 4
Comparison of labeling accuracy on COSEG segmentation.

Method	Vases	Chairs	Telealiens
Edge-based methods			
MeshCNN [8]	92.38	92.99	96.26
PD-MeshNet [39]	95.36	97.23	**98.18**
Edge-based methods			
PointNet++ [10]	91.78	92.98	90.52
DGCNN($k = 20$) [27]	92.04	93.91	91.25
LGCPNet(baseline)	93.30	95.39	92.52
LGCPNet	**96.21**	**98.62**	93.09

performance for the shapes of which segmentation parts are more than 5. With respect to the average accuracy, the performance of the ShapeBoost [36] is almost as good as that of the ShapePFCN [9], while our performance is the best for categories with more than 5 part labels. This obviously implies that the proposed method can deal with difficult shape labeling tasks in classes with complex objects.

For literature comprehensiveness, we evaluate the performance of our method on the task of segmentation on the COSEG [37] datasets. For evaluation, the same shape categories and splitting strategies for both datasets are exploited as MeshCNN [8]. Following PD-MeshNet [39], the input meshes are divided into the following three categories: i) Aliens with 4 class labels consisting of 198 samples (169 training set, 29 test set); ii) Chairs with 3 class labels containing 397 samples (337 training set, 60 test set); iii) Vases with 4 class labels including 297 samples (252 training set, 45 test set). For fair comparison, we measure performance according to the percentage of correctly-labelled mesh faces (face labels) provided by PD-MeshNet [39] and the performance of various methods is reported in Table 4. Two classic point-based networks, PointNet++ [10] and DGCNN [27] respectively, are employed to label points transformed by faces of given shapes. From the results, our method gets the best results on Chairs and Vases, while the baseline is even beaten by PD-MeshNet [39] on these two categories. This further demonstrates the validity of our BDGEC in terms of improving the performance of point-based DNN on shape segmentation. For Aliens, our method is even not up to MeshCNN [8], because it is the least number of training set among three categories and the faces are simplified to $1.5k$ per shape. To sum up, both MeshCNN [8] and PD-MeshNet [39], which focus on edge features, are stable on shape segmentation but complex on pretreating, while our point-based DNN, which focus on face features, are simple and effective but dependent on the large size of either scale or number of shapes. Aggregating edge and face features will be further considered to improve the final segmentation results.

4.3. Ablation study

To analyze the effectiveness of modules in the LGCPNet, we also conduct experiments on three modules of our network, of which three kinds of basic convolutions in Local Module are exploited for comprehensiveness. In detail, we use two kinds of EdgeConv [27], of which k sets 3 and 20 respectively, to replace our BDGEC in Local Module and selecting both two feature-extracted modules without Fusion Module means that local and global features are aggregated via classic concatenation. As demonstrated in Table 5, the best performance appears under circumstance of employing all three modules in our framework. For Local Module, note that our BDGEC is actually a simple but effective case of EdgeConv referred in DGCNN [27], so if the k of EdgeConv is large enough (e.g., 20 or 30), it can achieve segmentation results as good as BDGEC. Here although EdgeConv ($k = 20$) can acquire segmentation accuracy as good as BDGEC, but it consumes 8.4 s for an epoch while 3.4 s for BDGEC. Obviously, our BDGEC is more efficient. For Fusion Module, gated mechanism is beneficial to aggregate local and global features.

4.4. Incomplete shape segmentation

Here we analyze and test the robustness of our network to incomplete shape segmentation. In our method, we focus on the spa-

Table 5
Ablation study of the proposed network on the ShapeNetCore dataset.

Local Module			Global Module	Fusion Module	Category Avg.	Dataset Avg.
BDGEC	EdgeConv $k = 3$	EdgeConv $k = 20$				
✓					89.5	90.0
	✓				88.8	89.1
		✓			89.5	89.9
			✓		88.1	88.7
✓			✓		91.0	91.2
✓			✓	✓	**91.2**	**91.6**

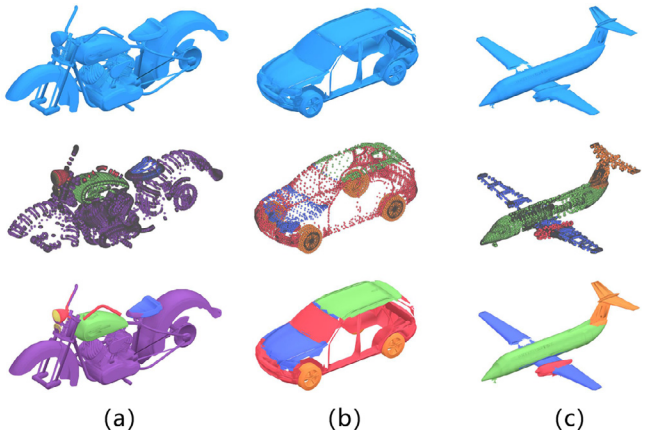

Fig. 6. Segmentation results for fragmentary objects from the ShapeNetCore dataset. (a) (c) are Motorbike, Car, and Airplane. From top to bottom are original incomplete shapes, labeled points, and labeled incomplete shapes respectively.

tial information by defining the point set as a no-frills approximation for an underlying three-dimensional shape. Since given shapes are firstly transformed into discrete points with the topology skipping, our method can be utilized to process the incomplete shape which can not be processed with edge-based [8,39] or view-based [9,15] networks. Actually, our network can learn from broken shapes so that it can also handle incomplete shape segmentation since incomplete shapes can be transformed into points. To validate that, we manually remove some parts of shapes from the test dataset of the ShapeNetCore and then partition them by our network. As shown in Fig. 6, our method can even carry out incomplete shape segmentation.

5. Conclusion and future work

In this paper, we mainly regard shape segmentation as point labeling task taking into account the spatial information of shapes. To achieve this aim, we exploit the pretreating strategy that each face of given shape is converted into its barycenter and normal vector, which focuses on the spatial information of shapes and fits datasets provided ground truth segmentation per face. In our architecture, we design a local-global combined point-based networks (LGCPNet) consisting of three modules, of which Local Module and Global Module capture local and global features respectively, while the Fusion Module uses the gate mechanism to aggregate two types of features. Especially, in Local Module, Barycentric Dual Graph Edge Convolution (BDGEC) is addressed as the basic operation to capture local features by taking into account from local point neighborhood information to construct a local graph capturing semantic characteristics. Compared with classic KNN edge convolution, our BDGEC is more intuitive, simple and efficient when operating on nodes of graphs. From experimental results, proposed method is valid for shape segmentation with point labeling taking the effect to shape segmentation and BDGEC taking the importance of the geometric and spatial features into account for the Barycentric Dual Graph (BDG).

However, our method also has some limitations. Firstly, the proposed method rely on the scale of datasets, namely that larger the numbers of shapes and their faces are, better the final results are. Faced with the small scale datasets, the pretreating strategies for data augment can be used, such as mesh subdivision [38]. Secondly, our method mainly focuses on the spatial information of shapes, while the topology of the given shape is just taken into account the Barycentric Dual Graph (BDG), so more relationships between edges can be considered referred to [8,39]. Last but not least, the attention mechanism [40] can be also utilized to improve the performance once more structural information of shapes will be exploited.

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.

CRediT authorship contribution statement

Boliang Guan: Conceptualization, Methodology, Software, Data curation, Writing - original draft. **Hanhui Li:** Methodology, Data curation, Writing - review & editing. **Fan Zhou:** Investigation, Supervision. **Shujin Lin:** Visualization, Investigation. **Ruomei Wang:** Visualization, Investigation, Writing - review & editing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61902087, No. 61902088), the Guangdong Provincial Natural Science Foundation (No. 2019A1515011953), and the Guangzhou Municipal Science and Technology Project (No. 201902010056).

References

[1] Xu K, Kim VG, Huang Q, Kalogerakis E. Data-driven shape analysis and processing. Computer graphics forum, 36. Wiley Online Library; 2017. p. 101–32.
[2] Rostami R, Bashiri FS, Rostami B, Yu Z. A survey on data-driven 3d shape descriptors. Computer graphics forum, 38. Wiley Online Library; 2019. p. 356–93.
[3] Lavoué G, Dupont F, Baskurt A. A new cad mesh segmentation method, based on curvature tensor analysis. Comput Aided Des 2005;37(10):975–87.
[4] Golovinskiy A, Funkhouser T. Randomized cuts for 3d mesh analysis. ACM Transactions on graphics (TOG), 27. ACM; 2008. p. 145.
[5] Yan D-M, Wang W, Liu Y, Yang Z. Variational mesh segmentation via quadric surface fitting. Comput Aided Des 2012;44(11):1072–82.
[6] Wang H, Lu T, Au OK-C, Tai C-L. Spectral 3d mesh segmentation with a novel single segmentation field. Graph Models 2014;76(5):440–56.
[7] Kaick OV, Fish N, Kleiman Y, Asafi S, Cohen-Or D. Shape segmentation by approximate convexity analysis. ACM Trans Graph (TOG) 2014;34(1):4.
[8] Hanocka R, Hertz A, Fish N, Giryes R, Fleishman S, Cohen-Or D. MeshCNN: a network with an edge. ACM Trans Graph (TOG) 2019;38(4):1–12.
[9] Kalogerakis E, Averkiou M, Maji S, Chaudhuri S. 3D shape segmentation with projective convolutional networks. In: Proceedings of the CVPR, 1; 2017. p. 8.
[10] Qi CR, Yi L, Su H, Guibas LJ. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the advances in neural information processing systems; 2017a. p. 5099–108.

[11] Guo K, Zou D, Chen X. 3D mesh labeling via deep convolutional neural networks. ACM Trans Graph (TOG) 2015;35(1):3.
[12] Wang P, Gan Y, Shui P, Yu F, Zhang Y, Chen S, et al. 3D shape segmentation via shape fully convolutional networks. Comput Graph 2018;70:128–39.
[13] Yi L, Su H, Guo X, Guibas LJ. SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation.. In: Proceedings of the CVPR; 2017. p. 6584–92.
[14] Zhou J, Liu X, Cao J, Wang W, Yin B. Deep mesh labeling via learned semantic boundary guidance. Comput Aided Des 2018;101:72–81.
[15] Le T, Bui G, Duan Y. A multi-view recurrent neural network for 3D mesh segmentation. Comput Graph 2017;66:103–12.
[16] Guo Y, Wang H, Hu Q, Liu H, Liu L, Bennamoun M. Deep learning for 3d point clouds: A survey. arXiv 2019:arXiv–1912.
[17] Tsuchie S, Hosino T, Higashi M. High-quality vertex clustering for surface mesh segmentation using student-t mixture model. Comput Aided Des 2014;46:69–78.
[18] Huang Q, Koltun V, Guibas L. Joint shape segmentation with linear programming. ACM Transactions on graphics (TOG), 30. ACM; 2011. p. 125.
[19] Sidi O, van Kaick O, Kleiman Y, Zhang H, Cohen-Or D. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering, 30. ACM; 2011.
[20] Meng M, Xia J, Luo J, He Y. Unsupervised co-segmentation for 3D shapes using iterative multi-label optimization. Comput Aided Des 2013;45(2):312–20.
[21] Wu Z, Wang Y, Shou R, Chen B, Liu X. Unsupervised co-segmentation of 3D shapes via affinity aggregation spectral clustering. Comput Graph 2013;37(6):628–37.
[22] Shu Z, Qi C, Xin S, Hu C, Wang L, Zhang Y, et al. Unsupervised 3d shape segmentation and co-segmentation via deep learning. Comput Aided Geom Des 2016;43:39–52.
[23] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: deep learning on point sets for 3d classification and segmentation. In: Proceedings of the computer vision and pattern recognition (CVPR), IEEE, 1; 2017b. p. 4.
[24] Klokov R, Lempitsky V. Escape from cells: deep kd-networks for the recognition of 3D point cloud models. In: Proceedings of the 2017 IEEE international conference on computer vision (ICCV). IEEE; 2017. p. 863–72.
[25] Atzmon M, Maron H, Lipman Y. Point convolutional neural networks by extension operators. ACM Trans Graph (TOG) 2018;37(4):1–12.
[26] Li Y, Bu R, Sun M, Wu W, Di X, Chen B. Pointcnn: convolution on x-transformed points. In: Proceedings of the advances in neural information processing systems; 2018. p. 828–38.
[27] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph CNN for learning on point clouds. ACM Trans Graph (TOG) 2019;38(5):146.
[28] Zhao H, Jiang L, Fu C-W, Jia J. Pointweb: enhancing local neighborhood features for point cloud processing. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2019. p. 5565–73.
[29] Wu W, Qi Z, Fuxin L. Pointconv: deep convolutional networks on 3D point clouds. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2019. p. 9621–30.
[30] Rustamov RM. Barycentric coordinates on surfaces. In: Computer Graphics Forum, 29. Wiley Online Library; 2010. p. 1507–16.
[31] Mullen P, Memari P, de Goes F, Desbrun M. Hot: Hodge-optimized triangulations. In: ACM SIGGRAPH 2011 Papers; 2011. p. 1–12.
[32] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. MIT Press; 2009.
[33] Takikawa T, Acuna D, Jampani V, Fidler S. Gated-SCNN: gated shape CNNs for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision; 2019. p. 5229–38.
[34] Yi L, Kim VG, Ceylan D, Shen I, Yan M, Su H, et al. A scalable active framework for region annotation in 3D shape collections. ACM Trans Graph (TOG) 2016;35(6):210.
[35] Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3D mesh segmentation. In: ACM Transactions on graphics (TOG), 28. ACM; 2009. p. 73.
[36] Kalogerakis E, Hertzmann A, Singh K. Learning 3D mesh segmentation and labeling. ACM Trans Graph (TOG) 2010;29(4):102.
[37] Wang Y, Asafi S, Van Kaick O, Zhang H, Cohen-Or D, Chen B. Active co-analysis of a set of shapes. ACM Trans Graph (TOG) 2012;31(6):165.
[38] Cashman TJ. Beyond Catmull–Clark? A survey of advances in subdivision surface methods. In: Computer graphics forum, 31. Wiley Online Library; 2012. p. 42–61.
[39] Milano F, Loquercio A, Rosinol A, Scaramuzza D, Carlone L. Primal-dual mesh convolutional neural networks. In: Proceedings of the conference on neural information processing systems (NeurIPS); 2020.
[40] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the advances in neural information processing systems; 2017. p. 5998–6008.