Learning on 3D Meshes with Laplacian Encoding and Pooling

Yi-Ling Qiao, Lin Gao*, Jie Yang, Paul L. Rosin, Yu-Kun Lai and Xilin Chen

Abstract—3D models are commonly used in computer vision and graphics. With the wider availability of mesh data, an efficient and intrinsic deep learning approach to processing 3D meshes is in great need. Unlike images, 3D meshes have irregular connectivity, requiring careful design to capture relations in the data. To utilize the topology information while staying robust under different triangulations, we propose to encode mesh connectivity using Laplacian spectral analysis, along with mesh feature aggregation blocks (MFABs) that can split the surface domain into local pooling patches and aggregate global information amongst them. We build a mesh hierarchy from fine to coarse using Laplacian spectral clustering, which is flexible under isometric transformations. Inside the MFABs there are pooling layers to collect local information and multi-layer perceptrons to compute vertex features of increasing complexity. To obtain the relationships among different clusters, we introduce a Correlation Net to compute a correlation matrix, which can aggregate the features globally by matrix multiplication with cluster features. Our network architecture is flexible enough to be used on meshes with different numbers of vertices. We conduct several experiments including shape segmentation and classification, and our method outperforms state-of-the-art algorithms for these tasks on the ShapeNet and COSEG datasets.

Index Terms—Mesh Processing, Segmentation, Laplacian, Deep Learning

1 Introduction

Analyzing high-quality 3D models is of great significance in computer vision and graphics. Better understanding of 3D shapes would benefit many tasks such as segmentation, classification and shape analysis. Recently, deep learning methods have been prevalent in 2D image processing tasks such as image classification [1], [2] and semantic segmentation [3], [4]. With the help of large-scale image datasets and improved computational resources, deep learning methods boost the performance of image processing algorithms by a large margin. Inspired by the success in images, researchers also apply learning algorithms to 3D data.

Recently, large-scale 3D datasets have made it possible to train neural networks for 3D shapes. Nevertheless, it is not a simple extension to apply neural networks in the 3D space. There are various 3D representations. The majority of 3D representations such as meshes, point clouds etc. are non-canonical, requiring special design to feed them through neural networks. To address this, some approaches are trained on ModelNet [5] and deal with voxels, but the resolution of voxel data is limited due to the curse of dimensionality. Alternatively, point clouds representing an object by a set of unstructured points with their xyz coordinates are commonly used. However, point clouds do not carry connectivity information and therefore are less efficient than meshes to represent shapes, and may have ambiguities when two surfaces are close. Another representation, the 3D mesh, is a fundamental data structure in computer graphics and vision, which not only encodes geometry but also topology and therefore has better descriptive power than the point cloud. A mesh is a graph with vertices, edges and faces that characterize the surface of a shape. For deep learning methods, mesh data is more compact but irregular when compared to voxels, making the equivalent of simple operations in the image domain such as convolutional kernels highly non-trivial. It also contains richer structure than a point cloud, which can be exploited when learning on 3D meshes. This paper proposes a flexible network structure that can utilize connectivity information while staying robust under different triangulation.

To learn on 3D meshes, we propose the Laplacian Encoding and Pooling Network, which takes raw features of mesh models as input and outputs a function defined on the vertices. Inspired by image processing networks, we observe an intuitive principle that vertex features should be computed independently and associatively in different parts of the network. We therefore extend this methodology into the non-Euclidean space of 2-manifolds. In our design as in Fig. 1, the basic network structure involves consecutive mesh feature aggregation blocks (MFABs). Each block can split the surface into patches, like super-pixels in the image domain, by Laplacian spectral clustering. After splitting, the MFAB can simultaneously compute features of individual vertices and clusters. Considering the relationships between clusters, we use a Correlation Net to compute a matrix that can fuse the information globally. Compared to images, a

* *Corresponding author is Lin Gao.*

- *Y.-L Qiao is with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences and also with with Department of Computer Science, University of Maryland, College Park, US. E-mail: yilingq97@gmail.com*
- *L. Gao and J. Yang are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences and also with the University of Chinese Academy of Sciences, Beijing, China. E-mail: {gaolin, yangjie01}@ict.ac.cn*
- *P.L Rosin and Y.-K Lai are with School of Computer Science and Informatics, Cardiff University, Wales, UK. E-mail: {RosinPL, LaiY4}@cardiff.ac.uk*
- *X. Chen is with Institute of Computing Technology, Chinese Academy of Sciences and also with the University of Chinese Academy of Sciences, Beijing, China. E-mail: xlchen@ict.ac.cn.*

major difficulty for learning on meshes is that the vertices are unordered, and so are the clusters. For this reason, a fully-connected layer cannot work out the correlation matrix effectively. Therefore, we disentangle the correlation by independently mapping the clusters into a vector space. Then, the correlation between a pair of clusters is determined by the inner product of their corresponding vectors.

Prior to our method, effort has been made to learn on 3D meshes. Please refer to [6] for the history and frontier research of deep learning on geometric data. Some works exploit geodesic distances on shapes [7], [8]. Based on this metric they design a spatial convolutional operator. Geodesic distance is invariant under isometric transformation, making their networks more robust. Compared to their directly computing the distance, our network utilizes the geodesic distance in an implicit way. The clustering is performed in the Laplacian feature domain, where Euclidean distance can approximate geodesic distance on the manifold [9]. As a result, the pooling is robust to isometric deformation and different triangulations (shown in Fig. 5). Others choose to work in the spectral domain [10], [11], by defining convolutional kernels in the Fourier domain. However, the dependency of coefficients on the domain basis makes it difficult to share weights across shapes. Consequently, works like [12] have modules purposefully designed to synchronize the basis of domains. Our network does not suffer from changing domains, and we also propose a flexible structure, Correlation Net, to address the alignment of clusters across models. Compared to all the aforementioned methods, we use both spectral and spatial information, such that our network can utilize the connectivity of meshes while staying robust under different domains with inconsistent triangulation.

The pipeline of our approach is shown in Fig. 1, which learns cross-domain mesh functions using both spatial and spectral information. Overall, our network has the following three modules: 1) the preprocessing step computes vertex features and clusters from the raw mesh data; 2) the Mesh Feature Aggregation Block (MFAB) calculates local features within local regional clusters and collects global information through Correlation Net; 3) the last part of the network depends on the specific application, e.g., it may output segmentation masks or classification labels.

In the experiments, we first evaluate the importance of mesh feature aggregation blocks and the choice of the input features. We then justify that our single network can deal well with different mesh models with different numbers of vertices. To test the capability of our network, we train our network on the ShapeNet and COSEG datasets to perform classification and segmentation tasks, which are fundamental shape understanding tasks in computer vision, and show superior overall performance.

The main contributions of our method are as follows:

- We propose the Laplacian Encoding and Pooling Network, a general network for learning on 3D meshes, which can utilize the connectivity of meshes while staying robust under different triangulations.
- We propose a flexible pooling operation that can split model surfaces into clusters, like superpixels in images. By varying the clusters from fine to coarse, the network can process meshes hierarchically.
- We introduce a Correlation Net to compute the relationship among clusters. The computation process circumvents the randomness of cluster ordering, enabling consistency across domains.

2 Related Work

We first summarize deep learning methods on 3D representations, and then provide a brief introduction to alternative input shape features. Finally, we review recent methods for mesh segmentation, which is a fundamental task when analyzing shapes.

3D Deep Learning. With the increasing availability of 3D models, deep learning methods for analyzing 3D data structure have been widely studied nowadays. There are several representations for 3D shapes, including voxels, point clouds and meshes. The voxel representation is similar to pixels in the 3D space, which can utilize a direct extension of 2D convolutional networks [13]. The point cloud is intensively researched, with works like [14] learning the transformation matrix of points and obtaining decent results on multiple datasets. Qi et al. [15] further propose PointNet++ that adds pooling operations, where pooling areas are selected by nearest neighbors. We aim at different problems: While they address problems on point clouds, our method focuses on meshes. To better understand 3D meshes, the topology information is deeply exploited in our method. First, we use topology to perform clustering. In contrast, the nearest neighbor clustering strategy of PointNet++ cannot perceive surface structure of meshes. Second, we use features that contain topology information as input. Meanwhile, we believe that it is not appropriate to carry the entire topology during the whole process. The connectivity information is irregular, large, hard to compute, and sensitive to noise. We instead condense the topology information by encoding the connectivity into the pooling areas and input features.

For the mesh representation, spatial methods [7], [8] define convolutional kernels on surfaces. Our method also utilizes spatial information, as the pooling operation partitions the surface into patches, acting like super-pixels in image processing. For spectral methods, Bruna et al. [10] introduce a spectral convolutional layer on graphs, which can be interpreted as convolutions in the Fourier domain. Henaff et al. [16] handle large-scale classification problems. They propose to exploit the underlying relationships among individual data points by constructing a graph of the dataset and then solving a graph-based classification. When processing the graph, they introduce a CNN structure with spectral pooling. Compared with ours, their work focuses on the nodes and the pooling operation does not take advantage of geometric information.

The way our method uses the Laplacian is different from many other works on graphs. Defferrard et al. [17] use the graph Laplacian operator to construct a convolution kernel which can extract localized features in a discrete graph. After that, variants of graph neural networks (GNNs) have been applied to learning from 3D data [18], [19]. Recently, Kostrikov et al. [20] propose upgrades to GNNs and use the Dirac operator in the network. Our method uses the

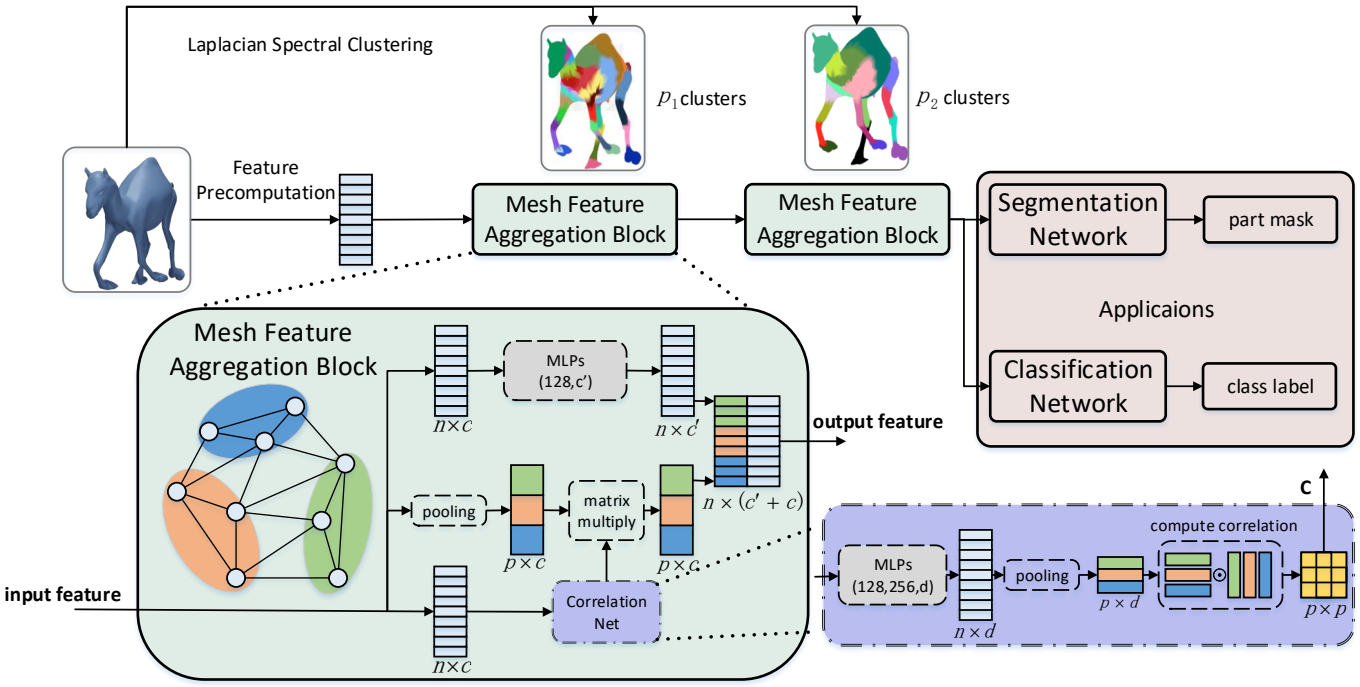


Fig. 1. Overview of our Laplacian Pooling Network. Given a 3D mesh as input with the aim of producing some kind of vertex function (depending on the application) as output, the pipeline of our network has three main components. First we preprocess the mesh to compute Laplacian eigenvectors and spectral clustering. These along with vertex coordinates and normals form the input feature to the network. Second, we stack several Mesh Feature Aggregation Blocks (MFABs) to analyze the shape model under multiple resolutions. An MFAB includes some multi-layer perceptrons (MLP) to compute features independently for each vertex, and also a pooling layer to compute the features for clusters. A Correlation Net learns a correlation matrix C to fuse features across clusters. Then the fused features are concatenated with the MLP results, and then associated with individual vertices in the cluster. After a sequence of MFABs (two in this illustration), the features are fed into the Application Network to produce output according to specific applications.

Laplacian in a different way. To exploit the manifold structure, we perform spectral analysis on the mesh Laplacian with the cotangent weights instead of using the graph Laplacian operator. These two Laplacians also have different formulations.

A fundamental problem of spectral convolution is generalizing across domains since the coefficients of spectral filters are basis dependent [6]. To address this problem, Yi et al. [12] further propose SyncSpecCNN to perform convolutional operations in a synchronized spectral domain, where they train a Spectral Transformer Network to align the functions in different domains. Compared to all the spectral CNN methods above, our method takes advantage of Laplacian spectral analysis to encode mesh topology and identify a spatial clustering strategy but avoids suffering from its dependency on the domain. We also show in the Experiments section that our method is robust under different object categories, number of vertices, and triangulations.

Recently, Song et al. [21] use a multi-view representation of meshes and apply a CNN on it. Some other works design regular CNNs on surface meshes by parametrizing the manifold. Ezuz et al. [22] map unstructured geometric data to a regular domain by optimizing the metric distortion. Toric covering [23], [24] is also used to define convolutions. MeshCNN [25] learns features of a mesh defined on edges and the authors design a learnable pooling operation via edge collapse.

Shape Features. Over the years, many shape features have been developed to describe shape characteristics, including curvatures, geodesic shape contexts, geodesic distance features as used in [26], Heat Kernel Signatures [27], Wave Kernel Signatures [28], etc. Our network exploits mesh Laplacian spectral analysis, which provides an effective encoding of mesh topology. Laplacian eigenvectors also help us to cluster vertices for pooling layers, and it is an intrinsic feature describing the geometry of shapes. Readers can refer to [29] for details about computation and applications of graph Laplacian. However, two essential problems with Laplacian eigenvectors are that the sign is not well-defined, and perturbation occasionally occurs in high frequency terms. To eliminate these ambiguities, we use the absolute value of low frequency terms as input.

Mesh Segmentation. Mesh segmentation has long been a fundamental task in the field of computer vision and graphics. There are unsupervised and supervised methods to perform this task. For unsupervised methods, recent work usually uses the correspondence or correlation between shape units to co-segment a collection of objects in the same category [30], [31], [32], [33]. Those methods essentially analyze a whole dataset of similar 3D shapes and cluster shape parts that can be consistently segmented into one class. Other works try to take advantage of labeled data to develop a supervised method. Thanks to recent shape segmentation datasets [34], [35], [36], [37], supervised methods obtain higher accuracy than unsupervised ones. Among all those datasets, COSEG [35] and ShapeNet [35]

have sufficiently many samples to train a network with reasonable generalizability, so we conduct experiments on the two datasets. Previous deep learning methods usually design different architectures to perform segmentation. For example, George et al. [38] design a multi-branch 1D convolutional network and Wang et al. [39] put convolutional kernels on neighboring points and a pooling layer on the coarsened mesh. A 2D CNN is embedded into a 3D surface network by a projection layer in [40]. Guo et al. [41] concatenate different feature vectors into a rectangular block and apply CNNs to this image-like domain. Our method aims to develop a general network for learning on 3D meshes, and we demonstrate that our general approach outperforms existing methods in most cases.

3 METHODOLOGY

3.1 Problem Statement

Our proposed network is a general method for 3D meshes, capable of dealing with different numbers of vertices. Suppose that the current input $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mesh with N vertices. There is an input feature function f defined on vertices $\mathcal{V}$, i.e., $f : \mathcal{V} \to \mathbb{R}^{N \times c}$ where c is the dimension of input features, typically containing coordinates, normals, the mesh Laplacian, curvatures, etc. At the same time, there is a target function g that we aim to produce. It can usually be a vector function such as a segmentation $g : \mathcal{V} \to {C_s}^N$ where each entry corresponds to the label of a vertex, or a single value like classification category for the whole object $g : \mathcal{G} \to C_l$, where C_s and C_l are the sets of segmentation labels and classification categories respectively. We would like to mention that g may also represent other functions including texture or normals. Our aim is to design a general neural network that learns the mapping from input feature f to the output g. For the segmentation and classification tasks, we precompute the normals and mesh Laplacian eigenvectors as input features. Our network will output an $N \times |C_s|$ matrix for segmentation, which gives the score for each vertex belonging to each segmentation label, or a $|C_l|$ dimensional softmax vector for classification.

3.2 Feature Precomputation Module

The inputs to our network are vertices, faces, and features. To use minimal features to characterize local geometric information, we design a feature precomputation module in the network to compute normals and Laplacian vectors from vertices and faces. Moreover, a k-means operator is also included in the Feature Precomputation module to perform clustering.

For later pooling layers, we use Laplacian spectral clustering [29] at multiple resolutions. Different from spectral convolutions in [16], our pooling layers reduce any number of vertices to a desired dimension, which makes it possible for our method to cope with meshes with different topology. In practice, the Laplacian matrix is computed by

$$\boldsymbol{L} = \boldsymbol{A}^{-1}(\boldsymbol{D} - \boldsymbol{W}), \tag{1}$$

where $\boldsymbol{A} = diag(a_1, ..., a_n)$ are vertex weights defined as local Voronoi areas a_i, equal to one third of the sum of one-ring triangles areas. $\boldsymbol{W} = \{w_{ij}\}_{i,j=1,...,N}$ is the sparse cotangent weight matrix, a discretization of the continuous Laplacian over mesh surfaces [42], and $\boldsymbol{D}$ is the degree matrix which is a diagonal matrix with diagonal entries $d_{ii} = \sum_{j=1}^{N} w_{ij}$. The Laplacian feature $\boldsymbol{\Phi}$ is the set of eigenvectors of matrix $\boldsymbol{L}$. Please refer to [29] for detailed computation and applications of graph Laplacian.

To cluster vertices at different levels, we perform k-means clustering on $\boldsymbol{\Phi}$ with different numbers of clusters $k = p_l$ such that vertices are clustered into p_l clusters for the l^{th} pooling block. $l = 1, 2, \ldots, L$, and L is the number of levels. To achieve local-global feature extraction, p_l decreases as l increases. Note that since the clustering is in the feature domain, vertices on the surface within one cluster are not necessarily connected. This is also reasonable because some semantically similar vertices can be far away.

3.3 Network Architecture

The architecture of our network is illustrated in Fig. 1.

Given the preprocessed feature function f defined on vertices $\mathcal{V}$, our network takes the feature matrix as input, each row being a feature vector of a vertex. By reducing the input features to a matrix, we avoid a complex graph structure and make it tractable for neural networks.

Several mesh feature aggregation blocks (MFABs) are then applied in various resolutions for multiple times. At the end of MFAB blocks, the application network outputs the target function g. Details about pooling blocks will be further discussed in the next section. The architecture of the application network for classification and segmentation is shown in Fig. 2.

In our design, to circumvent the complex and irregular topology of mesh data, we seek a pipeline that can concisely describe the relationship among vertices. Instead of directly processing edges $\mathcal{E}$, we simplify this problem by only processing vertices $\mathcal{V}$ of mesh $\mathcal{G}$. Nevertheless, edges are not ignored, but instead implicitly encoded into the Laplacian eigenvectors and spectral clusters as described in the previous subsection. Since the mesh Laplacian is intrinsically induced from geodesic distances, our method is robust to remeshing and isometric transformations.

Moreover, since the total number of vertices in a mesh can vary significantly from model to model, an ideal network architecture should be able to deal with meshes with different numbers of vertices. Our solution is to design mesh feature aggregation blocks, which turn meshes of arbitrary sizes into levels with the same number of clusters. By stacking together several blocks in a multi-resolution manner, our network can learn to extract useful features from the mesh. In addition, our network uses parameters more effectively with shared weight Multi-Layer Perceptron (MLP), which also helps avoid over-fitting by reducing the complexity of our network. Our method consequently achieves good results for shape classification and segmentation.

3.4 Mesh Feature Aggregation Blocks

A mesh feature aggregation block is composed of three modules: the Multi-Layer Perceptron (MLP) layers [43], the pooling layers, and a Correlation Net. Fig. 1 shows an illustration of a mesh feature aggregation block. Each mesh feature aggregation block obtains its input feature $\boldsymbol{F}_l$ of

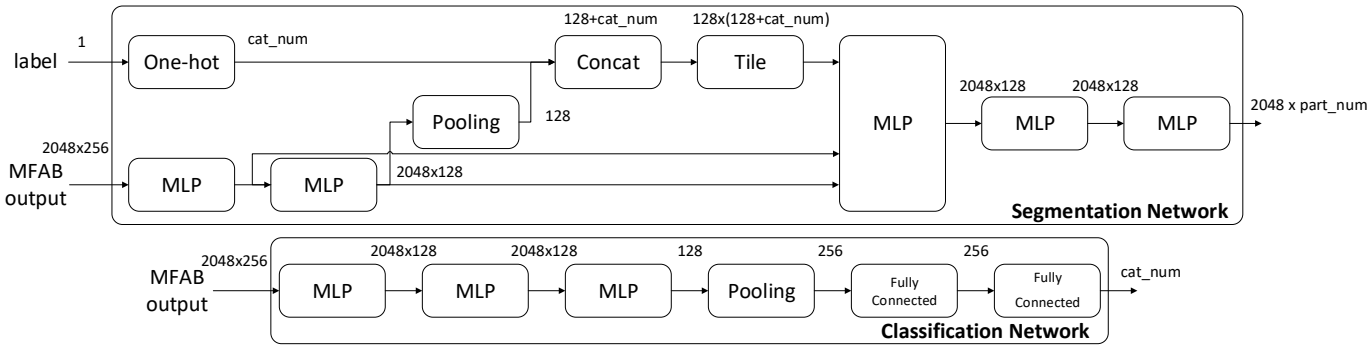


Fig. 2. The application network for segmentation and classification. For the segmentation task (top row), the application specific network takes as input the category label of a certain item and the features defined on vertices. The vertex features go through two MLP layers and then are duplicated into two branches, one of which is sent to a global pooling, combined with the one-hot vector of input label and then attached back to vertex features from the other branch. Finally, two MLP layers are used to compute the final segmentation mask. For the classification network (bottom row), the vertex features sequentially go through MLP layers, global pooling, and fully connected layers. Eventually a softmax vector for candidate labels is predicted.

size $N \times c_l$ from the previous layer. It also gets the cluster mask $\boldsymbol{M}_l$ from the precomputation step, where an entry $m_{l,i} \in \{1, 2, \ldots, p_l\}$ in the mask $\boldsymbol{M}_l \in \mathbb{Z}^N$ indicates for node i which cluster it belongs to. The total number of clusters for the l^{th} block is p_l.

As illustrated in Fig. 1, the data flows through three branches. The upper path is a series of MLP layers, learning vertex features of increasing complexity; the middle path is the pooling layer followed by a correlation matrix multiplication, which fuses global and local information; the bottom path is a Correlation Net that computes the correlation matrix, learning the interaction among clusters.

The upper path of the MFAB in Fig. 1 is a set of MLP layers with shared weight perceptrons connected to all the vertices. For a certain vertex i, an MLP layer multiplies its input $\boldsymbol{F}_{l,i}$ and weight matrix $\boldsymbol{W}$ with bias $\boldsymbol{b}$, followed by a $ReLU(\cdot)$ activation function. The operation of this layer can be expressed as

$$MLP(\boldsymbol{F}_{l,i}) = ReLU(\boldsymbol{W}\boldsymbol{F}_{l,i} + \boldsymbol{b}). \tag{2}$$

For the pooling layer, the input includes features $\boldsymbol{F}_l$ from the last block and a cluster mask $\boldsymbol{M}_l$. Its result is defined as applying the operation to all the nodes belonging to the same cluster. Take max pooling as an example. The pooling result $\boldsymbol{P}_{l,j}$ corresponding to the j^{th} cluster of the l^{th} block is:

$$\boldsymbol{P}_{l,j} = \max_{m_{l,i}=j} \boldsymbol{F}_{l,i}. \tag{3}$$

Furthermore, we want the features to be computed across clusters. For images, convolutional kernels can be used to fuse pooling results. However, since triangle meshes do not have a regular grid and consistent clustering, such an approach does not work. Standard global pooling can be a simple choice, but each cluster has equal contribution and detailed information is lost. At the bottom path in Fig. 1, in order to aggregate information from all clusters, we multiply the pooling results with a correlation matrix $\boldsymbol{C}_l = \{c_{ij}\}_{p_l \times p_l}$. Each entry c_{ij} measures the correlation between the i^{th} and j^{th} clusters, such that the aggregated pooling result is obtained as $\tilde{\boldsymbol{P}}_{\boldsymbol{l}} = \boldsymbol{C}_{\boldsymbol{l}}\boldsymbol{P}_{\boldsymbol{l}}$. The Correlation Net computes the matrix $\boldsymbol{C}$ by learning a latent vector embedding for each cluster:

$$\boldsymbol{\Psi}_l = \{\psi_{lj}\} = Pooling(MLP(\boldsymbol{F}_l)), \tag{4}$$

and entries of the correlation matrix are inner products of latent vectors of cluster pairs $c_{lij} =< \boldsymbol{\psi}_{li}, \boldsymbol{\psi}_{lj} >$. Then, we can get

$$\tilde{\boldsymbol{P}}_l = \boldsymbol{\Psi}_l \boldsymbol{\Psi}_l^{\boldsymbol{T}} \boldsymbol{P}_l. \tag{5}$$

The concatenation layer combines vertex features from the upper-path MLPs and aggregated pooling results. For the vertex i in cluster j, its output feature is written as

$$\{MLP(\boldsymbol{F}_{l,i}), \tilde{\boldsymbol{P}}_{l,j}\}. \tag{6}$$

In summary, the MFAB is used for both processing the local information and finding the relationships among local patches. The motivation for pooling in the mesh is to better understand spatial information. Ideally, the input vertices are hierarchically clustered into different areas, and the relationships between areas need to be considered. In practice, we process the hierarchical information by clustering vertices into different numbers of clusters. The Correlation Net learns to compute a correlation matrix to describe the relative relationships among spatial areas after pooling. Using Laplacian clustering ensures that the clustering is robust under different triangulation. Also, the Euclidean distance in the Laplacian feature domain approximates the geodesic distance [9]. Therefore, such a clustering can find meaningful patches in the spatial domain. Moreover, the clustered areas give a reasonable partition, as shown in the camel example in Fig. 1 where vertices with the same color roughly belong to one semantic area.

4 Experiments

4.1 Implementing Details

We implement our method using Tensorflow. We train and test the network on a desktop computer with an NVIDIA GTX1080 GPU. The optimization is achieved using the Adam [44] solver with learning rate 7×10^{-4} and momentum 0.9. The network is trained for 200 epochs with batch

size 8. The input features have 22 dimensions, including 6 dimensions of positions and normals, and the other 16 dimensions are the absolute values of the Laplacian eigenvectors corresponding to the 16 lowest frequencies. Eigenvectors corresponding to similar eigenvalues might have the order swapping problem. This issue is mitigated since the swapped eigenvectors usually lie in a subspace with similar values, and instead we choose to use low-frequency ones which are less likely to have repeated eigenvalues. Note that users can also specify their own input features depending on different datasets and tasks. According to the experimental results in the next section, by default we choose to use two mesh feature aggregation blocks with 16 and 8 clusters respectively. There are two MLP layers in a mesh feature aggregation block outputting 128 and 256 channels. Following MFABs is a specific application network based on the task to be performed. In total, the network's depth is 11 for the segmentation and classification tasks.

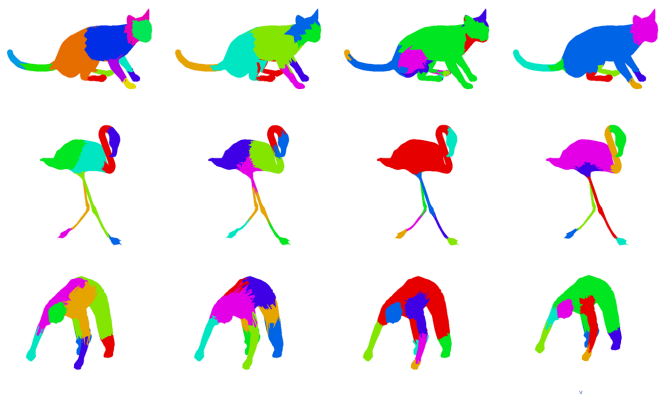
4.2 Network Evaluation

We now evaluate different parts of our network. This series of evaluation is conducted on the COSEG dataset, with results shown in Tab. 1.

Input features. First we test the usefulness of the input features. The performance of a certain algorithm can be affected by the features used. In the experiments, we use coordinates, normals and Laplacian eigenvectors as vertex features. In this part, we test the network without one of those three features. We can see that a combination of all the features achieves the best results.

MFAB design. Second, we test the setting of mesh feature aggregation blocks. By default, our method uses 2 MFABs. We compare this with alternative numbers of MFABs ranging from 0 to 3. The results show that 2 MFABs (our default method) gives the best performance. We also test the usefulness of our Correlation Net. Ours-noCorrNet is the network without the Correlation Net and matrix $\boldsymbol{C}$. We observe that the aggregation of global information is important for our network.

Clustering strategy. We compare with alternative clustering strategies in the segmentation task on the Human Body Segmentation [23] dataset. The illustration of different clustering strategies is shown in Fig. 3. Here, (a) shows the clustering results of our method. In (b), we treat the manifold as a point cloud and cluster the points based on Euclidean coordinates. (c) adopts the strategy in PointNet++ [15], which uses furthest point sampling and k-nearest neighbors to group vertices. (d) enforces inclusion relationships across pooling hierarchies. We also design an experiment where the multi-level clusters have rigorous inclusion relationships. Fig. 4 gives an illustration of the pooling process of (d). We still use k-means to cluster the shape. In each level, clusters have their centroids, and the next level of hierarchy performs clustering of the centroids of the previous level. By doing so, an area in level l will be completely included in an area in level $l+1$. The results in Table 3 show that the inclusion relationship in fact has a negative impact. The representative points of clusters may not exist in a manifold, so the follow-up clustering results



(a) Ours (b) Euclidean (c) Grouping (d) Inclusion

Fig. 3. Visualization of different clustering strategies. (a) shows the pooling areas of our method. (b) shows the segmentation results clustered using Euclidean coordinates. This clustering does not perceive underlying manifold structure, e.g. the cat's feet and stomach are grouped together, and the flamingo's head and neck are in the same cluster. (c) shows the pooling areas computed by PointNet++ [15] grouping layers. PointNet++ uses furthest point sampling and KNN (k-nearest neighbors) to compute clusters, which could lead to unbalanced and unreasonable groups. (d) are clusters where coarse-to-fine hierarchies are enforced to have inclusion relationships. This strategy also results in an unbalanced distribution.

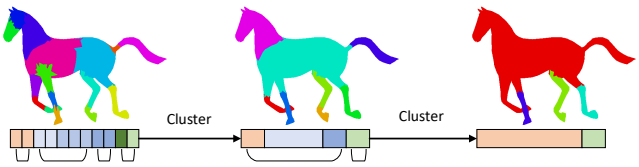


Fig. 4. Inclusion relationship across the hierarchy. An intuitive idea is to perform k-means based on the previous clusters, ensuring that coarser areas consist of multiple finer areas. However, such clustering strategy would result in an unbalanced distribution at coarse levels.

do not have as a good structure as it would have on the original points. Moreover, this kind of consecutive clustering may result in pooling with unbalanced distributions as we can see in Figure 3. Similar problems also exist in the grouping layers (c) of PointNet++ that the coarse groups might be substantially unbalanced. As a result, our method outperforms alternative strategies.

4.3 Robustness Evaluation

In this section, we evaluate the robustness of our network. We will use meshes of different triangulations and numbers of vertices as input to test the performance of the proposed network.

A difficulty when handling mesh data is the varying mesh connectivity. As mentioned before, our method is robust under different mesh triangulations as a result of the clustering strategy. The Laplacian eigenfunction is induced from geodesic distances and therefore invariant under isometric transformation, so the pooling areas as well as input features can essentially stay unchanged when we remesh the object. We visualize the connectivity of the objects before and after remeshing in Fig. 5, which is obtained by subdivision followed by mesh decimation [45] to generate irregular connectivity. Moreover, the quantitative results

(Ours-remesh) in Tab. 1 show that our network is robust under different triangulation.

In addition, our network does not rely on the same vertex numbers for models. An experiment is performed to test this. In this experiment, we first simplify COSEG models to 1500, 2000 and 2500 vertices. Then we split the training and test set according to the same strategy for all three resolutions. We train our network on a mix of two of the datasets and test models from the third (Ours-1500, Ours-2000 and Ours-2500 in the table). Accuracies in Tab. 1 show that our network works well with varying numbers of vertices, compared to the last row where the network is trained with models all containing 2048 vertices.

4.4 Part Segmentation

In this section, we use MFABs to conduct part segmentation on the ShapeNet [35], [46], Human Body Segmentation [23], and COSEG [36] datasets. The application network for segmentation is illustrated in Fig. 2 (top row). Its input is the category label and features from the previous MFAB. The output is a softmax score for each category. The application network has two perceptron layers, a maxpooling layer and two fully connected layers. We minimize the cross-entropy loss between the one-hot vector of ground truth and the network output.

ShapeNet is a large repository of shapes from multiple categories. To leverage this dataset to perform segmentation, Yi et al. [35] develop an active learning method for efficient annotation. However, their annotations are not directly on the mesh vertices, but on the point cloud resampled from the shapes. To recover the graph structure of manifold surfaces for computation of mesh Laplacian and segmentation, we apply [47] to the original ShapeNet models. After that, we transfer the annotations on the point clouds to the nearest mesh vertices.

Two metrics are used in previous segmentation results on ShapeNet, namely accuracy and IoU (Intersection-over-union). We compute both of them to compare with state-of-the-art deep learning methods on 3D shapes [12], [40], and some other methods performing segmentation on ShapeNet based on point clouds such as [48]. Tab. 2 shows that our method achieves the highest average accuracy, and outperforms state-of-the-art methods on 10 out of 16 categories. In terms of IoU, our performance is comparable to the state-of-the-art, achieving the best performance in 8 categories. Some segmentation results are presented in Fig. 6.

COSEG [36] dataset is also a commonly used benchmark for shape segmentation. Compared to ShapeNet, COSEG is much smaller. It has 8 scanned categories and 3 synthetic categories. Each of the 8 categories has around 20 models, which are too few for deep learning. The 3 synthetic categories each have 900 models, so we test our algorithm with the synthetic categories. We compare our result with [49] and [39] in Tab. 1. Our approach outperforms both of them.

Human Body Segmentation is a watertight human mesh dataset proposed by Maron et al. [23]. It has 370 training models from SCAPE [50], MIT [51], FAUST [52], Adobe Fuse [53], and 18 testing models from SHREC07 [54]. We compare with MeshCNN [25], Toric Cover [23], GCNN [39], Dynamic Graph CNN [19], and MDGCNN [55]. Our method outperforms previous methods in this segmentation task. We also perform an ablation study of different clustering strategies on the dataset. Compared to pooling in Euclidean clusters, pooling in the clusters with inclusion relationships, and pooling using the PointNet++ strategy, our design has the best performance.

TABLE 1
Segmentation accuracy on COSEG. We compare with [49] and [39] in the first two rows. In the last row, out network is trained on models with 2048 vertices. To test the robustness on different vertex, we simplify COSEG models to 1500, 2000 and 2500 vertices respectively. We train three networks on two of the three datasets but test on the third. Ours-1500, Ours-2000 and Ours-2500 show the accuracy when the test set has 1500, 2000 and 2500 vertices. We observe that our network performs similarly well with different vertex numbers. Stable performance is also obtained when applying our method to remeshed models with more irregular connectivity (Ours-remesh). The ablation test on the features shows that all three kinds of features contribute to the performance. We also vary the number of MFABs and find that using two MFABs performs best. In general, our method achieves state-of-the-art results in all three categories.

	Chair	Vase	Tele-alien
Xie et al. [49]	87.1%	85.9%	83.2%
Wang et al. [39]	**95.9**%	91.2%	93.0%
Ours-noMFAB	76.6%	77.8%	80.7%
Ours-1MFAB	85.3%	86.1%	88.9%
Ours-3MFABs	90.1%	**92.2**%	91.6%
Ours-noCorrNet	86.9%	85.6%	90.7%
Ours-1500	90.3%	90.0%	89.0%
Ours-2000	90.9%	91.6%	89.3%
Ours-2500	87.0%	86.6%	88.5%
Ours-remesh	92.1%	91.5%	91.8%
Ours-noCoordinates	90.6%	88.6%	84.2%
Ours-noNormal	87.6%	86.1%	85.0%
Ours-noLaplacian	79.6%	87.1%	86.1%
Ours	94.2%	**92.2**%	**93.9**%

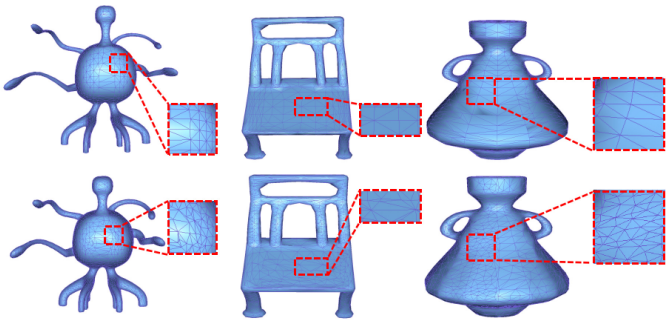


Fig. 5. Robustness to changing connectivity. In this experiment we change the connectivity of meshes from the COSEG datasets, and test whether our network can consistently perform well. The first row shows the original objects, while the meshes in the second row are remeshed into more irregular triangulation. As shown in Table 1, the segmentation accuracy still remains high.

4.5 Mesh Classification

In this section, we evaluate the accuracy of object categorization. The input features to this task are the same as the segmentation task. The application-specific network following Mesh Feature Aggregation Blocks is shown in Fig. 2 (bottom row). It contains two perceptron layers, a maxpooling layer and two fully connected layers. In this case the output is a softmax score for each category. We

TABLE 2
Accuracy and IoU of different methods on ShapeNet. For the task of 3D shape segmentation, we compare our method with Shapeboost [26], Guo [41], and ShapePFCN [40] using the accuracy metric. For FeaStNet [48], ACNN [8], Yi [12], and VoxelCNN [12] we compare with the IoU (Intersection-over-union) metric. Our network achieves highest accuracy.

Method	mean	plane	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor bike	mug	pistol	rocket	skate board	table
Shapeboost (acc)	77.2	85.8	93.1	85.9	79.5	70.1	81.4	89.0	81.2	71.1	86.1	77.2	94.9	88.2	79.2	91.0	74.5
Guo (acc)	77.6	87.4	91.0	85.7	80.1	66.8	79.8	89.9	77.1	71.6	82.7	80.1	95.1	84.1	76.9	89.6	77.8
ShapePFCN (acc)	85.7	**90.3**	**94.6**	**94.5**	90.2	82.9	**84.9**	91.8	82.8	78.0	95.3	**87.0**	96.0	91.5	81.6	91.9	84.8
ours (acc)	**91.5**	89.6	90.2	88.2	88.2	**83.2**	82.3	**95.6**	**88.7**	**87.4**	**96.3**	70.6	**97.0**	**92.7**	**82.2**	**94.7**	**92.6**
FeaStNet (IoU)	81.5	79.3	74.2	69.9	71.7	87.5	64.2	90.0	80.1	78.7	94.7	62.4	91.8	78.3	48.1	71.6	79.6
ACNN (IoU)	79.6	76.4	72.9	70.8	72.7	86.1	71.1	87.8	82.0	77.4	95.5	45.7	89.5	77.4	49.2	82.1	76.7
VoxelCNN (IoU)	79.4	75.1	72.8	73.3	70.0	87.2	63.5	88.4	79.6	74.4	93.5	58.7	91.8	76.4	51.2	65.3	77.1
Yi (IoU)	**84.7**	81.6	81.7	**81.9**	75.2	**90.2**	**74.9**	**93.0**	**86.1**	**84.7**	**95.6**	**66.7**	92.7	81.6	62.1	82.9	82.1
ours (IoU)	84.3	**82.9**	**83.4**	81.7	**80.0**	75.4	71.8	91.9	81.0	80.9	92.5	59.2	**93.5**	**86.3**	**74.3**	**90.3**	**86.4**

TABLE 3
Mesh segmentation accuracy on Human Body Segmentation dataset. Our method achieves the highest accuracy when compared with MeshCNN [25], Toric Cover [23], GCNN [39], Dynamic Graph CNN [19], and MDGCNN [55]. The ablation tests use different clustering strategies as follows. We run k-means on Euclidean coordinates (Ours-Euc), enforce clusters to have inclusion relationship (Ours-Inc), and use grouping layers from PointNet++ for pooling (Ours-Group), while keeping the remaining pipeline of our method. Results show that our original clustering strategy has the best performance.

Method	Accuracy	Method	Accuracy
MeshCNN	92.30%	DynGraphCNN	89.72
Toric Cover	88.00%	GCNN	86.40%
PointNet++	90.77%	MDGCNN	89.47%
Ours	92.58%	Ours-Euc	90.30%
Ours-Inc	90.41%	Ours-Group	89.02%

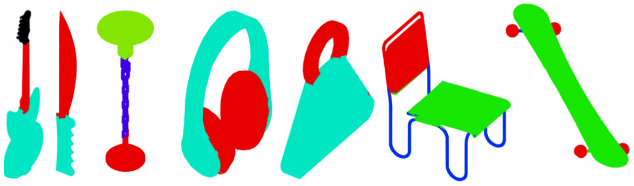


Fig. 6. Qualitative results on ShapeNet. The segmentation results produced by our method are plausible.

minimize the cross-entropy loss between one-hot vector of ground truth and network output.

For this classification task, we compare with other mesh-based approaches on the ModelNet [5], ShapeNet, and SHREC11 [61] datasets. There are 30 categories of watertight meshes in SHREC11, each having 20 models. Split 16 and Split 10 refer to using 16 and 10 models in each category for training. Table 4 and Table 5 show that our method is comparable with state-of-the-art methods.

4.6 Failure Cases

We would like to restate that our method does not work directly on the original ShapeNet models for two reasons: 1) the annotations are labeled on point clouds uniformly sampled from shapes, instead of vertices of the mesh; 2) meshes in ShapeNet are not manifold meshes, preventing us from performing high-quality spectral clustering. Therefore

TABLE 4
Classification accuracy on ModelNet10, ModelNet40 and ShapeNet. Distinguished by input representations, SPH [56], SyncSpecCNN [12], ACNN [8] and FoldingNet [8] use meshes; PointNet [14], PointNet++ [14] and SO-Net [57] use point clouds; Voxelnet [58] and 3DShapeNets [5] take voxels as input. '-' in the table indicates the performance is not reported. For the classification accuracy on ShapeNet, our network has comparable performance to state-of-the-art methods. Our method outperforms all those single model classification methods.

Method	input	MN10	MN40	ShapeNet
PointNet	point	-	89.2%	-
PointNet++	point	-	91.9%	-
SO-Net	point	95.7%	93.4%	-
3DShapeNets	volume	83.5%	77.0%	-
Voxnet	volume	91.0%	84.5%	-
ACNN	mesh	-	-	93.99%
SyncSpecCNN	mesh	-	-	99.71%
SPH	mesh	-	68.2%	-
Ours	mesh	**97.4%**	**94.21%**	**99.88%**

TABLE 5
Classification on the 30 classes of SHREC11. Split 16 and Split 10 are different training/testing splits, where 16 and 10 models are used for training for each category, respectively. Compared with MeshCNN [25], GWCNN [22], Shape Google (SG) [59], 3D ShapeNets (SN) [5] and Geometry Images (GI) [60], our method achieves a similar performance as state of the art methods.

Method	Split 16	Split 10
MeshCNN	98.6%	91.0%
GWCNN	96.6%	90.3%
GI	96.6%	88.6%
SN	48.4%	52.7%
SG	70.4%	62.6%
Ours	98.0%	90.3%

we convert shapes into watertight manifold surfaces using [47], simplify the mesh to a reasonable level, and transfer part labels to vertices from the point cloud through nearest neighbor matching. Error accumulates during this process. In Fig. 7 we show some failure cases when performing part segmentation in ShapeNet. There are several typical problems for models that are poorly segmented. In the chair example, we can observe sharp edges in ground truth labels, and such artifacts could be caused by the noise in the original point cloud. This kind of noise makes learning reasonable segmentation more challenging. As in the bag

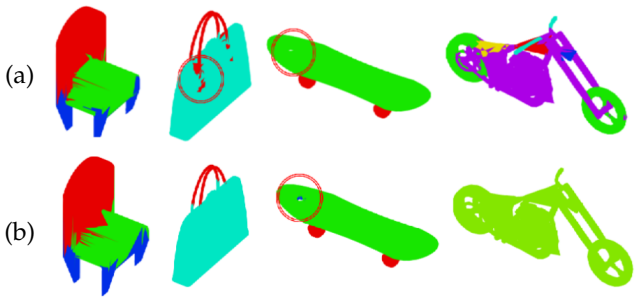


Fig. 7. Failure cases in ShapeNet segmentation. We show poor segmentation results in four categories in ShapeNet. (a) is the ground truth and (b) is our prediction. Those imperfect results are not necessarily caused by our segmentation method but also imperfect labeling in the dataset and error accumulation during preprocessing. The following examples show the main challenges. The ground truth segmentation mask of the chair is not smooth, which shows the limitations of the annotation. Some regions of segmentation annotation on the bag are incorrect. These kind of errors can be caused during transferring the mesh onto a watertight manifold. Our method also suffers from lacking training examples and fails to produce correct results on the motorbike. The skateboard has a hole in the mesh which leads to poor segmentation around it.

case, we can observe in the red circle that the ground truth is incorrect. This may be caused by changes of the shape while using [47] to make the model watertight, so that the nearest neighbor algorithm gets the wrong correspondence when transferring labels. We can see that our algorithm actually gets a more correct answer than the "ground truth". Failure in the motorbike case is caused by lack of training data. In the annotations, the motorbike class has the fewest data samples. Worse still, some of the training examples fail to be converted when performing [47], because motorbikes have complex topological structure, making transfer challenging. In practice, it can be a big problem that errors in the transfer and simplification will decrease the amount of training data. Last but not least, the skateboard shows that there might be some holes, or other artifacts, in the surface, that could affect graph structure and mislead our algorithm.

5 Conclusion

In this paper, we present a deep learning approach to predicting functions defined on shapes. The key idea is to perform multiscale pooling based on Laplacian spectral clustering, and use a Correlation Net following pooling to fuse global information. Compared to the pooling operation in the previous literature, our network does not require a uniform number of vertices in each model. Our work outperforms state-of-the-art methods in most categories for shape classification and segmentation.

Our method may be applied to other tasks. For example, 3D reconstruction is fundamental but challenging. A capable and general method to generate meshes is of great demand. Our network has the potential to achieve this task, because it can neatly encode connectivity in vertices, and intrinsically understands the topology through spectral clustering and spatial pooling. Furthermore, as a general structure for mesh processing, our network may also be applied to shape deformation, completion and correspondence.

Finally it would be interesting to see how our network can work on general graphs. In this paper, we mainly deal with manifold meshes, using geodesic distances to construct Laplacian. However in other problem settings, we can use any distance metric that can best describe the problem. For example, we might experiment with our method on social networks with arbitrary size and connectivity.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61872440 and No. 61828204), Beijing Municipal Natural Science Foundation (No. L182016), Royal Society Newton Advanced Fellowship (No. NAF\R2\192151), Youth Innovation Promotion Association CAS, CCF-Tencent Open Fund and Open Project Program of the National Laboratory of Pattern Recognition (No. 201900055).

References

[1] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[2] M. Chen, G. Ding, S. Zhao, H. Chen, Q. Liu, and J. Han, "Reference based LSTM for image captioning." in *AAAI*, 2017, pp. 3981–3987.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[4] S. Kwak, S. Hong, B. Han *et al.*, "Weakly supervised semantic segmentation using superpixel pooling network." in *AAAI*, 2017, pp. 4111–4117.

[5] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d ShapeNets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[7] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on Riemannian manifolds," in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 37–45.

[8] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 3189–3197.

[9] K. Crane, C. Weischedel, and M. Wardetzky, "The heat method for distance computation," *Commun. ACM*, vol. 60, no. 11, pp. 90–99, Oct. 2017. [Online]. Available: http://doi.acm.org/10.1145/3131280

[10] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[11] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," *arXiv preprint arXiv:1801.03226*, 2018.

[12] L. Yi, H. Su, X. Guo, and L. J. Guibas, "SyncSpecCNN: Synchronized spectral CNN for 3d shape segmentation." in *CVPR*, 2017, pp. 6584–6592.

[13] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3d data," in *CVPR*, 2016, pp. 5648–5656.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.

[16] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[17] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[18] Q. Tan, L. Gao, Y.-K. Lai, J. Yang, and S. Xia, "Mesh-based autoencoders for localized deformation component analysis," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[19] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, pp. 1–12, 2019.

[20] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and J. Bruna, "Surface networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2540–2548.

[21] R. Song, Y. Liu, and P. L. Rosin, "Mesh saliency via weakly supervised classification-for-saliency CNN," *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[22] D. Ezuz, J. Solomon, V. G. Kim, and M. Ben-Chen, "GWCNN: A metric alignment layer for deep shape analysis," in *Computer Graphics Forum*, vol. 36, no. 5. Wiley Online Library, 2017, pp. 49–57.

[23] H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, and Y. Lipman, "Convolutional neural networks on surfaces via seamless toric covers." *ACM Trans. Graph.*, vol. 36, no. 4, pp. 71–1, 2017.

[24] N. Haim, N. Segol, H. Ben-Hamu, H. Maron, and Y. Lipman, "Surface networks via general covers," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 632–641.

[25] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: a network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[26] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3d mesh segmentation and labeling," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 102, 2010.

[27] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Computer graphics forum*, vol. 28, no. 5. Wiley Online Library, 2009, pp. 1383–1392.

[28] M. Aubry, U. Schlickewei, and D. Cremers, "The wave kernel signature: A quantum mechanical approach to shape analysis," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1626–1633.

[29] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[30] Z. Wu, Y. Wang, R. Shou, B. Chen, and X. Liu, "Unsupervised co-segmentation of 3d shapes via affinity aggregation spectral clustering," *Computers & Graphics*, vol. 37, no. 6, pp. 628–637, 2013.

[31] Z. Shu, C. Qi, S. Xin, C. Hu, L. Wang, Y. Zhang, and L. Liu, "Unsupervised 3d shape segmentation and co-segmentation via deep learning," *Computer Aided Geometric Design*, vol. 43, pp. 39–52, 2016.

[32] Q. Huang, V. Koltun, and L. Guibas, "Joint shape segmentation with linear programming," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6, p. 125, 2011.

[33] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or, *Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering*. ACM, 2011, vol. 30, no. 6.

[34] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3d mesh segmentation," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, p. 73, 2009.

[35] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, L. Guibas *et al.*, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 210, 2016.

[36] Y. Wang, S. Asafi, O. Van Kaick, H. Zhang, D. Cohen-Or, and B. Chen, "Active co-analysis of a set of shapes," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 165, 2012.

[37] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton shape benchmark," in *Shape modeling applications, 2004. Proceedings*. IEEE, 2004, pp. 167–178.

[38] D. George, X. Xie, and G. K. Tam, "3d mesh segmentation via multi-branch 1d convolutional neural networks," *Graphical Models*, vol. 96, pp. 1–10, 2018.

[39] P. Wang, Y. Gan, P. Shui, F. Yu, Y. Zhang, S. Chen, and Z. Sun, "3d shape segmentation via shape fully convolutional networks," *Computers & Graphics*, vol. 70, pp. 128–139, 2018.

[40] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3d shape segmentation with projective convolutional networks," in *Proc. CVPR*, vol. 1, no. 2, 2017, p. 8.

[41] K. Guo, D. Zou, and X. Chen, "3d mesh labeling via deep convolutional neural networks," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 1, p. 3, 2015.

[42] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *Visualization and Mathematics III*, 2003, pp. 35–57.

[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[45] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 209–216. [Online]. Available: https://doi.org/10.1145/258734.258849

[46] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "ShapeNet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[47] J. Huang, H. Su, and L. Guibas, "Robust watertight manifold surface generation method for ShapeNet models," *arXiv preprint arXiv:1802.01698*, 2018.

[48] N. Verma, E. Boyer, and J. Verbeek, "FeaStNet: Feature-steered graph convolutions for 3d shape analysis," in *CVPR 2018-IEEE Conference on Computer Vision & Pattern Recognition*, 2018.

[49] Z. Xie, K. Xu, L. Liu, and Y. Xiong, "3d shape segmentation and labeling via extreme learning machine," in *Computer graphics forum*, vol. 33, no. 5. Wiley Online Library, 2014, pp. 85–95.

[50] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," in *ACM SIGGRAPH 2005 Papers*, 2005, pp. 408–416.

[51] D. Vlasic, I. Baran, W. Matusik, and J. Popović, "Articulated mesh animation from multi-view silhouettes," in *ACM SIGGRAPH 2008 papers*, 2008, pp. 1–9.

[52] F. Bogo, J. Romero, M. Loper, and M. J. Black, "Faust: Dataset and evaluation for 3d mesh registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3794–3801.

[53] Adobe, "Adobe fuse 3d charactersadobe fuse 3d characters," in *https://www.adobe.com/products/fuse.html*, 2016.

[54] D. Giorgi, S. Biasotti, and L. Paraboschi, "SHREC '07 Track: watertight models," in *Shape Modeling International*, 2007.

[55] A. Poulenard and M. Ovsjanikov, "Multi-directional geodesic neural networks via equivariant convolution," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–14, 2018.

[56] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation invariant spherical harmonic representation of 3d shape descriptors," in *Symposium on geometry processing*, vol. 6, 2003, pp. 156–164.

[57] J. Li, B. M. Chen, and G. H. Lee, "SO-Net: Self-organizing network for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9397–9406.

[58] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 922–928.

[59] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov, "Shape google: Geometric words and expressions for invariant shape retrieval," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 1, pp. 1–20, 2011.

[60] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3d shape surfaces using geometry images," in *European Conference on Computer Vision*. Springer, 2016, pp. 223–240.

[61] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens,

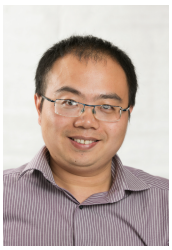
H. Tabia, and D. Vandermeulen, "SHREC '11 Track: shape retrieval on non-rigid 3d watertight meshes," in *Eurographics Workshop on 3D Object Retrieval (3DOR)*, 2011.

Yi-Ling Qiao received a bachelor's degree in computer science and technology from the University of Chinese Academy of Sciences in 2019. He is currently a PhD student in computer science at University of Maryland, College Park. His research interests include computer graphics and geometric processing.

Lin Gao received a bachelor's degree in mathematics from Sichuan University and a PhD degree in computer science from Tsinghua University. He is currently an Associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences. He has been awarded a Newton Advanced Fellowship from the Royal Society. His research interests include computer graphics and geometric processing.

Jie Yang received a bachelor's degree in mathematics from Sichuan University in 2016. He is currently a PhD candidate in the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer graphics and geometric processing.

Paul L. Rosin is a Professor at the School of Computer Science & Informatics, Cardiff University. Previous posts were at Brunel University, Joint Research Centre (Italy), and Curtin University of Technology (Australia). His research interests include low level image processing, performance evaluation, shape analysis, facial analysis, medical image analysis, 3D mesh processing, cellular automata, non-photorealistic rendering and cultural heritage.



Yu-Kun Lai received his bachelor's degree and PhD degree in computer science from Tsinghua University in 2003 and 2008, respectively. He is currently a Professor in the School of Computer Science & Informatics, Cardiff University. His research interests include computer graphics, geometry processing, image processing and computer vision. He is on the editorial boards of *Computer Graphics Forum* and *The Visual Computer*.

Xilin Chen received the BS, MS, and PhD degrees in Computer Science from Harbin Institute of Technology, China, in 1988, 1991, and 1994 respectively. Then he joined the Department of Computer Science and Engineering, Harbin Institute of Technology, where he was a lecturer (1994), associate professor (1996), and professor (1999-2005) . He was a visiting scholar with Carnegie Mellon University from 2001 to 2004. He was elected into the 100 Talents Program of Chinese Academy of Sciences (CAS) and joined the Institute of Computing Technology, CAS in August, 2004. His research interests are Image Understanding, Computer Vision, Pattern Recognition, Image Processing, Multimodal Interface, and Digital Video Broadcasting.