Laplacian2Mesh: Laplacian-Based Mesh Understanding

Qiujie Dong
Shandong University
Qingdao, Shandong
qiujie.jay.dong@gmail.com

Zixiong Wang
Shandong University
Qingdao, Shandong
zixiong_wang@outlook.com

Manyi Li
Shandong University
Jinan, Shandong
manyili@sdu.edu.cn

Junjie Gao
Shandong University
Qingdao, Shandong
gjjsdnu@163.com

Shuangmin Chen
Qingdao University of Science and Technology
Qingdao, Shandong
csmqq@163.com

Zhenyu Shu
Ningbo Institute of Technology, Zhejiang University
Ningbo, Zhejiang
shuzhenyu@nit.zju.edu.cn

Shiqing Xin*
Shandong University
Qingdao, Shandong
xinshiqing@sdu.edu.cn

Changhe Tu
Shandong University
Qingdao, Shandong
chtu@sdu.edu.cn

Wenping Wang
Texas A&M University
Texas, USA
wenping@cs.hku.hk

Abstract

Geometric deep learning has sparked a rising interest in computer graphics to perform shape understanding tasks, such as shape classification and semantic segmentation. When the input is a polygonal surface, one has to suffer from the irregular mesh structure. Motivated by the geometric spectral theory, we introduce *Laplacian2Mesh*, a novel and flexible convolutional neural network (CNN) framework for coping with irregular triangle meshes (vertices may have any valence). By mapping the input mesh surface to the multi-dimensional Laplacian-Beltrami space, Laplacian2Mesh enables one to perform shape analysis tasks directly using the mature CNNs, without the need to deal with the irregular connectivity of the mesh structure. We further define a mesh pooling operation such that the receptive field of the network can be expanded while retaining the original vertex set as well as the connections between them. Besides, we introduce a channel-wise self-attention block to learn the individual importance of feature ingredients. Laplacian2Mesh not only decouples the geometry from the irregular connectivity of the mesh structure but also better captures the global features that are central to shape classification and segmentation. Extensive tests on various datasets demonstrate the effectiveness and efficiency of Laplacian2Mesh, particularly in terms of the capability of being vulnerable to noise to fulfill various learning tasks.

Keywords: ***Geometric Deep Learning, Mesh Understanding, Laplacian-Beltrami space, Laplacian Pooling.***

1. Introduction

The rapidly emerging 3D geometric learning techniques have achieved impressive performance in various applications, such as classification [61, 70], semantic segmentation [20, 63], and shape reconstruction [5, 40]. A large number of deep neural networks have been designed to deal with 3D shapes of various representations, including voxels [30, 67], multi-view images [33, 61], point clouds [49, 50], meshes [20, 31], etc. Among all the rep-

resentations, polygonal surfaces, as one of the most popular shape representations, are flexible in characterizing an arbitrarily complex 3D shape in an unambiguous manner. However, the irregular connections between vertices make designing learning-based methods extremely difficult. To be more specific, the challenges include the following aspects: First, polygonal surfaces that represent the same shape may have varying numbers of vertices and faces. Second, the valence of a vertex is not fixed, making it hard to define a regular structure as the receptive field for the conventional convolution operator. Finally, even if the number of vertices is given, various triangulation forms may exist while the difference in representation accuracy is neglectable.

In the research community, there have been many attempts for developing mesh-specific convolution and pooling/unpooling operations. For example, GeodesicCNN [43] and CurvaNet [23] are proposed to define convolution kernels on the parameterization plane for each local surface patch. MeshCNN [20] and PD-MeshNet [45] take a mesh as a graph with regard to the vertex set. They introduce edge-contraction-based pooling/unpooling operations to deal with the mesh-based deep learning tasks. Very recently, SubdivNet [25] suggests a convolutional operation by referring to the coarse-to-fine Loop subdivision algorithm. All the above-mentioned approaches have to invent an artificial convolutional scheme and explicitly come into contact with the irregular triangulation, imposing harsh requirements on the input mesh and suffering from various mesh imperfections.

In this paper, we propose *Laplacian2Mesh*, inheriting the spirit of those spectral approaches, to make it possible to conduct learning tasks in the spectral domain when the input is a polygonal mesh. Note that the main technique of spectral approaches is to encode the overall shape by a subset of the eigenvectors decomposed from the Laplacian matrix of the input mesh. Two reasons account for why we advocate using the new representation in deep learning. First, as the low-frequency signals, given by those eigenvectors with small eigenvalues, encode the overall shape, they are more semantically important than high-frequency signals for most understanding tasks. It is easy for one to separate low-frequency signals from high-frequency signals in spectral approaches by setting a simple parameter k. Second, the Laplacian-based spectral transform can decouple shape variations from the tedious triangulation, avoiding tangling with tedious and irregular triangulations. Even if the input mesh contains a limited number of defects (e.g., non-manifold), the representation still works.

Some researchers have turned their attention to this promising research area. For example, the recent DiffusionNet [57] took the vertex positions and the spatial gradients as the original signal and then projected them w.r.t. the Laplacian-based spectral basis. It is pointed out in this work that directly learning the frequency signals suffers from the different eigenvectors decomposed from different shapes. They merely use the spectral transform to facilitate the spectral acceleration and project the spectral features back to the spatial domain, yielding the vertex-wise features,which means that it is not a spectral learning method but just a numerical scheme to compute diffusion.

Therefore, there is still a long way to go to develop a deep network for the spectral signals of the polygonal meshes. The first difficulty lies in that different shapes have different eigenvectors [21, 57], resulting in a learned spectral convolution filter on one shape that cannot generalize to a different shape. The second difficulty is how to bring this kind of network to its full potential, i.e., learning the useful features for understanding a shape. Potentially helpful network architecture is the famous U-Net [53] with the encoder part and the decoder part, where the encoder consists of the convolutional layers followed by pooling operation, and the decoder, also consisting of multiple connected layers, uses transposed convolution to permit localization.

Our Laplacian2Mesh is designed to deal with the above-mentioned two difficulties by transforming the features of the polygonal meshes into eigenspace [36]. On the one hand, we select three groups of eigenvectors to construct the multi-resolution bases. We use the Squeeze-and-Excitation ResNet (SE-ResNet) [24] as the basic unit to define the feature extractor, as well as the carefully designed Laplacian pooling/unpooling operations to fuse the resulting features with different dimensions. Additionally, we consider the shape descriptors (vertex normals, dihedral angles of vertices, Gaussian curvature, Laplacian eigenvectors, heat kernel signature (HKS) [62]) and the pose descriptor (vertex coordinates) at the same time to feed the network so that useful shape information can be fully utilized.

We conduct comprehensive ablation/comparison experiments to validate the effectiveness of our approach. Tests on various shape understanding tasks, including shape classification and semantic segmentation, show that our algorithm outperforms the state-of-the-art on average. The biggest advantage of our approach lies in the robustness to noise. Besides, our approach can tolerate a limited number of mesh defects.

2. Related work

In this section, we first briefly introduce the related works on the 3D geometric learning [8] with various shape representations. We refer the readers to [9, 71] for a more comprehensive survey. After that, we review the learning-based mesh understanding methods, which are highly relevant to the theme of this paper. Finally, we talk about the spectral-based technique and its applications.

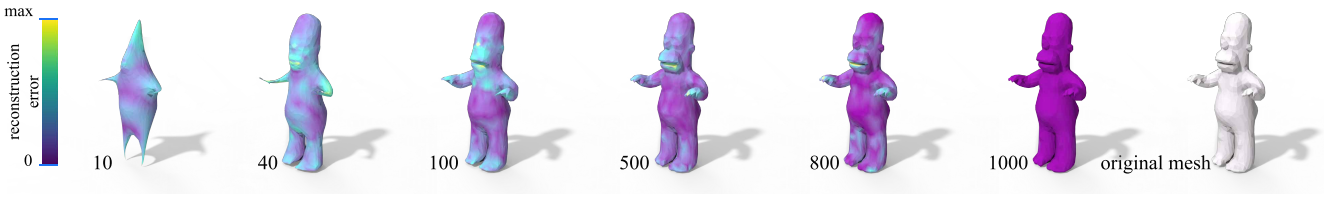


Figure 1. The spectral surface reconstruction when selecting different numbers of Laplacian eigenvectors as the spectral basis. From left to the right, the fewer smallest eigenvectors used, the less high-frequency features preserved in the reconstruction. Therefore, by selecting the proper basis, the spectral transforms not only encode the irregular mesh structure into fix-sized signals but also are resistant to the high-frequency noise for the mesh understanding tasks.

2.1. 3D Geometric Deep Learning

The design of geometric deep learning networks relies heavily on the shape representations. Wu et al. [70] proposed a 3D convolutional network based on the voxel representation for the shape classification and retrieval tasks. Due to their regular data structure, 3D voxels are easily processed by different 3D convolutional networks, making the voxel representation frequently utilized in the applications such as shape reconstruction [5], semantic segmentation [63], and alignment [19]. Its usage is greatly limited by the cubically increasing complexity in the memory and computation cost. To deal with this issue, the octree-based volumetric representation [52, 66] is more compact and has received more attention recently. Through parameterization, a surface can be encoded by an image-like data structure, e.g. the geometry images [59] and the toric covers [18, 42]. Another way is to render the 3D shapes as multi-view images, so that 2D CNN can be directly used for the understanding tasks [33, 61]. However, it may fail for a complicated shape with a highly curved surface.

Point cloud is also a popular representation in 3D geometric learning. For example, PointNet [49] and PointNet++ [50] use the KNN search to build the spatial correlations between points for feature extraction. Some following works, such as PointCNN [37] and KPConv [64], further improve feature learning from point cloud. Recently, PCT [17] applies the attention-based transformer to deal with point cloud, achieving state-of-the-art performance. However, for a point cloud, the direct exploitation of the geometry is challenging due to inherent obstacles such as noise, occlusions, sparsity or variance in the density. On the other hand, polygonal surfaces typically offer more detailed geometry and topology descriptions, hopefully improving the understanding of behavior.

2.2. Learning-based Mesh Understanding Methods

The mesh representation is often considered as a graph structure. MeshWalker [31] explores the mesh geometry and topology via random walks. The Recurrent Neural Network (RNN) is used to learn the deep features along each walk. Some other methods construct the local relationship between the vertices by considering the K-ring neighborhood. Masci et al. [43] proposed to flatten each of the patches with small curvature to the 2D plane, and apply the geodesic convolutional neural network on the non-Euclidean manifolds. He et al. [23] defined a directional curvature along with rotation in the tangent plane to obtain neighbor information with a consistent structure. These methods focus on defining the localized convolution kernels. Some similar strategies include local spectral filters [4] and local geometric descriptors (e.g. B-splines [15], wavelets [55] and extrinsic Euclidean convolution [56]) to align a local patch with the convolution kernel.

Some recent works leverage the classical mesh processing techniques to define the key modules of a geometric deep learning network, such as the convolutional operation and the pooling operation. For example, MeshCNN [20] defines the receptive field for the convolutional operation, based on the observation that each edge is correlated with two neighbor triangular faces. Furthermore, MeshCNN defines the unique pooling operation based on the edge-collapse algorithm. The PD-MeshNet [45] constructs the primal graph and the dual graph to capture the adjacency information encoded in the triangle mesh. The pooling operation of PD-MeshNet is implemented based on mesh simplification. Benefiting from the up-sampling process of the Loop subdivision algorithm, SubdivNet [25] proposes a mesh convolution operator that allows one to aggregate local features from nearby faces. But it requires the input to hold the subdivision connectivity. All the aforementioned methods focus on the spatial structure and have to handle the irregular connections.

A very recent work DiffusionNet [57] uses the spectral acceleration technique to diffuse the vertex features, showing a great potential of the spectral analysis in mesh-based learning tasks. We propose to extend the backbone from the spatial to the spectral domain, and further improve the performance of shape understanding, particularly in the presence of noise.

2.3. Spectral Surface Representation

As pointed out in [9], spectral surface representation is important in encoding the shape information with varying frequencies, regardless of the connection structure. In the discrete setting, one can take the eigenvectors of the Laplacian matrix as the basis for the spectral transform. As shown in Figure 1, the low-frequency signals, given by the eigenvectors with the k smallest eigenvalues, can capture the essential shape information, as long as k is large enough. Therefore, by selecting the proper number of eigenvectors, the spectral projection can be seen as a low-pass filter to the mesh feature signals, naturally resisting noise and sampling bias. The spectral analysis plays a central role in many geometry processing and shape analysis tasks, such as shape segmentation [28], symmetry detection [47], shape correspondence [46], shape recognition [7], and shape retrieval [6].

Some existing research works [36, 27] show that it is possible to use the spectral analysis technique to define a compact representation of a polygonal surface. Heat Kernel Signature (HKS) [62] inspires us in that the local-to-global shape variations can be encoded through heat diffusion. Please refer to [69, 36] for a more detailed survey.

There are already some works using spectral signals in deep learning. The spectral-based graph networks, e.g., SpectralCNN[12] and ChebNet [11], convert the graph structure into the spectral domain and perform the multiplication on the spectral signals. LaplacianNet [51] uses the graph Laplacian-based spectral clustering for over-segmentation, followed by performing a max-pooling operation on each class to obtain local information. LaplacianNet includes the Correlation Net as a component to learn a correlation matrix to fuse features across clusters. But it's worth noting that LaplacianNet still focuses on the spatial relation among the local patches.

3. Methodology

We compute a collection of intrinsic and extrinsic mesh features, and transform them into the spectral domain to feed the network. Then the network utilizes the U-Net architecture to handle various mesh understanding tasks. Additionally, we develop a channel-wise attention mechanism employing the small-sized Squeeze-and-Excitation blocks (SE-block) [24], before fusing features of various resolutions using Laplacian pooling/unpooling. The whole network is connected by different head blocks. Finally, different loss functions are proposed to cope with mesh classification and segmentation tasks, respectively.

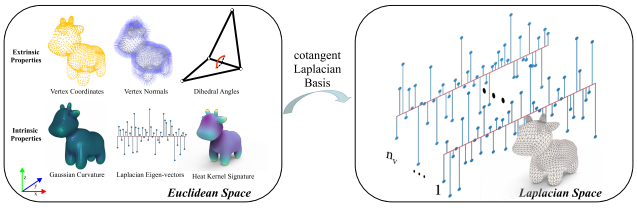


Figure 2. We compute a set of extrinsic and intrinsic geometric features on the mesh surface and convert them into spectral signals as the input of our network.

3.1. Laplacian spectral transform

Given a triangle mesh with n vertices, the Laplacian matrix [44, 54] can be written as:

$$\mathbf{L} = \mathbf{M}^{-1}\mathbf{C}, \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ is the diagonal matrix whose i-th entry along the diagonal is twice the influence area of the vertex v_i, $\mathbf{C}$ is the sparse cotangent weighted matrix as in Eq. (2):

$$\mathbf{C}_{ij} = \begin{cases} -(\cot\alpha_{ij} + \cot\beta_{ij}), & i \neq j, v_j \in r_1(v_i) \\ \sum_{v_j \in r_1(v_i)} (\cot\alpha_{ij} + \cot\beta_{ij}), & i = j \\ 0, & v_j \notin r_1(v_i), \end{cases} \tag{2}$$

where $r_1(v_i)$ is the set of all the 1-ring adjacent vertices of v_i, and α_{ij} and β_{ij} are the two angles opposite to edge $v_i v_j$. Please refer to our Appendices for more details.

The Laplacian matrix completely encodes the intrinsic geometry. The eigendecomposition of the Laplacian matrix $\mathbf{L}$ enables the transformation between the spatial and the spectral domain. Specifically, after performing the eigendecomposition, we can sort and select the k eigenvectors $\mathbf{\Phi}_k = [\boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{k-1}]$ corresponding to the k smallest eigenvalues. It's worth noting that the smallest eigenvalue is 0. $\mathbf{\Phi}_k$ can be understood as a low-frequency filter. Suppose that we define a scalar field f on the surface. It can be decomposed into a combination of the eigenvalues. If we keep only the part spanned by $\mathbf{\Phi}_k$, we obtain the reconstructed counterpart of f.

Figure 1 shows the Laplacian spectral reconstruction with different numbers of eigenvectors as the basis. The mesh vertex set $\mathbf{V}$ is first projected to $\widetilde{\mathbf{V}}$ in the spectral domain, and then reconstructed as $\widehat{\mathbf{V}}$ following Eq. (3):

$$\begin{aligned} \widetilde{\mathbf{V}} &= \mathbf{\Phi}_k^T \mathbf{V}, \\ \widehat{\mathbf{V}} &= \mathbf{\Phi}_k (\mathbf{\Phi}_k^T \mathbf{V}). \end{aligned} \tag{3}$$

Figure 1 also illustrates that the spectral transform can serve as a low-pass filter of the mesh feature signal. Therefore, it prevents the mesh processing from being affected by the high-frequency noise.

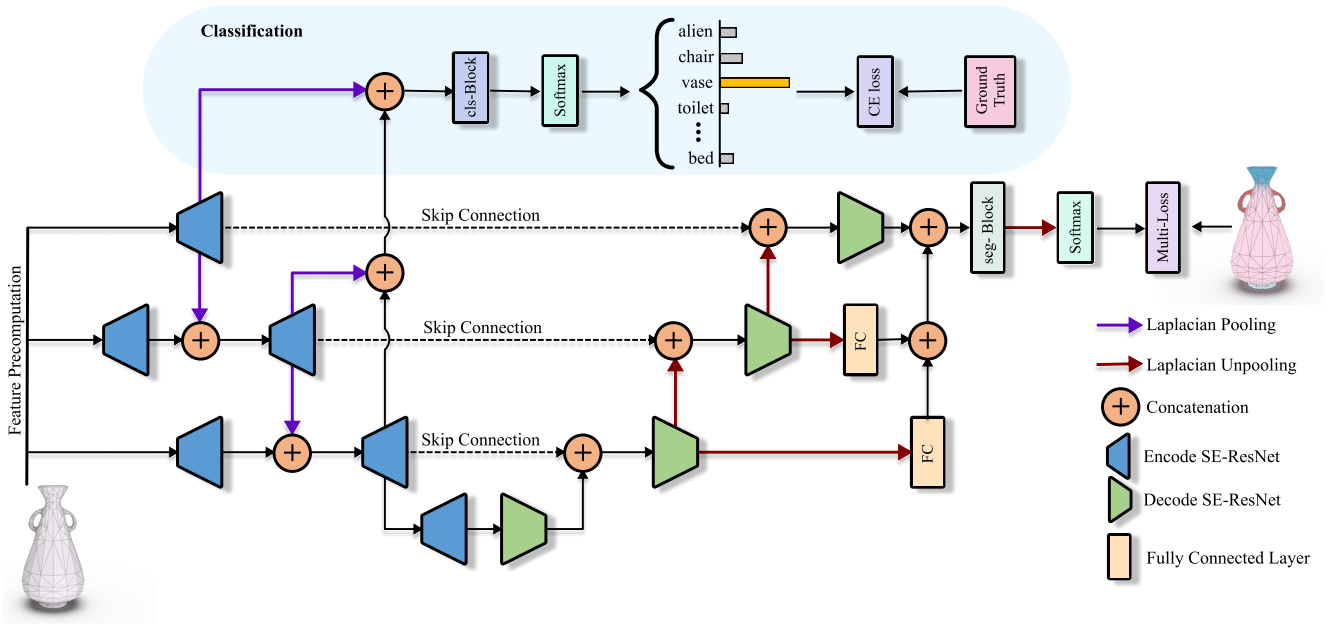


Figure 3. Our network pipeline for coping with the mesh classification and segmentation tasks. Given a 3D mesh as the input, we precompute the extrinsic and intrinsic geometric features and project them into the spectral domain w.r.t. three different resolutions. Inspired by the U-Net architecture, we propose to use the SE-ResNet blocks with small-sized convolution kernels to fuse the nearby-frequency features, and the Laplacian pooling/unpooling to fuse the spectral features of different resolutions. For the segmentation task, we re-scale (with the yellow block) and concatenate the features together to be processed by the segmentation block.

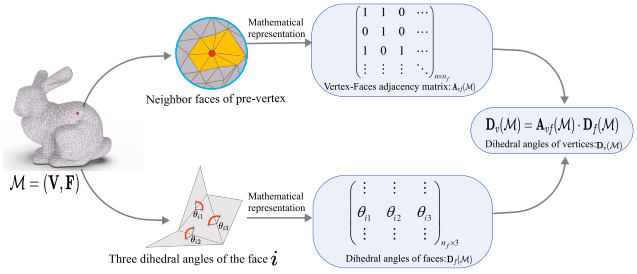


Figure 4. The vertex-wise feature based on the dihedral angles, which are originally defined on the shared edges between faces. We use the vertex-face adjacency matrix to average the three dihedral angles of the one-ring faces to form the vertex-wise dihedral angle.

We take $\mathbf{\Phi}_k$ as the basis to perform the Laplacian spectral transform. To simplify the computation, the eigenvector matrix $\mathbf{\Phi}$ is obtained by performing the eigendecomposition on the cotangent weight matrix instead of the discrete Laplace-Beltrami matrix itself. Please refer to Appendices for more detailed explanation.

3.2. Input Features

In geometric deep learning, a commonly used way is to extract various geometric features to define a synthesized shape descriptor. The intrinsic and extrinsic features characterize the shape from different perspectives, and their combination can describe the shape [20, 25, 51, 57].

Our input feature $\mathbf{G} \in \mathbb{R}^{n \times 39}$ in the spatial domain is composed of the per-vertex features, each of which is a concatenation of a 30-dimensional intrinsic shape descriptor and a 9-dimensional extrinsic shape descriptor (see Figure 2). The intrinsic shape descriptor is formed by the 1-dimensional Gaussian curvature, the 9-dimensional HKS (Heat Kernel Signature) [62], and low-frequency eigenvectors of the cotangent weight matrix corresponding to the 20 lowest frequencies excluding 0. The extrinsic shape descriptor includes the 3-dimensional vertex coordinates, the 3-dimensional vertex normal, and the 3-dimensional dihedral angles for each vertex (we will explain it later). Finally, the input feature vector $\mathbf{G}$, upon being mapped to the Laplacian spectral domain, becomes:

$$\widetilde{\mathbf{G}} = \mathbf{\Phi}_k^T \mathbf{G}. \tag{4}$$

To this end, an arbitrary polygonal surface is transformed into a $k \times 39$ matrix $\widetilde{\mathbf{G}}$, regardless of the geometry, topology, connection structure, and mesh complexity.

It is worth noting that the dihedral angles are originally defined as the angle between two adjacent faces sharing a common edge. We extend the concept to a per-vertex feature by distributing the dihedral angle of a mesh edge to the endpoints and the opposite vertices. Recall that the face $f =$

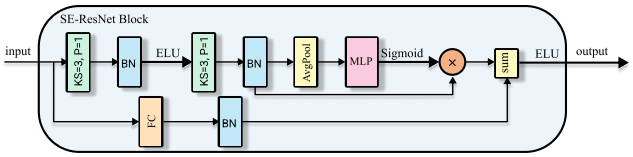


Figure 5. The layers of SE-ResNet Block.

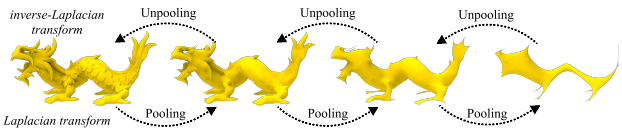


Figure 6. The Laplacian pooling and unpooling operations transform the spectral-based features between different resolutions, where the pooling operation proceeds from a finer level to a coarser level while the unpooling operation does the opposite.

$\triangle v_1v_2v_3$ has three bounding edges $\overrightarrow{v_2v_3}, \overrightarrow{v_3v_1}, \overrightarrow{v_1v_2}$, and they give three dihedral angles $\theta_1, \theta_2, \theta_3$, respectively. In our scenario, we need to define vertex-wise geometric features. For this purpose, we can relate the dihedral angles to the three vertices of f. Particularly, we assign $\theta_1, \theta_2, \theta_3$ to v_1, assign $\theta_2, \theta_3, \theta_1$ to v_2, and assign $\theta_3, \theta_1, \theta_2$ to v_3. This can be implemented by a simple matrix multiplication. As illustrated in Figure 4, the vertex-based dihedral-angle matrix $\mathbf{D}_v(\mathcal{M})$ is defined as the multiplication of the vertex-face adjacency matrix $\mathbf{A}_{vf}(\mathcal{M})$ and the face-based dihedral-angle matrix $\mathbf{D}_f(\mathcal{M})$, i.e.,

$$\mathbf{D}_v(\mathcal{M}) = \mathbf{A}_{vf}(\mathcal{M}) \cdot \mathbf{D}_f(\mathcal{M}). \tag{5}$$

3.3. Network

We feed the aforementioned feature matrix $\widetilde{\mathbf{G}}$ into the network in a multi-resolution manner. Specifically, we select three different resolutions of $\widetilde{\mathbf{G}}$ in a decreasing order, where the dominant dimensions of $\widetilde{\mathbf{G}}$ are denoted as $\mathbf{k} = \{k_i \mid i = 0, 1, 2\}$. We then select the spectral basis $\mathbf{\Phi}_{k_i}$ and compute the corresponding feature matrix $\widetilde{\mathbf{G}}$, denoted as $\widetilde{\mathbf{G}}_0$, $\widetilde{\mathbf{G}}_1$, $\widetilde{\mathbf{G}}_2$, respectively. Our empirical evidence shows that k_0, i.e., the largest one, should be greater than or equal to $n_v/2$ [36], where n_v is the maximum total number of vertices for the 3D models in the dataset.

As shown in Figure 3, our network resembles the U-Net structure. The three spectral feature matrices, $\widetilde{\mathbf{G}}_0$, $\widetilde{\mathbf{G}}_1$, $\widetilde{\mathbf{G}}_2$, are processed by the corresponding convolutional blocks independently and then fused from the higher resolution to the lower resolution. We use the mirror structure during decoding, while taking the encoding feature at the same resolution via the skip connection at each level. The network can be easily instantiated for different tasks by adding the head blocks; See the classification block or the segmentation block in Figure 3. The main difference between ours and the existing U-Net networks lies in the SE-blocks and the proposed Laplacian pooling/unpooling operations, as described later.

3.3.1 SE-ResNet Block

Instead of learning the multiplication in the spectral domain as some spectral-based networks [29, 39] do, we apply the convolution operations to the spectral signals. However, due to the lack of shift-invariance in the spectral domain, the commonly-used convolutions with large receptive fields for pattern recognition are not suitable. Here, we use the convolution with small kernels. The small-sized convolution kernel acts as an aggregation function to fuse the close frequency features.

In addition, we take the Squeeze-and-Excitation ResNet (SE-ResNet) [24] (see Figure 5) as the basic unit to process the spectral signals. The Squeeze-and-Excitation (SE) module first performs the squeeze operation on the feature map obtained by convolution to obtain the channel-level global features. In order to obtain the weights of different channels, the excitation operation on the global features is then performed to learn the relationship between different channels. Finally, the channel weights are multiplied by the original feature map to get the final features. In its essence, the SE module performs the attention or gating operations on the channel dimension. This attention mechanism allows the model to pay more attention to the significant channel features while suppressing those less important channel features. The module alleviates the tedious work of manual feature selection and improves the representational capacity of a network by performing dynamic channel-wise feature recalibration.

Unlike the other networks that directly apply the same trainable network weights on the feature map of various data samples, the SE-block predicts the adaptive weights to fuse the channel-wise features, and the residual connection helps avoid the vanishing gradients of the deep network. We experimentally found that the SE-block achieves the best performance with the kernel size $ks = 3$ and padding $p = 1$ while the layers without the SE-block achieve the best performance when $ks = 1$ and $p = 0$.

3.3.2 Laplacian Pooling and Unpooling

Note that the input of our network is the multi-resolution spectral signals with different sizes of eigenvector basis, which is obtained by Eq. (3). The Laplacian pooling and unpooling operations are necessary to fuse them together. Figure 6 illustrates how the Laplacian pooling and unpooling operations transform the signals between the spectral basis with different resolutions.

For the i-th layer and the j-th layer in the network, we propagate the output of the i-th layer to the j-th layer based

on $\mathbf{\Phi}_{k_i}$ and $\mathbf{\Phi}_{k_j}$:

$$\mathbf{\Gamma}_{ij} = (\mathbf{\Phi}_{k_j}^T \mathbf{\Phi}_{k_i})\mathbf{\Gamma}_i. \tag{6}$$

The transformed feature $\mathbf{\Gamma}_{ij}$ is then concatenated with the j-th layer, yielding a concatenated feature:

$$\widetilde{\mathbf{\Gamma}}_j = \mathbf{\Gamma}_j \oplus \mathbf{\Gamma}_{ij}, \tag{7}$$

which shows the mechanism of our Laplacian pooling operation when the resolution of i-th layer is larger, and Laplacian unpooling operation otherwise.

3.3.3 Classification and Segmentation Modules

It's natural that our Laplacian2Mesh network can deal with different mesh understanding tasks if the head blocks are carefully set. Figure 3 illustrates the classification and segmentation blocks, as well as their connection with the backbone network. For the classification task, we only make use of the encoding stage of our network and fuse the multi-resolution features to form the input of the classification block (cls-block). The classification block follows the popular implementation in the related works [45], and uses the simple Global Average Pooling and softmax equipped MLP to obtain the final prediction. As for the segmentation task, it has to utilize the entire U-Net architecture and then add a segmentation block (seg-block) to compute the segmentation-aware features in the spectral domain. The spectral feature is then transformed to the spatial domain. Finally, we use a softmax function to obtain the segmentation label for each vertex.

We carefully designed cls-Block (see Figure 7) and seg-Block (see Figure 8) for the tasks of shape classification and semantic segmentation, respectively. In these blocks, $ELU(\cdot)$ [10] is chosen as the activation function in our Laplacian2Mesh, which eliminates the influence of bias shift and improves the robustness to noise.

3.4. Loss Function

For the shape classification task, one can simply use the cross-entropy loss between the predicted label and the ground-truth. For the shape segmentation task, however, we need to add one additional loss term to ensure the spatial coherence of the vertex-based part labels. Therefore, our loss function for the mesh segmentation task can be written as

$$\mathcal{L} = \mu \cdot \mathcal{L}_{\text{ce}} + \nu \cdot \mathcal{L}_{\text{adj}}, \tag{8}$$

where $\mathcal{L}_{\text{ce}}$ is the segmentation loss to minimize the error between the predicted vertex label and the ground-truth, $\mathcal{L}_{\text{adj}}$ is the adjacency loss to encourage the label coherence between neighboring vertices, and μ, ν are a pair of hyperparameters to balance the two terms.

Segmentation loss. Note that the output of the segmentation block still encodes the segmentation information in the spectral domain. We need to transform it to the spatial domain with Eq. (3). For each vertex, we obtain a one-hot vector to characterize the probability in which the vertex belongs to each part, and assign the vertex to the part with the maximum probability. The segmentation loss is then defined as the cross-entropy loss between the predicted labels and the ground-truth labels.

Adjacency loss. Intuitively speaking, a pair of neighboring vertices tend to own consistent labels. Therefore, it is necessary to enforce the label coherence in the one-ring neighborhood for each vertex. We introduce three matrices, i.e., $\mathbf{\Psi}, \mathbf{\Omega}$ and $\mathbf{A}$, to weigh the influence between neighboring vertices. Let $\mathbf{\Psi}$ be the all-pair straight-line distance matrix between vertices, satisfying $\mathbf{\Psi} = \left\{\|\mathbf{v}_i - \mathbf{v}_j\| \mid i, j \in \{0, 1, \cdots, n-1\}\right\}$. Let $\mathbf{\Omega} = \exp(-\mathbf{\Psi}/(2\delta))$ be the Gaussian filter with a bandwidth δ. Let $\mathbf{A}$ be the adjacency matrix of the mesh. By defining

$$\mathbf{\Theta} = \mathbf{\Omega} \odot \mathbf{A}, \tag{9}$$

we obtain a $n \times n$ matrix $\mathbf{\Theta}$ to characterize the mutual influence between neighboring vertices, where $\odot$ is the Hadamard product. It's worth noting that $\mathbf{\Theta}$ is sparse due to the fact that $\mathbf{A}$ is sparse, and thus can be quickly computed.

Recall that each vertex has a one-hot vector to encode the probability in which the vertex belongs to each part. Let v_i, v_j be a pair of neighboring vertices. For v_i, we extract the maximum component of its one-hot vector, denoted by h_i. Suppose that the maximum component occurs at the s_i-th slot. We then extract the corresponding component of the one-hot vector of v_j, denoted by $h_j^{s_i}$. Similarly, we can define h_j and $h_i^{s_j}$. We use $|h_i - h_j^{s_i}| + |h_j - h_i^{s_j}|$ to measure the symmetric label difference between v_i and v_j. In this way, we obtain a sparse matrix $\mathbf{H}$ to characterize the pairwise label difference. Therefore, the matrix $\mathbf{\Theta} \odot \mathbf{H}$ is the improved difference matrix weighted by the above-mentioned influence. The overall label coherence can be obtained by summing the elements together:

$$\mathcal{L}_{\text{adj}} = \frac{1}{n} \cdot \mathbf{1}^{\text{T}} \times \left(\frac{\mathbf{\Theta} \odot \mathbf{H} \times \mathbf{1}}{\mathbf{A} \times \mathbf{1}}\right), \tag{10}$$

where $\mathbf{1}$ is a column vector consisting of n 1's.

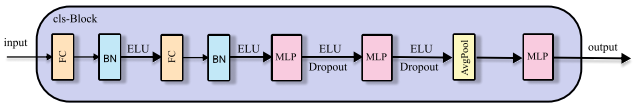


Figure 7. The head block for the classification task.

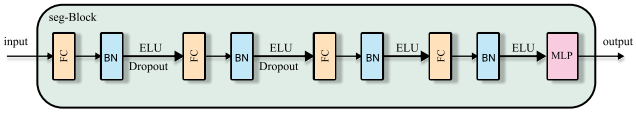


Figure 8. The head block for the semantic segmentation task.

4. Experiments

We present extensive experiments to validate the effectiveness on the mesh classification/segmentation tasks. The data preprocessing/augmentation step is conducted after SubdivNet [25]. All the meshes are scaled into a unit cube. During the training phase, we apply an isotropic scaling operator on each model, and the scaling factor is randomly sampled from a normal distribution with an expected value $\xi = 1$ and a standard deviation $\sigma = 0.1$. We also randomly select three Euler angles, each of which being 0, or $\pi/2$, or π, or $3\pi/2$, to perform the orientation augmentation. The meshes in each dataset are simplified to have roughly equally many faces.

4.1. Mesh Classification

We demonstrate the superior classification ability of Laplacian2Mesh on the two datasets: SHREC11 [38] and Manifold40 [25]. The classification accuracy statistics are reported to compare the performance among different methods.

SHREC11. The SHREC11 dataset consists of 30 classes, with 20 examples per class. Following the setting of [13], all the methods are evaluated based on the 16-4 and 10-10 train-test split, respectively. Our method outperforms the others on both train-test splits, and achieves the perfect classification accuracy (100%), as shown in Table 1. Tests on the SHREC11 dataset indicate that our Laplacian2Mesh is competitive on the mesh classification task.

Table 1. The classification accuracy statistics on the SHREC11 dataset [38].

Method	Split 16	Split 10
GWCNN [13]	96.6%	90.3%
MeshCNN [20]	98.6%	91.0%
PD-MeshNet [45]	99.7%	99.1%
MeshWalker [31]	98.6%	97.1%
SubdivNet [25]	99.9%	99.5%
HodgeNet [60]	99.2%	94.7%
DiffusionNet (xyz) [57]	-	99.4%
DiffusionNet (hks) [57]	-	99.5%
Laplacian2Mesh (ours)	**100**%	**100**%

Manifold40. Manifold40 is a larger and more challenging dataset, containing 12311 CAD models from 40 categories. It is reconstructed from ModelNet40 [70] to obtain better triangulation quality meshes. Therefore, the reconstruction and simplification operations during the dataset construction stage inevitably introduce a slight difference in shape and a significant difference in tessellation. As shown in Table 2, our method has a comparable performance to SubdivNet [25], and gains better accuracy than other methods. But we must point out that our approach does not include a tedious step of mesh subdivision [41] as included in SubdivNet [25].

Table 2. The classification accuracy statistics on Manifold40 [25].

Method	Input	Accuracy
PointNet++ [50]	point cloud	87.9%
PCT [17]	point cloud	**92.4**%
MeshNet [14]	mesh	88.4%
MeshWalker [31]	mesh	90.5%
SubdivNet [25]	mesh	**91.2**%
Laplacian2Mesh (ours)	mesh	90.9%

4.2. Mesh Semantic Segmentation

We conduct the mesh semantic segmentation experiments on the human body dataset [42] and COSEG dataset [68]. The mesh segmentation task aims to predict the segmentation label per face. Since the labels are defined on the mesh vertices in our approach, we apply the majority voting to obtain the labels on the faces, rather than the "soft" label conversion or some smooth processing technique [31].

Human Body Segmentation. The human body dataset, labeled by [42], consists of 370 training shapes from SCAPE [2], FAUST [3], MIT [65], and Adobe Fuse [1]. The 18 models in the test set are from SHREC07 [16] (humans) dataset. All the meshes are manually segmented into 8 parts [26]. We use the same version of human body dataset as in MeshCNN [20], which is downsampled to 1500 faces per mesh.

As shown in Figure 9, our method is able to learn the consistent and accurate part segmentation of human bodies. We report the segmentation performance of various methods in Table 3. Our method mostly outperforms the related works such as MeshCNN [20], PD-MeshNet [45], and HodgeNet [60]. Despite being slightly inferior to the DiffusionNet [57], our method is superior in terms of its noise-resistance property, as will be demonstrated in Section 5.1.

COSEG. We also evaluate the performance of Laplacian2Mesh on the three largest sets from the COSEG shape dataset: Vases, Chairs, and Tele-aliens containing 300, 400, and 200 models, respectively. The meshes are labeled into 3 parts (Chairs set) or 4 parts (Vases set & Tele-aliens set). We generate a random 85%-15% train-test split for each set, as in MeshCNN [20].

The quantitative results are provided in Table 4, and the qualitative results for all models in the test set are visualized in Figure 10. Our method outperforms other methods on the Vase and Chair classes, but performs worse on the

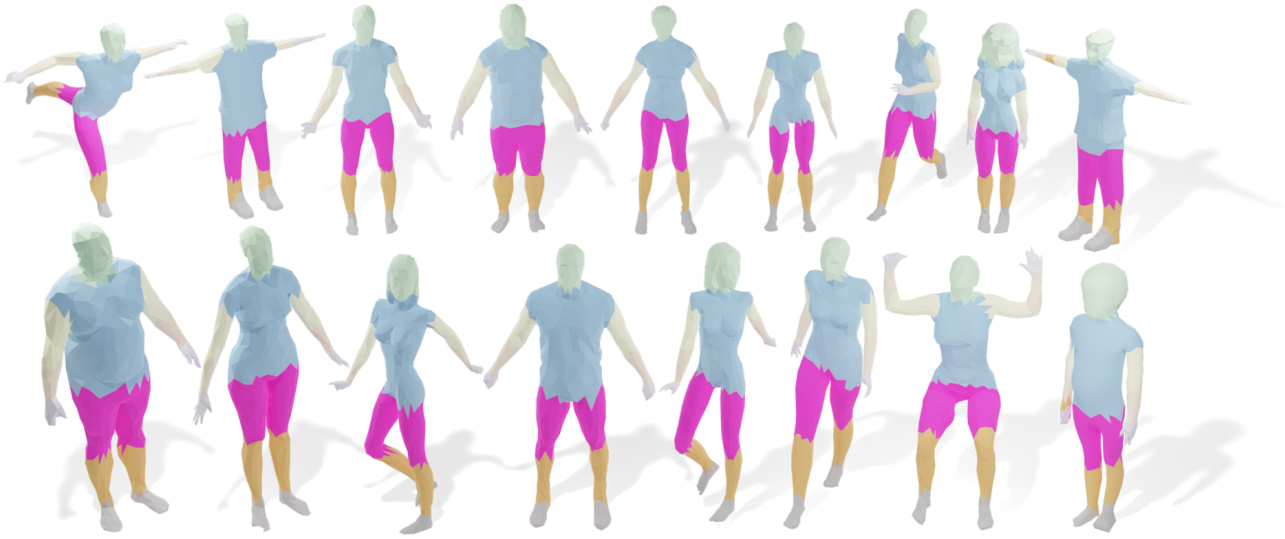


Figure 9. We test the dataset of Human Body, and visualize the segmentation result for every model.

Table 3. The mesh segmentation accuracy statistics on the HumanBody dataset [42]. Note that DiffusionNet [57] supports various inputs. The options "xyz" and "hks" denote the raw coordinates and the heat kernel signatures, respectively.

Method	Input	Accuracy
PointNet [49]	point cloud	74.7%
PointNet++ [50]	point cloud	82.3%
MeshCNN [20]	mesh	85.4%
PD-MeshNet [45]	mesh	85.6%
HodgeNet [60]	mesh	85.0%
DiffusionNet (xyz) [57]	mesh	88.8%
DiffusionNet (hks) [57]	mesh	**90.5**%
Laplacian2Mesh (ours)	mesh	88.6%

Table 4. The mesh segmentation accuracy statistics on the COSEG dataset [68].

Method	Vases	Chairs	Tele-aliens
DCN [72]	90.9%	95.7%	-
MeshCNN [20]	92.4%	93.0%	**96.3**%
HodgeNet [60]	90.3%	95.7%	96.0%
Laplacian2Mesh (ours)	**94.6**%	**96.6**%	95.0%

Tele-alien class. We explain this as follows. The triangulation quality of the Tele-alien meshes is bad, containing long and thin triangles. To our knowledge, the Laplacian matrix is likely to be ill-conditioned when bad triangles exist. Figure 10 visualizes the semantic segmentation results computed by our method.

4.3. Ablation Studies

The ingredients of our method lie in many aspects, including the selection of the input features, the multi-resolution spectral signals, the network architecture, the adjacency loss, and the selection of the kernel size for the training. We evaluate our method by conducting the ablation studies on the mesh segmentation task.

The selection of the input features. As described in Section 3.2, we pre-compute a set of intrinsic and extrinsic geometric features and transform them into the spectral signals as the input of our network. Table 5 reports the segmentation performance on the Vase class of COSEG dataset, Manifold40 dataset, and the human body dataset after removing each of the features. It shows that the Laplacian2Mesh with the full set of features achieves the best performance on all the three datasets. In Table 5, we highlight the features that cause the greatest performance degradation on each dataset. It indicates that various classes have different dependency on the input features. Our Laplacian2Mesh can automatically learn the weighting scheme to fuse the diverse geometric features.

The multi-resolution signals. As described in Section 3.1, the selection of the hyperparameter k is of vital importance to form the spectral basis, which helps filter out the high-frequency noise. We compute the multi-resolution spectral signals as the input of our network by selecting $\mathbf{k} = \{k_i \mid i = 0, 1, 2\}$ to form a set of basis and perform the spectral transform.

Table 6 reports the segmentation performance statistics

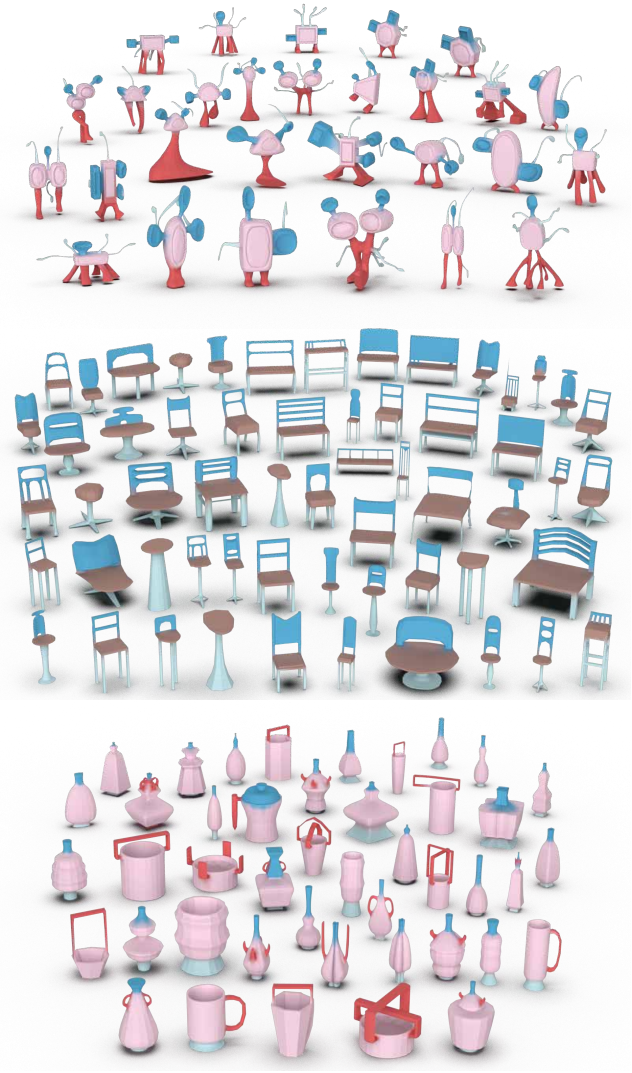


Figure 10. A gallery of segmentation results of the COSEG dataset. From top to bottom: Tele-aliens, Chairs, and Vases classes.

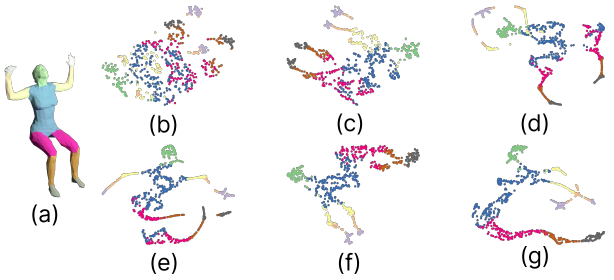


Figure 11. t-SNE visualization of the network processing. We project the intermediate spectral features back to the spatial vertex-wise features for a better understanding. The vertices are colored by their ground-truth segmentation labels. (a): the ground-truth segmentation; (b-d): the encoded feature of each resolution before any Laplacian pooling/unpooling; (e): the encoded feature after fusing all the resolutions; (f): the decoded feature after the U-Net processing, (g): the output segmentation predictions.

Table 5. The results of ablation experiments on the input features. We highlight the best (bold) and the worst (underlining) performance scores for each class. It can be seen that the importance of geometric features varies on different datasets.

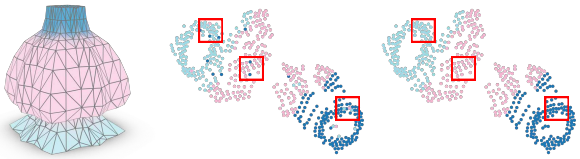
Method	**Vases**	**Manifold40**	**Human Body**
w/o vertex coordinates	<u>85.9%</u>	88.7%	87.4%
w/o vertex normal	93.2%	90.2%	87.3%
w/o Gaussian Curvature	92.2%	90.3%	87.7%
w/o EigenVectors	86.3%	<u>85.3%</u>	88.2%
w/o HKS	92.0%	89.8%	<u>77.5%</u>
w/o Dihedral Angles	90.5%	90.1%	87.1%
Laplacian2Mesh (ours)	**94.6%**	**90.9%**	**88.6%**

Table 6. The segmentation performance with the input spectral signals of different resolutions. The first three rows are the single-resolution experiments, indicating that it is proper to take k as one half of the total number of vertices. The other rows show the results of using the multi-resolution inputs.

input sizes	**Human Body**	**Tele-alien**
749-0-0	84.8%	90.7%
512-0-0	86.9%	93.1%
128-0-0	86.7%	92.9%
512-128-0	87.1%	93.3%
512-128-32	**88.6%**	94.5%
512-256-64	86.3%	**95.0%**

with different hyperparameter settings, including the single-resolution experiments (e.g. 128 – 0 – 0, 512 – 0 – 0, 749 – 0 – 0, with 749 being the smallest vertex number among all the meshes in the dataset) and the multi-resolution experiments (e.g. 512 – 128 – 0, 512 – 128 – 32, 512 – 256 – 64). By comparing the single-resolution experiments, it shows the importance of choosing a proper k, as either the largest or a smaller k will decrease the performance. On the other hand, the comparison between the single-resolution and multi-resolution experiments validates the effectiveness of our setting.

From a different perspective, it is intuitive and interesting to visualize the learned features to understand the role of the multi-resolution input signals. Note that the whole network processing is in the Laplacian spectral domain, and therefore we need to transform the spectral features $\widetilde{\mathbf{F}}$ back



Ground-Truth | Laplacian2Mesh (w/o adjacency-loss) | Laplacian2Mesh

Figure 12. t-SNE visualization of the segmentation results with and without the adjacency loss. The vertices are colored by their ground-truth label. The adjacency loss obviously improves the smoothness of the segmentation results.

Table 7. The ablation studies of the network architecture and the adjacency loss.

Experiments	Vases	Chairs
Baseline	89.2%	93.8%
+ SE-block	90.6%	94.9%
+ Multi-resolution	91.3%	95.5%
+ Laplacian pooling/unpooling	93.3%	96.3%
+ Adjacency loss (full)	**94.6%**	**96.6%**

to the spatial domain via the inverse mapping:

$$\widehat{\mathbf{F}} = \mathbf{\Phi}_k \widetilde{\mathbf{F}}. \tag{11}$$

We show the t-SNE visualization of the intermediate features in Figure 11, where the color indicates the ground-truth label of each vertex. The first row is the encoded feature of each resolution before any pooling/unpooling operation. The signals with mostly low-frequency signals, i.e. $k = 32$, obtain more disjoint clusters, while the signals with more high-frequency signals, i.e. $k = 512$ have the encoded features gathered together probably because of the distracting local details. However, after we fuse the features from the multi-resolution signals during encoding, the clusters are more separated at the boundary with the help of the local information (see (d) and (e)). Finally, the features are moved and clustered based on their semantic labels, as the vertices on the human legs are moved together after the decoding stage (see (e) and (f)). This visualization shows the necessity of fusing the multi-resolution signals.

Network Architecture and Loss. We show how it evolves with the key designs in our network and the loss function. The quantitative evaluations are in Table 7. We start from a baseline network implemented as a simple convolutional network with skip connections, which is similar to the one-resolution component of our network while the SE-blocks are replaced by the vanilla convolutional blocks. We progressively equip the network with the SE-blocks with small-sized convolution kernels, the multi-resolution network architecture but with spatial pooling/unpooling, our Laplacian pooling/unpooling operations, and the adjacency loss. The quantitative evaluation clearly demonstrates that our key designs are necessary to learn the mesh semantics in the spectral domain.

The adjacency loss is used to guarantee the label coherence property of the semantic segmentation on the mesh surface. As indicated by the t-SNE visualization shown in Figure 12, without the adjacency loss, some outlier vertices may be assigned with a wrong label, leading to conspicuous visual segmentation artifacts.

Kernel Size. As said before, our network should be equipped with small kernel convolution. As illustrated in Figure 5, we directly use the common convolution kernel settings for SE-block [22], which is $ks = 3$. We also tried using other kernel sizes for SE-block, but $ks = 3$ got the best results. For the layers without the SE-block (the MLPs in our network), we experimented with the impact of different sizes of the kernel on the segmentation results.

We replace the MLPs in our network with convolutions of different kernel sizes (i.e. 1, 3, 5, 7). From Table 8, we can also find that a large convolution kernel does not bring us similar benefits as it is used in 2D CNNs, on the contrary, the large-sized kernel will degrade the performance of our network. So we select $ks = 1$ (i.e. MLP) for the layers without the SE-block.

Table 8. The segmentation performance with the different kernel sizes.

	Vases	Chairs	Human Body
kernel_size = 7	92.3%	94.8%	86.5%
kernel_size = 5	92.8%	95.6%	87.1%
kernel_size = 3	93.5%	96.1%	87.5%
kernel_size = 1	**94.6%**	**96.6%**	**88.6%**

5. Strengths and Limitations

5.1. Resistance to Noise

To test the robustness against noise, we construct the noisy dataset by adding Gaussian noise of several levels (i.e. 0.005, 0.010, 0.050, 0.080, 0.1 w.r.t. the diagonal length of each model) to the original human body dataset [42]. We train our network with the original noise-free dataset and test on the noisy meshes. Figure 13 shows the comparison between DiffusionNet and ours with various settings.

Table 9 reports the quantitative comparisons. The DiffusionNet performance has significantly decreased, as can be seen, whereas our performance has essentially been steady. Interestingly, once we replace the Laplacian pooling/unpooling of our network by the max pooling (in other

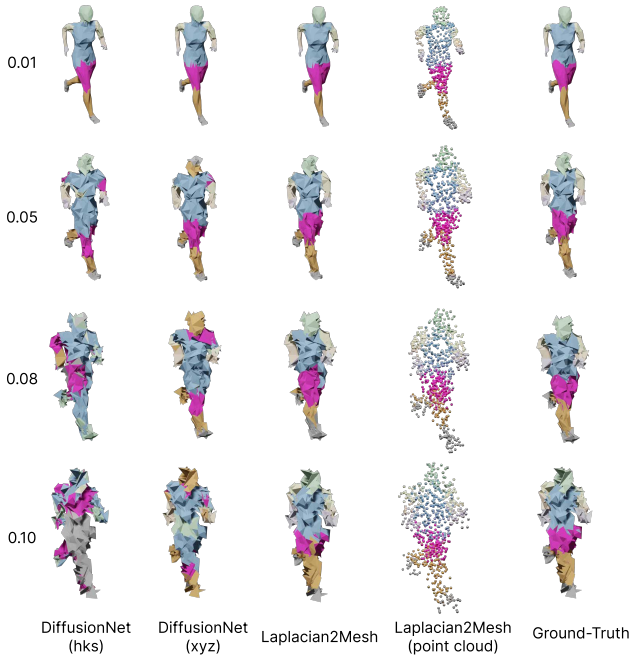


Figure 13. The qualitative results of testing the noise-resistance. From top to bottom: As noise levels rise, segmentation outputs of DiffusionNet diverge significantly from the ground-truth, while ours is more in line with it.

Table 9. The mesh segmentation accuracy statistics when training on the original human body dataset [42] and testing on the noisy meshes. ♣ means that we use the max-pooling instead of Laplacian pooling in our network. The DiffusionNet has a drastic decrease while ours does not, which shows that ours is more robust to noise.

Method	w/ Noise	w/o Noise
DiffusionNet (xyz) [57]	64.4%	88.8%
DiffusionNet (hks) [57]	62.0%	**90.5**%
Laplacian2Mesh (ours) ♣	60.2%	85.9%
Laplacian2Mesh (ours)	**86.7**%	88.6%

words, we apply the spatial U-Net directly to our spectral signals), the performance has an obvious drop. This indicates the necessity of our design to deal with the spectral signals.

5.2. Non-watertight & Non-manifold Mesh Segmentation

Many mesh segmentation methods [20, 25] are designed on the premise that the input mesh must be watertight or manifold or both, which limits their usage on a wider range of various mesh datasets. For example, SubdivNet [25] assumes that every face of the mesh should have 3 neighbor faces. MeshCNN [20] requires each edge to be shared by

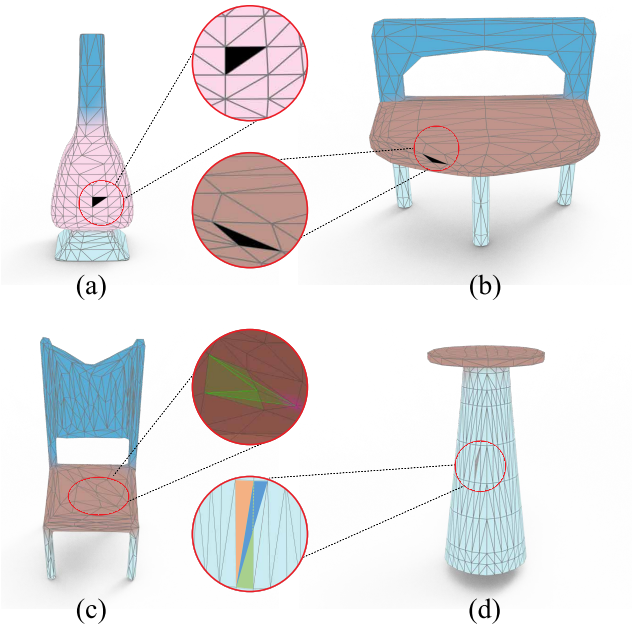


Figure 14. Our Laplacian2Mesh is able to perform segmentation on the non-watertight and non-manifold data. (a) and (b) are non-watertight vases and chairs, respectively; (c) has a non-manifold vertex, and (d) has a non-manifold edge.

two triangular faces. Therefore, these methods will fail even with a small number of small gaps or non-manifold elements (non-manifold vertices or non-manifold edges).

We show our segmentation results on the non-watertight and non-manifold meshes in Figure 14. Our Laplacian2Mesh is flexible to deal with various structures [48, 58]. If the edge is a non-manifold edge, our solution is to use robust-laplacian [58] and discard all dihedral angles corresponding to this edge. In addition, if the face $\mathbf{f}_i$ is a boundary face, we will fill the missing dihedral angle value with the value 1, which means that this location has gentle terrain.

5.3. Incomplete Models

We select some models from the COSEG dataset and the Human-Body dataset, and manually delete some triangle faces. Note that it results in open surfaces, which cannot be processed by other mesh understanding networks. We test these incomplete models with the pre-trained segmentation network.

The top two rows of Figure 15 show the segmentation results when more and more triangle faces are removed, while the last row shows the segmentation of different shapes. It can be observed that the overall segmentation results remain reasonable and sound, indicating that our model can be generalized to incomplete models. However, as shown in the human case, the broken models may have poorly fine-

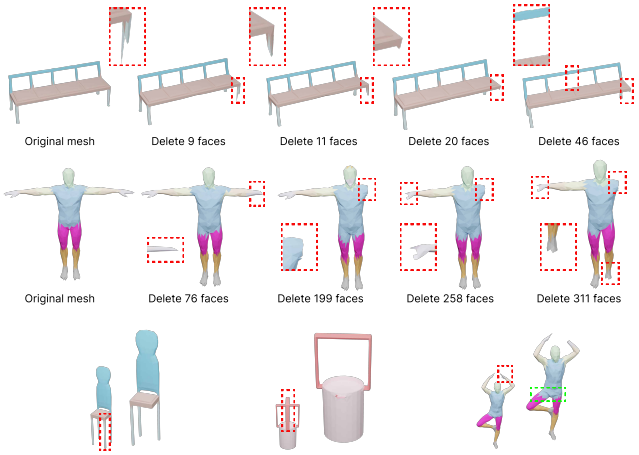


Figure 15. The results on incomplete models. The red dashed boxes on the mesh indicate the deleted parts. The green dashed boxes highlight the inaccurate segmentation boundaries.

grained segmentation at the boundaries of the parts near the removed region. It's an interesting future direction to study that how the spectral-based deep understanding methods are affected by different geometric and topological changes.

5.4. More Merits

Besides the ability to deal with all kinds of mesh defects and noise, the biggest advantage of our approach is to decouple various shape understanding tasks from the tedious and complicated triangulation. It can also support various kinds of inputs, such as meshes and point clouds, as long as the Laplacian matrix can be equipped. To summarize, learning the latent features in the spectral domain rather than the spatial domain inherits the nice properties of the spectral analysis.

5.5. Limitations

The side-effect of the high-frequency filtering in our spectral transform is the missing of local features in some unusual cases. Taking the Cube Engraving dataset [32] as an example, each model in this dataset is a cube with a facet being "engraved", as demonstrated in Figure 16. The engraved objects are important semantic hints. This dataset contains 4600 objects with the 3910-690 train/test split. But our approach regards the engraved objects as high-frequency information. Not surprisingly, as reported in Table 10, our Laplacian2Mesh ignores the local shape classification and performs worst among all the mesh classification methods.

6. Conclusion and Future Work

We present Laplacian2Mesh, a general network architecture for mesh understanding in the spectral domain. Our network takes the multi-resolution spectral signals as the input, and follows the structure of the U-Net architecture. We design the small-sized SE-blocks and propose the Laplacian pooling/unpooling operations to fuse the features from different levels of resolutions. The ablation studies demonstrate the necessity of our key designs to enable the learning of spectral mesh signals. Compared to state-of-the-art methods, our approach not only achieves competitive or even better performances on the mesh classification and segmentation tasks, but also can handle non-watertight and non-manifold meshes. It also owns the nice feature of being noise-resistant.

There is a vast of directions for further exploration. First, to perform the Laplacian spectral transform, we need to compute the eigendecomposition of the Laplacian matrix for each shape. Consequently, the eigenvectors, or to put it another way, the spectral basis, are not aligned among the shapes in the dataset. In our Laplacian2Mesh network, we seek for the small-sized SE-blocks to solve this problem. However, aligning the spectral basis will enable us to make use of the larger receptive fields, which will probably improve the performance in the mesh understanding tasks. Also, although we have concluded that the multi-resolution spectral signals are stronger than the single-resolution input, it's still a remaining question on how to automatically select the hyper-parameters, i.e. the number of eigenvectors, to adapt to different shape datasets.

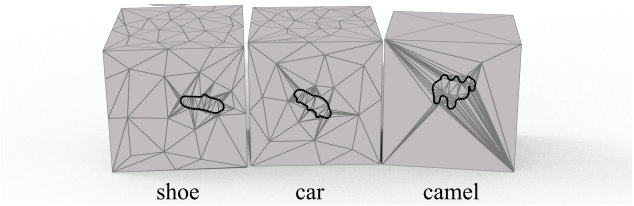


Figure 16. The Cube Engraving dataset [32]. Our method fails on this dataset since our approach deems the engraved objects as high-frequency information.

Table 10. The classification accuracy statistics on the Cube Engraving dataset [32].

Method	Accuracy
PointNet++ [50]	64.3%
MeshCNN [20]	92.2%
PD-MeshNet [45]	94.4%
MeshWalker [31]	98.6%
SubdivNet [25]	**98.9**%
Laplacian2Mesh (ours)	91.5%

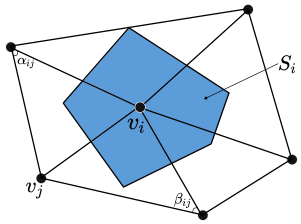
Acknowledgement

This work is supported by "The Fundamental Research Funds of Shandong University". The authors would like to thank the reviewers for their insightful comments.

Appendices

A. Eigendecomposition Matrix

For triangle meshes, a common choice of the discrete Laplacian is cotangent Laplacian, which forms the basis for the theory of discrete holomorphic functions and discrete Riemann surfaces. It arises naturally via linear finite elements, discrete exterior calculus, electrical networks, and minimal surfaces. This operator is very sparse, easy to build, and generally works well for unstructured meshes with irregular vertex distributions, which also can be used on non-manifold meshes by accumulating per-triangle matrices [48, 58].



For a manifold triangle mesh, the discrete Laplacian Δ of a scalar function f at a vertex v_i can be approximated as:

$$\Delta f(v_i) := \frac{1}{2S_i} \sum_{v_j \in r_1(v_i)} (cot\ \alpha_{ij} + cot\ \beta_{ij})(f(v_j) - f(v_i)) \tag{A.1}$$

where $r_1(v_i)$ is over all the 1-ring adjacent vertices of v_i, α_{ij} and β_{ij} are the two angles opposite to edge $v_i v_j$, and S_i is the vertex area of v_i.

Then, the Laplacian is encoded in a sparse matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$ for the mesh $\mathcal{M}$, such that

$$\begin{pmatrix} \vdots \\ \Delta f(v_i) \\ \vdots \end{pmatrix} = \mathbf{L} \cdot \begin{pmatrix} \vdots \\ f(v_i) \\ \vdots \end{pmatrix} \tag{A.2}$$

We define $\mathbf{M}$ as the diagonal matrix whose i-th entry along the diagonal is twice the vertex area S_i, that is

$$\mathbf{M}^{-1} = diag(\cdots, \frac{1}{2S_i}, \cdots). \tag{A.3}$$

Then, the discretization of the Laplacian can be represented as

$$\mathbf{L} = \mathbf{M}^{-1}\mathbf{C}. \tag{A.4}$$

As we know from [34], the eigendecomposition of $\mathbf{L}$ is the generalized eigenvalue problem for λ eigenvalues with corresponding eigenvectors Φ, which satisfies the following formula:

$$\mathbf{L}\mathbf{\Phi} = \mathbf{M}^{-1}\boldsymbol{\lambda}\mathbf{\Phi}, \tag{A.5}$$

where $\boldsymbol{\lambda} = diag(\lambda_0, \lambda_1, \cdots, \lambda_{n-1})$, and $\mathbf{\Phi}^T\mathbf{M}^{-1}\mathbf{\Phi} = \mathbf{I}$.

To facilitate computation, the generalized eigenvalue problem is equivalently transformed into the standard eigenvalue problem. So Eq. A.5 is rewritten as

$$(\mathbf{ML})\mathbf{\Phi} = \boldsymbol{\lambda}\mathbf{\Phi}. \tag{A.6}$$

Deriving from Eq. A.4, we know that Eq. A.6 can be written as

$$\mathbf{C}\mathbf{\Phi} = \boldsymbol{\lambda}\mathbf{\Phi}, \tag{A.7}$$

where $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}$ in the formula.

To simplify computations in our work, the eigenvector matrix $\mathbf{\Phi}$ is derived from performing an eigendecomposition on the cotangent weight matrix (Eq. A.7) instead of the discrete Laplace-Beltrummy matrix (Eq. A.5).

In [35], since the authors obtained the eigenvectors by eigendecomposition of Laplacian, the spectral reconstruction is written as

$$\widehat{\mathbf{V}} = \mathbf{\Phi}_k(\mathbf{\Phi}_k^T\mathbf{MV}). \tag{A.8}$$

If we directly eigendecompose the cotangent weight matrix, then the spectral reconstruction is

$$\widehat{\mathbf{V}} = \mathbf{\Phi}_k(\mathbf{\Phi}_k^T\mathbf{V}). \tag{A.9}$$

Comparing A.8 and A.9, we can find that the term of vertex area can be ignored in our Laplacian2Mesh.

B. Quantitative Comparison

We use the meshes in the Tele-Alien dataset to perform eigendecomposition and spectral reconstruction. There are three metrics we used to evaluate the performances of the two methods, which are the mean of eigendecomposition time (MET), the mean of spectral reconstruction errors (MRE), and the mean of spectral reconstruction time (MRT).

MET is the average time taken by all meshes in the dataset to perform eigendecomposition, which is defined as

$$MET(\mathcal{M}_0, \mathcal{M}_0, \cdots, \mathcal{M}_{N_m-1}) = \frac{1}{N_m} \sum_{i=0}^{N_m-1} t_i, \tag{A.10}$$

where t_i is the eigendecomposition time for the mesh $\mathcal{M}_i$, and N_m is the number of mesh models in the dataset.

For spectral reconstruction, the goal is that the mesh reconstructed with fewer eigenvectors has a smaller reconstruction error than the original mesh model, which means that the distance between the vertex coordinates $\widehat{\mathbf{v}}_i$ of the mesh $\widehat{\mathcal{M}_i}$ obtained by spectral reconstruction and the vertex coordinates $\mathbf{v}_i$ of the original mesh $\mathcal{M}_i$ should be as small as possible. Therefore, we perform spectral reconstruction by choosing different numbers of eigenvectors (i.e.

$\mathbf{k} = \{751, 400, 100, 80, 40, 20, 10\}$), and then the MRE is

$$MRE(\mathcal{M}_0, \mathcal{M}_0, \cdots, \mathcal{M}_{N_m-1}) = \frac{1}{N_m N_k} \sum_{i=0}^{N_m-1} \sum_{j=0}^{N_k-1} \|\widehat{\mathbf{v}}_{ij} - \mathbf{v}_{ij}\|, \tag{A.11}$$

where N_k is the number of combinations of eigenvectors, which is $N_k = 7$ in our experiment. MRT is the average spectral reconstruction time of the meshes in the dataset with different k_i, which is

$$MRT(\mathcal{M}_0, \mathcal{M}_0, \cdots, \mathcal{M}_{N_m-1}) = \frac{1}{N_m N_k} \sum_{i=0}^{N_m-1} \sum_{j=0}^{N_k-1} \bar{t}_{ij}, \tag{A.12}$$

where $\bar{t}_{ij}$ is the spectral reconstruction time for the mesh $\mathcal{M}_i$ when k_j eigenvectors are chosen.

As the results are shown in Table A.1, we can know that we can perform eigendecomposition faster because we do not need to calculate the area term, and our method achieves lower MRE and uses less MRT. The comparison between the eigendecomposition of Laplacian and the eigendecomposition of cotangent weight matrix experiments validates the effectiveness of our reasoning.

Table A.1. The performance of discrete Laplacian matrix and cotangent weight matrix for the task of spectral reconstruction. "MET" is the mean of eigendecomposition time for the meshes in the Tele-Alien dataset, "MRE" is the mean of spectral reconstruction error, and "MRT" is the mean of spectral reconstruction time.

Method	MET ↓	MRE ↓	MRT ↓
Laplacian	1048ms	6.55×10^{-1}	2.18ms
cotangent (ours)	**824**ms	$\mathbf{6.33 \times 10^{-1}}$	**1.27**ms

C. Dense Meshes

We compare the mesh segmentation of our Laplacian2Mesh on sparse meshes [68] (used in the main paper) and dense meshes [25]. Table A.2 reports the segmentation results and the running time on sparse meshes in the first two rows, and those on dense meshes in the bottom two rows. The experiment in the last row shows that our method can also achieve gratifying performance on dense meshes. However, as shown in this table, the performance relies on the input size of the network, i.e. the number of eigenvectors used to form our multi-resolution spectral signals. Dense meshes require larger memory space and computation amount to perform the eigenvector decomposition and the transformation between different eigenvector basis. With our current implementation, we can only run the experiment on dense meshes with input size 1024-256-64 in order to prevent the out-of-memory issue. It's worth noting that our method requires at least $N/2$ eigenvectors (input size 4096-1024-256 for the dense mesh dataset) to obtain the best performance, as described and validated in the main paper, which means that our method has the potential to get better results on dense meshes.

The sparse matrix computation techniques are useful to solve this issue. There has been some research [57, 60] using sparse computation for eigenvector decomposition and spectral signal processing. Implementing our Laplacian2Mesh with the sparse matrix computation techniques will further improve the performance.

References

[1] Adobe. Adobe fuse 3d characters. https://www.mixamo.com, 2016. 8

[2] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. *ACM Trans. Graph.*, 24:408–416, 2005. 8

[3] F. Bogo, J. Romero, M. Loper, and M. J. Black. Faust: Dataset and evaluation for 3d mesh registration. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014. 8

[4] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. *Computer Graphics Forum*, 34, 2015. 3

[5] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *ArXiv*, abs/1608.04236, 2016. 1, 3

[6] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. Graph.*, 30:1:1–1:20, 2011. 4

[7] M. M. Bronstein and A. M. Bronstein. Shape recognition with spectral distances. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:1065–1071, 2011. 4

[8] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velivckovi'c. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *ArXiv*, abs/2104.13478, 2021. 2

[9] M. M. Bronstein, J. Bruna, Y. LeCun, A. D. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34:18–42, 2017. 2, 4

[10] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv: Learning*, 2016. 7

[11] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016. 4

[12] J. B. Estrach, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and deep locally connected networks on graphs. In *2nd International conference on learning representations, ICLR*, volume 2014, 2014. 4

[13] D. Ezuz, J. M. Solomon, V. G. Kim, and M. Ben-Chen. Gwcnn: A metric alignment layer for deep shape analysis. *Computer Graphics Forum*, 36, 2017. 8

Table A.2. The mesh segmentation accuracy (Acc.) and the running time on the dense [25] and sparse [68] COSEG-Chairs dataset. The running time includes the preprocessing time (Pre.), training time (Train), and testing time (Test), averaged for each single input of the specified input size.

Input Size	Chairs		Acc.	Times		
	Dense [25]	Sparse [68]		Pre.	Train	Test
128 – 64 – 16		✓	90.5%	520ms	285ms	243ms
256 – 64 – 16		✓	96.6%	520ms	391ms	271ms
256 – 64 – 16	✓		89.9%	5830ms	435ms	281ms
1024 – 256 – 64	✓		95.7%	5830ms	887ms	452ms

[14] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8279–8286, 2019. 8

[15] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018. 3

[16] D. Giorgi, S. Biasotti, and L. Paraboschi. Shape retrieval contest 2007: Watertight models track. *SHREC competition*, 8(7):7, 2007. 8

[17] M.-H. Guo, J. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S. Hu. Pct: Point cloud transformer. *Comput. Vis. Media*, 7:187–199, 2021. 3, 8

[18] N. Haim, N. Segol, H. Ben-Hamu, H. Maron, and Y. Lipman. Surface networks via general covers. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 632–641, 2019. 3

[19] R. Hanocka, N. Fish, Z. Wang, R. Giryes, S. Fleishman, and D. Cohen-Or. Alignet: Partial-shape agnostic alignment via unsupervised learning. *ACM Trans. Graph.*, 38:1:1–1:14, 2019. 3

[20] 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)*, 38:1 – 12, 2019. 1, 2, 3, 5, 8, 9, 12, 13

[21] R. Hanocka and H.-T. D. Liu. An introduction to deep learning on meshes. *ACM SIGGRAPH 2021 Courses*, 2021. 2

[22] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 558–567, 2018. 11

[23] W. He, Z. Jiang, C. Zhang, and A. M. Sainju. Curvanet: Geometric deep learning based on directional curvature for 3d shape analysis. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020. 2, 3

[24] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:2011–2023, 2020. 2, 4, 6

[25] S.-M. Hu, Z.-N. Liu, M.-H. Guo, J.-X. Cai, J. Huang, T.-J. Mu, and R. R. Martin. Subdivision-based mesh convolution networks. *ACM Trans. Graph.*, 41(3), mar 2022. 2, 3, 5, 8, 12, 13, 15, 16

[26] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. In *SIGGRAPH 2010*, 2010. 8

[27] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000. 4

[28] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM SIGGRAPH 2003 Papers*, 2003. 4

[29] T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2017. 6

[30] R. Klokov and V. S. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 863–872, 2017. 1

[31] A. Lahav and A. Tal. Meshwalker: Deep mesh understanding by random walks. *ACM Trans. Graph.*, 39:263:1–263:13, 2020. 1, 3, 8, 13

[32] L. J. Latecki and R. Lakämper. Shape similarity measure based on correspondence of visual parts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:1185–1190, 2000. 13

[33] T. Le, G. Bui, and Y. Duan. A multi-view recurrent neural network for 3d mesh segmentation. *Comput. Graph.*, 66:103–112, 2017. 1, 3

[34] R. B. Lehoucq, D. C. Sorensen, and C. Yang. Arpack users' guide - solution of large-scale eigenvalue problems with implicitly restarted arnoldi methods. In *Software, environments, tools*, 1998. 14

[35] B. Lévy. Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry. *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 13–13, 2006. 14

[36] B. Lévy and H. Zhang. Spectral mesh processing. In *SIGGRAPH '10*, 2010. 2, 4, 6

[37] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018. 3

[38] 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. M. 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 *3DOR@Eurographics*, 2011. 8

[39] R. Litman and A. M. Bronstein. Learning spectral descriptors for deformable shape correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:171–180, 2014. 6

[40] S. Liu, H. Guo, H. Pan, P.-S. Wang, X. Tong, and Y. Liu. Deep implicit moving least-squares functions for 3d reconstruction. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1788–1797, 2021. 1

[41] C. T. Loop. Smooth subdivision surfaces based on triangles. In *Masters Thesis, Department of Mathematics, University of Utah (January 1987)*, 1987. 8

[42] 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 Transactions on Graphics (TOG)*, 36:1 – 10, 2017. 3, 8, 9, 11, 12

[43] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 832–840, 2015. 2, 3

[44] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath*, 2002. 4

[45] F. Milano, A. Loquercio, A. Rosinol, D. Scaramuzza, and L. Carlone. Primal-dual mesh convolutional neural networks. *ArXiv*, abs/2010.12455, 2020. 2, 3, 7, 8, 9, 13

[46] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas. Functional maps: A flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4), jul 2012. 4

[47] M. Ovsjanikov, J. Sun, and L. J. Guibas. Global intrinsic symmetries of shapes. *Computer Graphics Forum*, 27, 2008. 4

[48] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Exp. Math.*, 2:15–36, 1993. 12, 14

[49] C. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 1, 3, 9

[50] C. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 1, 3, 8, 9, 13

[51] Y.-L. Qiao, L. Gao, J. Yang, P. L. Rosin, Y.-K. Lai, and X. Chen. Learning on 3d meshes with laplacian encoding and pooling. *IEEE Transactions on Visualization and Computer Graphics*, 28:1317–1327, 2022. 4, 5

[52] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6620–6629, 2017. 3

[53] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[54] R. M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on Geometry Processing*, 2007. 4

[55] S. C. Schonsheck, B. Dong, and R. Lai. Parallel transport convolution: A new tool for convolutional neural networks on manifolds. *ArXiv*, abs/1805.07857, 2018. 3

[56] J. Schult, F. Engelmann, T. Kontogianni, and B. Leibe. Dualconvmesh-net: Joint geodesic and euclidean convolutions on 3d meshes. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8609–8619, 2020. 3

[57] N. Sharp, S. Attaiki, K. Crane, and M. Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Trans. Graph.*, 41(3), mar 2022. 2, 3, 5, 8, 9, 12, 15

[58] N. Sharp and K. Crane. A laplacian for nonmanifold triangle meshes. *Computer Graphics Forum*, 39, 2020. 12, 14

[59] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *ECCV*, 2016. 3

[60] D. Smirnov and J. M. Solomon. Hodgenet: Learning spectral geometry on triangle meshes. *ACM Trans. Graph.*, 40:166:1–166:11, 2021. 8, 9, 15

[61] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 945–953, 2015. 1, 3

[62] J. Sun, M. Ovsjanikov, and L. J. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum*, 28, 2009. 2, 4, 5

[63] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Gwak, and S. Savarese. Segcloud: Semantic segmentation of 3d point clouds. *2017 International Conference on 3D Vision (3DV)*, pages 537–547, 2017. 1, 3

[64] H. Thomas, C. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6410–6419, 2019. 3

[65] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM SIGGRAPH 2008 papers*, 2008. 8

[66] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. O-cnn. *ACM Transactions on Graphics (TOG)*, 36:1 – 11, 2017. 3

[67] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong. Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 37(6), 2018. 1

[68] Y. Wang, S. Asafi, O. M. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31:1 – 10, 2012. 8, 9, 15, 16

[69] Y. Wang and J. M. Solomon. Intrinsic and extrinsic operators for shape analysis. *Handbook of Numerical Analysis*, 2019. 4

[70] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 1, 3, 8

[71] Y. Xiao, Y.-K. Lai, F.-L. Zhang, C. Li, and L. Gao. A survey on deep geometry learning: From a representation perspective. *Computational Visual Media*, 6:113–133, 2020. 2

[72] H. Xu, M. Dong, and Z. Zhong. Directionally convolutional networks for 3d shape segmentation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2717–2726, 2017. 9