Mesh Neural Networks Based on Dual Graph Pyramids

Xiang-Li Li, Zheng-Ning Liu, Tuo Chen, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu, *Senior Member, IEEE*

***Abstract*—Deep neural networks (DNNs) have been widely used for mesh processing in recent years. However, current DNNs can not process arbitrary meshes efficiently. On the one hand, most DNNs expect 2-manifold, watertight meshes, but many meshes, whether manually designed or automatically generated, may have gaps, non-manifold geometry, or other defects. On the other hand, the irregular structure of meshes also brings challenges to building hierarchical structures and aggregating local geometric information, which is critical to conduct DNNs. In this paper, we present DGNet, an efficient, effective and generic deep neural mesh processing network based on dual graph pyramids; it can handle arbitrary meshes. First, we construct dual graph pyramids for meshes to guide feature propagation between hierarchical levels for both downsampling and upsampling. Second, we propose a novel convolution to aggregate local features on the proposed hierarchical graphs. By utilizing both geodesic neighbors and euclidean neighbors, the network enables feature aggregation both within local surface patches and between isolated mesh components. Experimental results demonstrate that DGNet can be applied to both shape analysis and large-scale scene understanding. Furthermore, it achieves superior performance on various benchmarks, including ShapeNetCore, HumanBody, ScanNet and Matterport3D. Code and models will be available at https://github.com/li-xl/DGNet.**

***Index Terms*—Geometric understanding, mesh processing, neural networks, shape analysis.**

I. Introduction

DEEP learning has greatly facilitated the development of various fields, including image processing, natural language processing, speech recognition, etc. Recently, it has played an increasingly important role in geometric understanding. Since the voxel-based approach of ShapeNets [1] was proposed, neural-network-assisted geometric understanding methods have flourished. Following PointNet [2], many works [3], [4], [5] began to utilize point representations for geometric understanding. Now, mesh-based methods [6], [7], [8], [9] have exploited the topological information of meshes to further improve the effectiveness of neural networks.

Point clouds provide raw 3D geometric information but lack the topological structure of mesh representations. Therefore, more and more works [10], [11], [12] are using mesh representation to improve the performance for geometric understanding. The major difference between processing meshes and 2D images is that regularity is an inherent feature of 2D images, while 3D meshes have an irregular structure. The regular structure of 2D images allows us to quickly build hierarchical structures and perform local feature aggregation on images [13], [14], two fundamental operations for deep image understanding.

Current mesh-based neural networks [6], [15] usually assume meshes to be of high quality, and in particular 2-manifold and watertight. However, both manually designed and automatically generated meshes can suffer from defects of many kinds e.g., non-manifold faces, cracks, disconnected pieces, and self-intersecting surfaces. Mesh-based neural networks need to be robust in the presence of such topological issues, and other mesh quality problems: feature aggregation and hierarchical structure generation should work even given imperfect input.

The main challenge for neural networks on meshes is to generate a hierarchical structure for the irregular input data. Various works [6], [8], [9] employ edge collapse to generate a hierarchical structure, but this naive approach can not guarantee a consistent enlarged receptive field and constant topology. Furthermore, simplification failures may occur, and mappings between hierarchical levels may be ill-defined. Recently, SubdivNet [15] proposed a different architecture based on the Loop subdivision to ensure the consistency of receptive fields during convolution and pooling. This method, however, is rather restrictive as it requires all meshes to be initially remeshed before subdivision and only can be applied to closed 2-manifold meshes.

Though graph neural network [16] is directly applied to the mesh, the number of neighbor nodes aggregated by each node is inconsistent, which will affect the network performance. We find that the dual graph of the mesh, which takes faces as nodes and adjacencies between faces as edges, can ensure that each node has three neighbor nodes in most cases. This can make the receptive field of each node consistent, just like the images, which will help us further build convolution and pooling operations. In addition, we also need to ensure consistent receptive fields

Manuscript received 14 October 2022; revised 23 February 2023; accepted 7 March 2023. Date of publication 14 March 2023; date of current version 26 June 2024. This work was supported in part by National Key R&D Program of China under Grant 2021ZD0112902, in part by the Natural Science Foundation of China under Grant 62220106003, in part by Research Grant of Beijing Higher Institution Engineering Research Center and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. Recommended for acceptance by A. Vaxman. *(Corresponding author: Shi-Min Hu.)*

Xiang-Li Li, Tuo Chen, Tai-Jiang Mu, and Shi-Min Hu are with the BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: lixl19@mails.tsinghua.edu.cn; 2509976039@qq.com; taijiang@tsinghua.edu.cn; shimin@tsinghua.edu.cn).

Zheng-Ning Liu is with the Fitten Tech Co., Ltd., Beijing 100080, China (e-mail: lzhengning@gmail.com).

Ralph R. Martin is with the School of Computer Science and Informatics, Cardiff University, CF10 3AT Cardiff, U.K. (e-mail: martinrr@cardiff.ac.uk).

Digital Object Identifier 10.1109/TVCG.2023.3257035

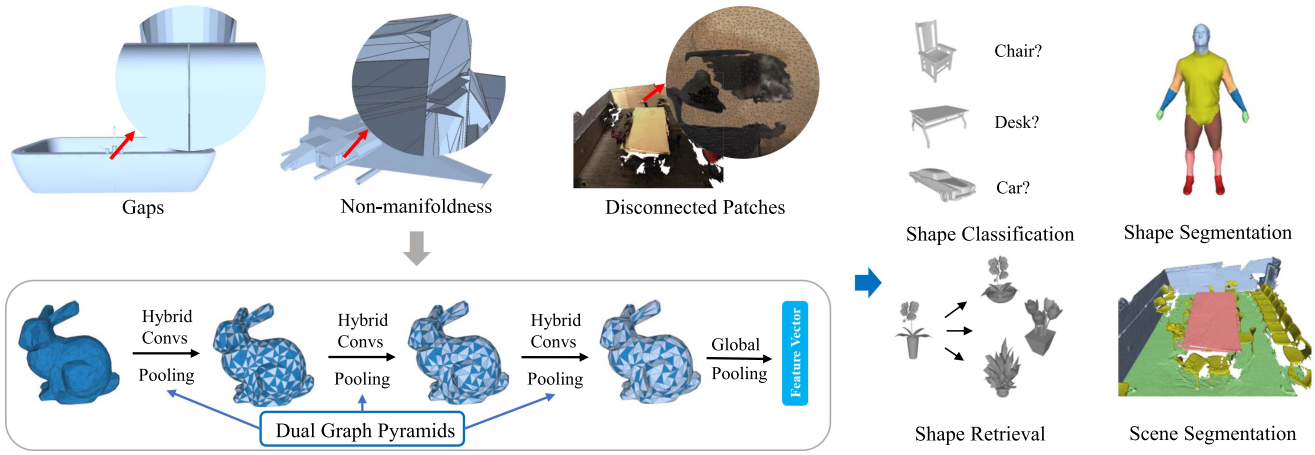

Fig. 1. DGNet, our general mesh network for geometric understanding based on dual graph pyramids. Given a possibly low-quality mesh as input, it can generate dual graph pyramids of the mesh for upsampling and downsampling operators. Taking advantage of hybrid convolution, it can aggregate geodesic features and euclidean features for the mesh. This efficient and effective network architecture can be used for various mesh analysis tasks.

and clear correspondence between different levels of feature propagation. Therefore, we sample the dual graph of the mesh to build a hierarchical structure for an arbitrary mesh. It can then be used for downsampling and upsampling while ensuring that the receptive field can be enlarged consistently during feature propagation.

In addition to the hierarchical structures, the feature aggregation method used on a mesh also has a great impact on network performance. Rotation-invariant convolutions on surfaces [6], [7], [12] can be used to aggregate features over neighborhoods, while other methods [17], [18] rely on parameterization methods to extract geometric features. However, both approaches have either strict requirements on mesh quality or suffer from poor feature aggregation. In contrast, we propose a concise and efficient feature aggregation method that can be widely and robustly applied to meshes of any quality. Specifically, we use a hybrid convolution, which finds geodesic neighbors and euclidean neighbors for local feature aggregation.

Benefiting from the dual graph pyramids and hybrid convolution, our network, named *DGNet*, provides a simple and general approach to handle meshes of arbitrary quality. It achieves state-of-the-art performance on a variety of geometric understanding tasks, including classification, retrieval and segmentation. Ablation studies demonstrate the effectiveness of the proposed hierarchical graph generation strategy, the hybrid convolution, and the overall mesh network.

The contributions of this paper can be summarized as follows:

- a general dual graph pyramids generation method that can generate an approximately uniform dual graph hierarchy for any mesh, as a basis for neural network processing;
- a hybrid convolution operation, which can learn both geodesic features and euclidean features, enhancing the network's capabilities and versatility;
- a simple, generic mesh network, which can process meshes of varying quality, while providing state-of-the-art performance for multiple geometric understanding tasks.

II. Related Work

A. Neural Networks on Multi-Views, Points and Voxels

Following successful applications of deep learning to 2D images, many researchers [2], [6], [19], [20] have considered applying neural networks to 3D geometric understanding. 3D data has more varied representations than 2D data [21]; these include multi-view images, point clouds, voxels, meshes, and so on. Pioneers [22], [23], [24], [25] applied 2D convolutional neural networks to 3D shapes using multiple 2D views (multi-views) on which feature extraction was performed, finally integrating multiple features for various downstream tasks.

With further improvements in neural networks, neural networks have been gradually applied directly to 3D objects [1], [2], [26]. Unlike images, point clouds are unstructured and unordered. To apply 3D convolutional neural networks to them, the easiest approach is to convert the input to a structured, voxel representation [1], [19], [27], [28]. As the voxels are regularly ordered, networks used for 2D image processing can be directly extended to them [29], [30] by using 3D convolutions. However, the increased number of voxels in 3D compared to pixels in 2D leads to significantly greater computational and memory requirements. Choy et al. [20] and Graham et al. [31] thus apply sparse convolution to reduce redundant computation. O-CNN [32] represents the 3D shape as an octree further reducing memory consumption and computation. [33], [34], [35] improve upon this approach by using a patch-guided partitioning strategy and output-guided skip-connections.

Because of the high computational cost of using voxels, other works [2], [3], [33] have explored methods of directly utilizing point clouds. PointNet [2] addresses the lack of structure in point clouds by using multilayer perceptrons and max pooling, while PointNet++ [3] considers a hierarchical structure to improve the performance of point-based networks by set abstraction. PointCNN [4] proposes a convolution operator for point clouds, which finesses their unstructured nature through a feature transformation matrix, while KPConv [36] suggests

deformable convolutions to further improve the performance of point networks. Recent works [37], [38], [39] use an attention mechanism to construct self-attention networks to enhance the results of point cloud processing. In addition, there has been a series of works [21], [40] summarizing problems and methods in deep geometric learning.

B. Neural Networks on Meshes and Graphs

The above approaches are hampered by their high computational cost or lack of topological information, so many works [6], [7], [41], [42], [43], [44], [45] have considered applying neural networks to meshes. Some researchers [46], [47], [48], [49] apply 2D convolutions to meshes by using parametric methods to map geometric features to a grid structure. TangentConv [18] proposes the concept of tangent convolution, which enables neural networks to be applied to large-scale meshes. TextureNet [17] improves the performance of neural networks by introducing a convolution that is invariant to the 4-RoSy ambiguity. PFCNN [50] further improves the capabilities of neural networks on surfaces by using parallel frames.

However, these methods usually have high computational costs and are insensitive to certain types of geodesic features. Therefore, many works [42], [51] treat the meshes as a graph and apply graph convolutional networks. MeshNet [7] directly uses 1-ring adjacency relationships of faces to perform surface convolution, but unlike neural networks on images, does not build a hierarchical structure for pooling. DiffusionNet [10] and HodgeNet [11] further explore the application of spectral methods on meshes to learn geometric features. Some works [6], [8], [9], [52] do build a hierarchical structure by using mesh simplification, which improves the capability of networks. Other efforts have tried new ways of building hierarchical structures, like random walks [41], Loop subdivision [15], parallel vertex clustering [52] and adaptive edge contraction [6]. However, these simplification methods [6], [8] do not guarantee consistent receptive fields for networks and there is no clear mapping between levels. The subdivision-based method [15] has high requirements on meshes, which limits the application of mesh networks. PD-MeshNet [12] proposes a primal-dual framework for mesh learning, and it constructs the primal graph and the dual graph from meshes, and then aggregates the features of the two graphs. Different from them, we build the dual graph pyramids without simplification methods, providing a consistent receptive field for the network and clear mapping for downsampling and upsampling operators. It can be applied to low-quality meshes while ensuring a consistent receptive field for networks.

Learning local features is indispensable to the success of neural networks for geometric understanding. Many works [6], [7], [8], [9], [53] explore local feature aggregation methods for meshes. MeshNet [7] proposes neural networks operating on a surface by aggregating features of 1-ring neighbors, while MeshCNN [6] takes advantage of edge features, using edge convolution for geometric understanding. SubdivNet [15] proposes a general mesh convolutional network based on closest manifold meshes. As well as geodesic features, euclidean features are also important for geometric understanding. There are also researches [8], [9] that combine the advantages of both by aggregating geodesic features and euclidean features. DCM-Net [8] proposes dual convolutions to fuse features on graphs with features in euclidean spaces. VMNet [9] improves the performance of mesh neural networks by combining the sparse voxel-based method and the graph method. In contrast, we propose a novel convolution in which we use the idea of space division to aggregate mesh features into voxels, and then from the voxels into faces. It can make the network aware of the features of different directions and reduce the computation of feature propagation.

The successes of the prior works [6], [7], [8], [15] have confirmed the importance of hierarchical structure, while more recent works [8], [9], [52] further demonstrate that combining geodesic features and euclidean features is advantageous. We incorporate both ideas in our DGNet, which overcomes the limitations of previous works, generating dual graph pyramids and making full use of geodesic features and euclidean features. Thus, our DGNet has a consistent receptive field by using hierarchical graphs and can capture the feature of the isolated patches. Compared to [6], [8], [15], DGNet is more versatile and provides stronger feature extraction capabilities.

III. Dual Graph Pyramids for Meshes

A. Considerations

The success of neural networks in 2D image processing depends on constructing hierarchical structures. Adjacency relationships and hierarchical structures are trivially captured from images, but since triangle meshes are irregular, for neural networks to process meshes, we need to explicitly build a hierarchical structure and appropriate adjacency relationships.

For convolutional neural networks on images, the same filter kernels are applied to every pixel, and the consistent receptive field helps to improve the network performance. The same is true for mesh-based neural networks, so our method is designed to generate dual graph pyramids that can ensure the approximate consistency of receptive fields of faces. Generally, for a mesh, more complex regions require more faces, while simpler regions require fewer faces, and complex regions tend to be more distinctive. Therefore, it is beneficial to oversample regions of interest instead of sampling the surface uniformly. Finally, clear correspondences between the hierarchical graphs will be desirable for the propagation of features at different levels for downsampling and upsampling operators.

Following the above requirements, we propose a hierarchy generation method on the dual graph of the mesh. Our algorithm can use meshes of any quality as input and generates dual graph pyramids through sampling and adjacency construction. In the dual graph, each node corresponds to a face of the mesh and two nodes are connected by an edge if and only if the corresponding faces are adjacent on the mesh. Although connectivity in meshes may differ, the degree of the dual graph is typically relatively uniform and close to 3. Therefore, we can generate the next level graph by sampling almost uniformly, and building adjacency with a consistent degree.

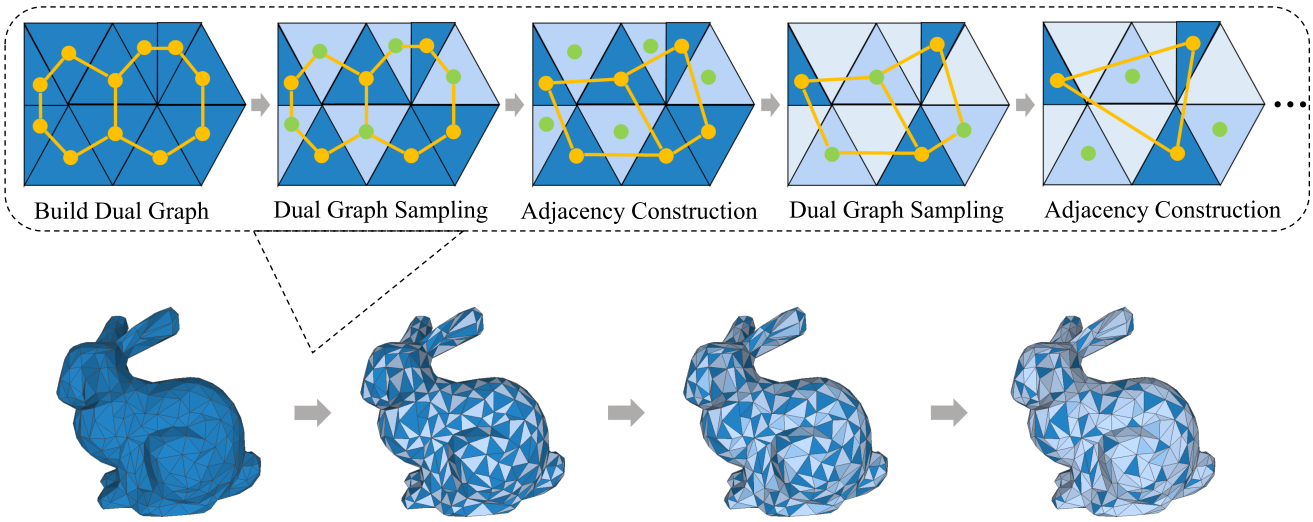

Fig. 2. Dual graph pyramids generation. **(Top)** Schematic diagram of pyramids generation. **(Bottom)** Sampling state visualized on an example mesh. Dark blue: faces to retain. Medium blue: faces to discard at the next level. Light blue: discarded faces. Yellow: nodes to retain. Green: nodes to discard. We construct the dual graph of the mesh, sample it, and rebuild the adjacency relations to generate a new dual graph at the next level. We build dual graph pyramids by alternate sampling and adjacency construction.

B. Dual Graph Construction

Before describing our method, we first recap dual graph construction for a mesh.

Let a triangle mesh M be defined as (V, E, F), where $V = \{v_i \mid v_i \in \mathbb{R}^3\}$ is a set of vertices, $E = \{e_i \mid e_i \in \{1, \ldots, |V|\}^2\}$ is a set of pairs of vertices, or edges, and $F = \{f_i \mid f_i \in \{1, \ldots, |V|\}^3\}$ is a set of triplets of vertices, or faces. Our method uses faces to hold features; the adjacency relationships of faces are used to build hierarchical graphs.

We say that two faces sharing an edge are *adjacent*. Fig. 2 shows the unweighted dual graph G for a mesh M, where each face of M is a node and two nodes are connected by an edge if and only if their corresponding faces are adjacent in M. Faces with more than three adjacent faces are rare, and in such cases, we just randomly select three of the adjacent faces. Although many meshes have non-manifold or disconnected faces, the unweighted dual graph can always be generated. If the mesh has several unconnected components or gaps, the dual graph is not fully connected, and thus cannot be directly used to aggregate features from disconnected components. We propose a hybrid convolution in Section IV-A to solve this problem.

Algorithm 1: Sampling.

Input : initial dual graph G with N faces
Output: the sampling state S
initialize the queue Q and the sampling state S;
for $i \leftarrow 1$ **to** N **do**
  if *face* f_i *is not visited* **then**
    add f_i into Q and set S_i to $retained$;
    while Q *is not empty* **do**
      take f_{now} from Q ;
      foreach *unvisited face* f_{next} *adjacent to* f_{now} **do**
        add f_{next} into Q;
        if f_{next} *has an adjacent face whose state is* $discarded$ **then**
          set S_{next} to $retained$;
        else
          set S_{next} to $discarded$;
        end
      end
    end
  end
end

C. Dual Graph Pyramids Generation

To provide a hierarchical structure for mesh neural networks, we need to build a pyramid from the dual graph of the mesh, as is done for images. Building this graph hierarchy for a mesh proceeds in two key steps: sampling, and adjacency construction, as shown in Fig. 2.

Sampling. Given the dual graph G of the mesh, we first set the sampling state S, *retained* or *discarded* at the next level, for each node in G. As shown in Algorithm 1, we first set up an empty queue Q, and select an unvisited node f_i from the graph G to add to the queue; its status is set to *retained*. Then, we take the first node f_{now} from the queue. For each unvisited node f_{next} adjacent to f_{now}, we set its state to *retained* if there are visited *discarded* nodes amongst its 1-ring neighbors, and otherwise set its state to *discarded*. We then add it to the queue. We process the nodes in the queue one by one until the queue is empty. Finally, we repeat the above two steps until all nodes have been visited.

Adjacency Construction. Given the dual graph G and the sampling states S as inputs, we build the next level graph G_1 using the *retained* faces in G. As shown in Algorithm 2, we first take all *retained* nodes in G without edges as G_1. Then, for each

Algorithm 2: Adjacency Construction.

Input : initial dual graph G, the sampling state S
Output: the dual graph G_1 of the next level
take the retained faces in G without edges as G_1;
foreach *face* f_i *in* G_1 **do**
  foreach f_{next} *adjacent to* f_i *in* G **do**
    if *the state of* f_{next} *is retrained* **then**
      add edge $\langle f_i, f_{\text{next}} \rangle$ into G_1;
    else
      foreach f_j *adjacent to* f_{next} *in* G **do**
        if f_j *not equal* f_i *and* f_j*'s state is retained* **then**
          add edge $\langle f_i, f_j \rangle$ into G_1;
          break;
        end
      end
    end
  end
end

TABLE I
TIME TAKEN TO BUILD DUAL GRAPH PYRAMIDS

Method	4000 faces	8000 faces	16000 faces	100000 faces
FPS	39.00 ms	149.60 ms	583.65 ms	22629.50 ms
QEM	32.12 ms	65.83 ms	137.75 ms	1328.02 ms
Ours	**1.96 ms**	**3.64 ms**	**7.13 ms**	**56.83 ms**

node f_i in G_1, we check its 1-ring neighbors f_{next} in G, and if f_{next} is *retained*, we add the edge $\langle f_i, f_{\text{next}} \rangle$ to G_1. Otherwise, we randomly choose one of f_{next}'s 1-ring neighbors f_j (other than f_i), whose state is *retained*. If such an f_j exists, we add the edge $\langle f_i, f_j \rangle$ to G_1. After the adjacency relations of all nodes in G_1 have been constructed, we obtain the dual graph G_1 of the next level.

The sampling states S provide a clear mapping between the two dual graphs G and G_1. Using the above two steps, we build a graph hierarchy $G_0, \ldots, G_L$ for the mesh iteratively, starting from $G_0 = G$. The time taken to build the L-layer pyramid is $O(LN)$, where N is the number of faces in G. Table I compares the speed of building dual graph pyramids by using quadric error metrics (QEM), farthest point sampling (FPS) and our method. The QEM approach builds a hierarchical mapping by recording the deleted faces, and adjacency relations come from the dual graph of the simplified mesh. In the FPS, we replace our sampling method with the furthest point sampling, and use the same adjacency construction method. Table I gives times for generating a 5-layer pyramid by the above methods on an AMD EPYC 7302 CPU for the *bunny* meshes containing 4000, 8000, 16000 and 100000 faces, respectively. Compared to FPS and QEM, our method is much faster.

This approach can build a graph hierarchy for aggregating, downsampling and upsampling any mesh, even if it is non-manifold or has disconnected pieces. Our approach has two advantages. First, our sampling method distributes the retained faces evenly on the mesh, helping to ensure that the neural network has a consistent receptive field. Second, there is a clear mapping between different levels of the hierarchical structure. This helps to define how features are propagated during upsampling and downsampling, with clear benefits as demonstrated by our ablation study.

IV. MESH NEURAL NETWORKS BASED ON DUAL GRAPH PYRAMIDS

We now show how to construct mesh neural networks by defining basic convolution, downsampling and upsampling operations on the above dual graph pyramids.

A. Hybrid Convolution

Most mesh neural networks [6], [15] can only extract features on a connected surface, and fail to capture relationships between unconnected components, which are commonly found in a shape or a scene. DCM-Net [8] employs a dual convolution to combine geodesic features with euclidean features (i.e., features nearby on the mesh, and nearby in space), and shows that additionally using euclidean features can improve the performance of neural networks on meshes. For example, when there are cracks in the mesh, features across the crack can be aggregated according to their euclidean distance, helping to overcome this common problem of low-quality meshes. Geodesic features are also important. When two objects are nearby (e.g., a table and a chair), euclidean features will not distinguish them, but geodesic features will, as they are on different surfaces.

Inspired by this prior work, to take advantage of both geodesic and euclidean features, we fuse them to fully capture the relationships between faces, using a novel approach we call *hybrid convolution*.

To guarantee order-invariance of feature extraction for many tasks [2], [3], we compute the geodesic feature of a face f_i by aggregating the linearly transformed features of its 1-ring neighbors with the max operation:

$$H_{\text{geo}}(\mathbf{x}_i) = \max_{j \in N_s(i)} (\mathbf{W}_0 \mathbf{x}_j), \tag{1}$$

where $N_s(i)$ is the 1-ring surface neighborhood of face f_i, $\mathbf{W}_0$ is the weight for the linear transformation module.

Self features of face f_i are extracted by another linear module with weight $\mathbf{W}_1$ shared among all faces:

$$H_{\text{self}}(\mathbf{x}_i) = \mathbf{W}_1 \mathbf{x}_i \tag{2}$$

For euclidean features, we first divide the space into voxels of size r. For each voxel v we randomly select m faces whose centers of mass lie within the voxel, and the voxel feature $\mathbf{x}^v$ is computed as the maximum of the features of these m faces. In our experiments, we set $m = 5$ to balance the memory cost and the quality of results, as demonstrated in Section V-H2.

For face f_i, we aggregate the features of its owning voxel and those of that voxel's 6-connected neighbors to obtain its euclidean feature:

$$H_{\text{euc}}(\mathbf{x}_i) = \sum_{k=0}^{6} \mathbf{W}_k^v \mathbf{x}_k^v, \tag{3}$$

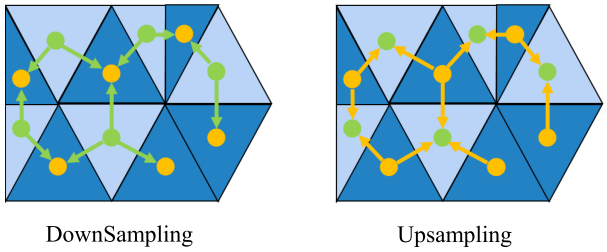

Fig. 3. Downsampling and upsampling based on the dual graph pyramid. Arrows denote the directions of feature propagation. Light blue: faces discarded at the next level. Dark blue: retained faces. Yellow: retained nodes. Green: discarded nodes.

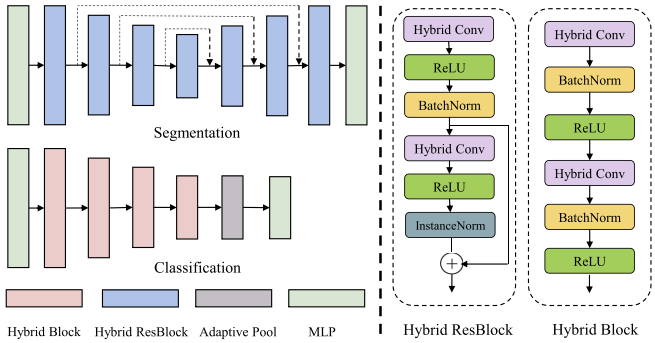

Fig. 4. The classification and segmentation network architectures of DGNet.

where k numbers voxels in the voxel neighborhood of face f_i and $\mathbf{W}_k^v$ is the corresponding weight. For efficiency, if $r <= 0.1$, we simply use the central cube feature as the euclidean feature.

We fuse the features by using addition to obtain a hybrid feature $\mathbf{y}_i$ for face f_i:

$$\mathbf{y}_i = H_{\text{self}}(\mathbf{x}_i) + H_{\text{geo}}(\mathbf{x}_i) + H_{\text{euc}}(\mathbf{x}_i). \tag{4}$$

We also apply channel attention [54] to this hybrid feature to further improve its capability for feature extraction.

We use voxel neighborhoods as doing so is faster than using the nearest neighbor search. We further use 6-connected neighbors rather than 26-connected neighbors again for speed and a reduced number of parameters. Unlike KPConv [36], we use fixed directions to construct neighborhood features, and randomly select faces in each direction to ensure the generality of the network. This fixed approach can be executed quickly and efficiently, and random selection also prevents the network from overfitting. While euclidean features are sensitive to the rotation of input features, we mitigate this shortcoming by randomly selecting faces and combining them with the geodesic features.

Compared to DCM-Net [8] which simply aggregates the euclidean features from neighbors within a spherical neighborhood, we use the idea of spatial division to get features for each position in space and then aggregate those features onto the surface. Our method allows the network to perceive features in different directions, which gives the network a stronger feature extraction capability. Further, by randomly selecting the faces for each voxel, our network has stronger generality and higher efficiency, as demonstrated by later experiments.

B. Downsampling and Upsampling

1) Downsampling: In the downsampling process, features are propagated from G_l to G_{l+1}. This can be regarded as transferring features from the discarded faces to the retained faces as shown in Fig. 3. Let $A_l(i)$ be the set of 1-ring discarded neighbors of face f_i in G_l, and $\Omega(f_i) = \{i\} \cup A_l(i)$ be the aggregation scope of face f_i. We define two kinds of aggregations:

$$P_{\text{max}}(\mathbf{x}_i) = \max_{j \in \Omega(f_i)} (\mathbf{x}_j), \tag{5}$$

$$P_{\text{none}}(\mathbf{x}_i) = \mathbf{x}_i. \tag{6}$$

We use P_{none} as the aggregation method in classification networks and P_{max} in segmentation networks. In the above aggregation approach, we can easily map features from G_l to G_{l+1} by using the state S_l.

2) Upsampling: Fig. 3 shows how features are propagated from G_{l+1} to G_l during upsampling. We first map features from G_{l+1} to G_l using the state S_l for the retained faces, and for faces discarded from G_l to G_{l+1}, we take the mean of the retained features around the discarded face as its feature.

C. Network Architecture

Having the dual graph pyramids built for meshes and convolution, downsampling and upsampling operations defined, we now propose a family of mesh neural networks, named DGNet, which is highly adaptable to the quality of the mesh and able to be applied to various mesh processing tasks.

Our networks use faces to hold features. The input of our networks is a triangular mesh of arbitrary quality which may be non-manifold, disconnected, etc. Our network can learn the global features of the mesh and the local features of each face by taking advantage of geodesic and euclidean features. We show the typical network structures for mesh classification and segmentation in the left of Fig. 4, respectively. Our mesh classification network takes the typical architecture of convolutional neural networks, while our mesh segmentation network is a U-Net like architecture. As shown in the right of Fig. 4, we build the Hybrid Block for classification by using two Hybrid Convolution + BatchNorm + ReLU, and the Hybrid ResBlock for segmentation by using one Hybrid Convolution + BatchNorm + ReLU as the main path and one Hybrid + InstanceNorm + ReLU as the residual path. For Hybrid ResBlock, we place the normalization layer behind the activation layer for segmentation, which can ensure that the data distribution is consistent with that before convolution. Due to the large difference in the number of meshes and faces between datasets, we adjust the number of network layers and channels according to the sizes of the datasets.

V. Experiments and Discussion

In order to evaluate the performance and generality of our DGNet, we conducted experiments on various mesh tasks, including shape classification, shape retrieval, shape segmentation and scene segmentation. In addition, we conducted ablation studies to evaluate the key components of DGNet.

A. Implementation Details

Unlike the pixels in an image, the local geometry of each mesh face differs, so we not only use pose descriptors (center, normal, etc.) of the face, but we also use shape descriptors (angle, area, curvature, etc.) as the input features of the network. For meshes of large scenes, as in ScanNet [55], we also use color as an input feature. In segmentation, some datasets have labelled vertices (such as Humanbody [48], COSEG [56] and ScanNet [55]). We determine the label of each face as the mode(the value that appears most often) of the labels of its three vertices. When calculating evaluation metrics, we determine the label of each vertex as the mode of the labels of the faces containing it.

For the tasks of shape classification, retrieval and segmentation, we applied augmentation via random scaling, random rotation, random translation and normalization during training. For large-scale scene segmentation, we cropped the whole meshes and used elastic distortion, color distortion and random rotation to augment each cropped mesh.

Since the sizes of the datasets are quite different, we set different numbers of layers for the networks according to dataset difficulty. For all tasks, we used the Adam optimizer with weight decay 10^{-4} and an initial learning rate of 10^{-3}. For shape classification and retrieval, the learning rate was decayed by a cosine annealing schedule with minimum learning rate 10^{-4}. For shape segmentation and scene segmentation, the learning rate was decayed by a poly schedule with minimum learning rate 0. All experiments were performed on an AMD EPYC 7302 CPU and GeForce RTX 3090 GPUs. Our models were implemented using Jittor [57], a high-performance deep learning framework.

B. Shape Classification

We verified the effectiveness of DGNet for classification by using ShapeNetCore, Manifold40, SHREC11 and Cube Engraving datasets. We used classification accuracy as the evaluation metric.

1) ShapeNetCore: ShapeNetCore v2 [58] is a challenging and widely used benchmark. It contains 52,472 shapes in 55 common categories. About 52,000 shapes have more than one component and about 50,000 shapes have more than 10 components. More than 37,000 shapes have non-manifold edges shared by more than two faces. Because of these characteristics, most mesh networks [6] cannot be directly applied to this dataset, so we modified DCM-Net [8] to provide a baseline method. We randomly divided the shapes into a training and test split in the ratio 4:1. For the speed of training, we reduced the number of faces to 1024 for all shapes.

Even on this challenging dataset, our method achieved an accuracy of 88.5% as shown in Table II, much better than DCM-Net [8]. This is because our graph hierarchy provides an even receptive field for the network during downsampling, and clear correspondences between levels, which enables the features of each discarded face to be spread to multiple retained faces, improving feature propagation, even for unconnected components.

TABLE II
Classification Accuracy on ShapeNetCore

Method	Accuracy
DCM-Net [8]	85.1%
DGNet (Ours)	**88.5%**

TABLE III
Classification Accuracy on Manifold40

Method	Accuracy
PointNet++ [3]	87.9%
MeshNet [7]	88.4%
MeshWalker [41]	90.5%
SubdivNet [15]	91.2%
DGNet (Ours)	**91.7%**

TABLE IV
Classification Accuracy on Cube Engraving

Method	Accuracy
PointNet++ [3]	64.3%
MeshCNN [6]	92.2%
PD-MeshNet [12]	94.4%
MeshWalker [41]	98.6%
SubdivNet [15]	98.9%
DGNet (Ours)	**99.5%**

2) Manifold40: ModelNet40 [1] is another widely used benchmark for mesh classification. It contains 12,311 shapes in 40 categories and is divided into training and test sets with 9843 and 2468 shapes respectively. Most previous works have used Manifold40 for evaluation, containing reconstructions of manifold meshes from ModelNet40. For comparison, we also used it to train and test our network.

We compare our results with several well-known methods, including PointNet++ [3], MeshNet [7], MeshWalker [41], SubdivNet [15] in Table III. Our method achieves greatest classification accuracy.

3) Cube Engraving: The Cube Engraving dataset [6] has icons engraved on 3D cubes, and contains 4381 shapes in 22 categories. It is divided into training and test sets with 3722 and 659 shapes. As shown in Table IV, DGNet again achieves the best results.

4) SHREC11: SHREC11 [60] has 30 categories, each with 20 shapes. Following previous works, we used 16 or 10 training samples for each class. The results are reported in Table V; DGNet achieved 100% accuracy.

C. Shape Retrieval

SHREC16 [61] is a large-scale shape retrieval dataset from ShapeNet core55. It contains 51300 shapes in 55 categories and 204 subcategories, and is split in the ratio 7:1:2 for training,

TABLE V
CLASSIFICATION ACCURACY ON SHREC11. SHREC11-16 AND SHREC11-10 DENOTE SPLITS 16 AND 10 RESPECTIVELY

Method	SHREC11-16	SHREC11-10
GWCNN [59]	96.6%	90.3%
MeshCNN [6]	98.6%	91.0%
PD-MeshNet [12]	99.7%	99.1%
MeshWalker [41]	98.6 %	97.1%
HodgeNet [11]	99.2%	94.7%
DiffusionNet [10]	-	99.7%
SubdivNet [15]	99.9%	99.5%
DGNet (Ours)	**100%**	**100%**

TABLE VI
RETRIEVAL RESULTS ON SHREC16

Method	P@N	R@N	F1@N	mAP	NDCG@N
CCMLT	0.718	0.350	0.391	0.823	0.886
ViewAggregation	0.508	**0.868**	0.582	0.829	0.904
DB-FMCD-FUL-LCDR	0.427	0.689	0.472	0.728	0.875
GIFT	0.706	0.695	0.689	0.825	0.896
MVCNN	0.770	0.770	0.764	0.873	0.899
O-CNN [32]	0.778	0.782	0.776	**0.875**	0.905
DGNet (ours)	**0.783**	0.785	**0.779**	0.874	**0.913**

validation, and test. For speed of training, we reduced the number of faces to 1024 for all shapes.

The key in mesh retrieval is to extract a descriptive feature for each shape and use this feature to calculate similarities between shapes. Like O-CNN [32], we trained our classification network with 55 main categories and extracted the feature for the classification network after adaptive max pooling as the shape descriptor.

We used the evaluation scripts provided by SHREC16, which are based on five metrics: precision, recall, F1-score, mAP and NDCG. Precision is the ratio of positive samples in the retrieval result, while recall represents the number of positive samples divided by the total number of samples. F1 balances precision and recall. mAP is based on the area under the precision and recall curves. NDCG is the normalized discounted cumulative gain, which represents the quality of the retrieval order. We calculate these metrics: P@N, R@N, F1@N, mAP and NDCG@N, where the N is the total retrieval list length [61].

We report shape retrieval results in Table VI; the first five rows are the results of competitors from [61], the penultimate row is a voxel-based method [32] and the last row is ours. These experiments show that our method achieves competitive results for most metrics, and in particular, our method provides the best retrieval order as assessed by NDCG@N. Fig. 5 shows the Top-5 retrieval results from DGNet.

D. Shape Segmentation

We use the Humanbody [48] and COSEG [56] benchmarks to evaluate DGNet on shape segmentation. The task is to determine a category for each face of the mesh. To enable a fair comparison, we use the dataset processed by [15] and map the predicted results back to the original mesh using the nearest face.

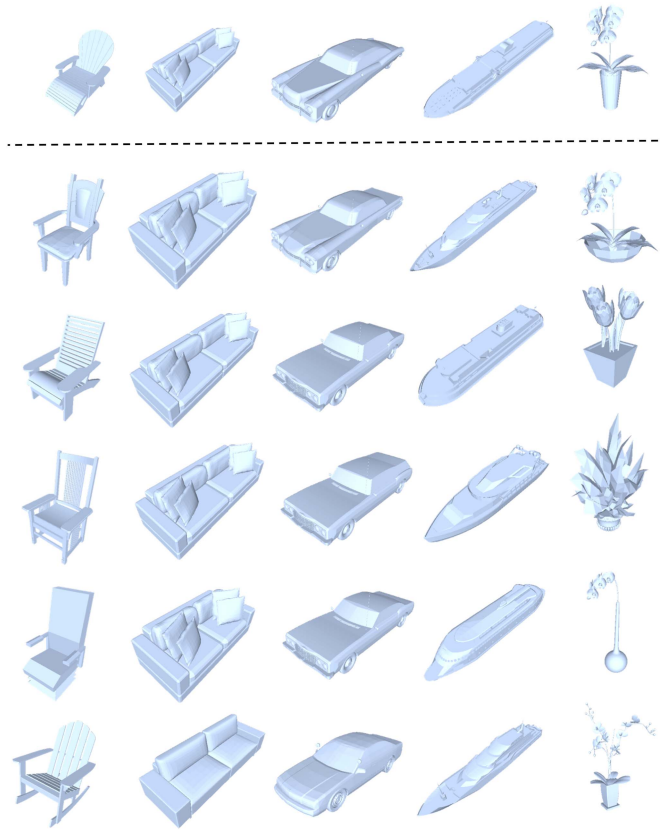

Fig. 5. Top-5 retrieval results of our method on SHREC16. Above: query objects. Below: top 5 retrieval results. Note that the top 4 sofas in the second column differ in scale.

TABLE VII
SEGMENTATION ACCURACY ON HUMANBODY

Method	Accuracy
Pointnet [2]	74.7%
Pointnet++ [3]	82.3%
MeshCNN [6]	87.8%
DiffusionNet [10]	91.7%
PD-MeshNet [12]	86.9%
Toric Cover [48]	88.0%
SNGC [49]	91.3%
PFCNN [50]	91.5%
MeshWalker [41]	92.7%
SubdivNet [15]	93.0%
DGNet (Ours)	**93.2%**

1) HumanBody: The HumanBody dataset [48] contains 399 shapes, 381 for training and 18 for testing; each shape is segmented into 8 parts. Table VII shows that our method achieved the best results. Fig. 6 shows that our method accurately segments the body into different parts with clear boundaries.

2) Coseg: The COSEG dataset [56] has 3 subsets: tele-aliens, chairs, and vases, with 200, 400 and 300 shapes, respectively; each shape is segmented into 3 or 4 parts. Quantitative results are given in Table VIII and example output is shown in Fig. 7. Our method again achieved the best results for all 3 subsets.

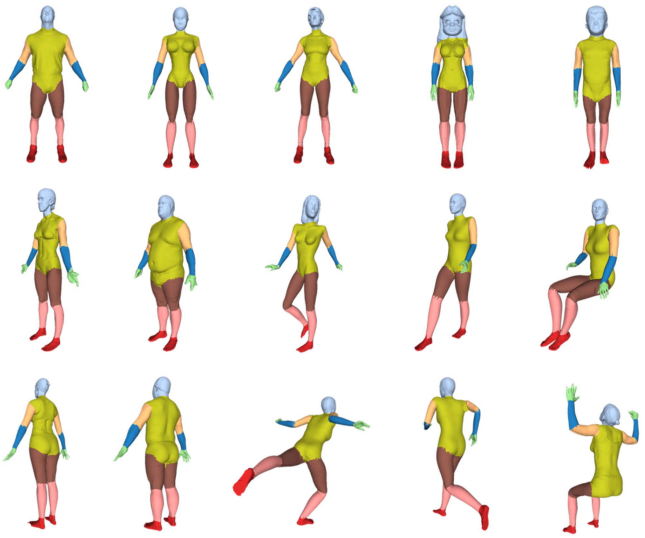

Fig. 6. DGNet segmentation results on Humanbody.

TABLE VIII
SEGMENTATION ACCURACY ON COSEG

Method	Vases	Chairs	Tele-aliens
MeshCNN [6]	85.2%	92.8%	94.4%
PD-MeshNet [12]	81.6%	90.0%	89.0%
SubdivNet [15]	96.7%	**96.7%**	97.3%
DGNet (Ours)	**97.0%**	**96.7%**	**97.4%**

Fig. 7. DGNet segmentation results on COSEG.

E. Scene Segmentation

We used the large-scale scene datasets ScanNet and Matterport3D to demonstrate the generality and robustness of our method. These meshes were reconstructed from point clouds automatically, so generally have lower quality than hand-built datasets, providing a greater challenge for geometric understanding.

Since the number of faces in each scene varies greatly, we simplified each scene according to the ratio of the average face area to a target area A_f, the average area of each face desired after simplification. Assuming that the number of faces is O, the area of each face is A_i, then the number of faces after simplification is:

$$O' = \left\lfloor \sum_{i=0}^{O} A_i / A_f \right\rfloor. \tag{7}$$

During simplification, we record the reduced edges to allow mapping of the predicted results back to the original raw mesh. For each scene in ScanNet, we set the target face area to 10 cm^2. Due to the large number of faces in Matterport3D, we set the target face area to 25 cm^2. After simplification, we randomly cropped the mesh to about 12,000 faces for training.

To demonstrate the ability of our model to handle large meshes, we also conducted experiments without simplification on ScanNet. We voxelized the input meshes with a resolution of 2 cm, and cropped the mesh to about 100,000 faces for training.

1) ScanNet: ScanNet v2 [55] is an indoor scene dataset with a training, validation and test split of 1201:312:100 scenes; each scene contains 20 valid semantic categories. We adopt mean IoU as the evaluation metric, following DCM-Net [8].

As Table IX shows, our method outperforms DCM-Net in mIoU for both large meshes and simplified meshes. Fig. 8 shows some segmentation results from the ScanNet validation dataset.

2) Matterport3D: Matterport3D [64] contains training, validation, and test sets with 61, 11, and 18 buildings, respectively; there are several scenes for each building, labeled with 21 valid categories. Like DCM-Net [8], we used mean accuracy (mAcc) as the evaluation metric. During training, we used the class weights from DCM-Net. We report our experimental results in Table X: our approach provides a significant improvement over DCM-Net.

F. Efficiency

We compared the number of parameters and accuracy of methods on the Simplified Humanbody dataset from HodgeNet [11]. To be lightweight, we reduced the number of channels of DGNet, using 16/24/32 encoder channels and 32/24/24 decoder channels. Results are shown in Table XI, in which we compared our method with the classic mesh network MeshCNN [6] and the lightweight network HodgeNet. We achieve a significant

TABLE IX
MEAN IOU SCORES ON SCANNET TEST SET. * MEANS THAT WE USE SIMPLIFIED MESHES AS INPUT

Method	mIoU	wall	floor	cab	bed	chair	sofa	table	door	wind	shf	pic	cntr	desk	curt	fridg	show	toil	sink	bath	other
PointNet++ [3]	33.9	52.3	67.7	25.6	47.8	36.0	34.6	23.2	26.1	25.2	45.8	11.7	25.0	27.8	24.7	21.2	14.5	54.8	36.4	58.4	18.3
SplatNet [62]	39.3	69.9	92.5	31.1	51.1	65.6	51.0	38.3	19.7	26.7	60.6	0.0	24.5	32.8	40.5	0.0	24.9	59.3	27.1	47.2	22.7
TangentConv [18]	43.8	63.3	91.8	36.9	64.6	64.5	56.2	42.7	27.9	35.2	47.4	14.7	35.3	28.2	25.8	28.3	29.4	61.9	48.7	43.7	29.8
3DMV [63]	48.4	60.2	79.6	42.4	53.8	60.6	50.7	41.3	37.8	53.9	64.3	21.4	31.0	43.3	57.4	53.7	20.8	69.3	47.2	48.4	30.1
TextureNet [17]	56.6	68.0	93.5	49.4	66.4	71.9	63.6	46.4	39.6	56.8	67.1	22.5	44.5	41.1	67.8	41.2	53.5	79.4	56.5	67.2	35.6
DCM-Net [8]	65.8	80.3	94.1	61.9	70.2	81.3	72.7	56.8	52.4	63.7	**80.6**	**29.8**	46.8	49.4	69.3	51.0	**82.1**	82.6	67.5	77.8	44.9
DGNet* (Ours)	67.2	83.6	94.5	61.5	75.2	81.4	72.1	57.8	56.7	67.4	76.3	24.4	47.6	51.4	77.4	**59.0**	75.5	**92.4**	**68.7**	**78.2**	43.6
DGNet (Ours)	**68.4**	**84.6**	**95.0**	**65.8**	**78.4**	**83.5**	**76.4**	**62.0**	**59.7**	**68.8**	78.2	28.1	**49.9**	**64.1**	**82.3**	57.5	61.9	87.1	64.7	71.2	**48.7**

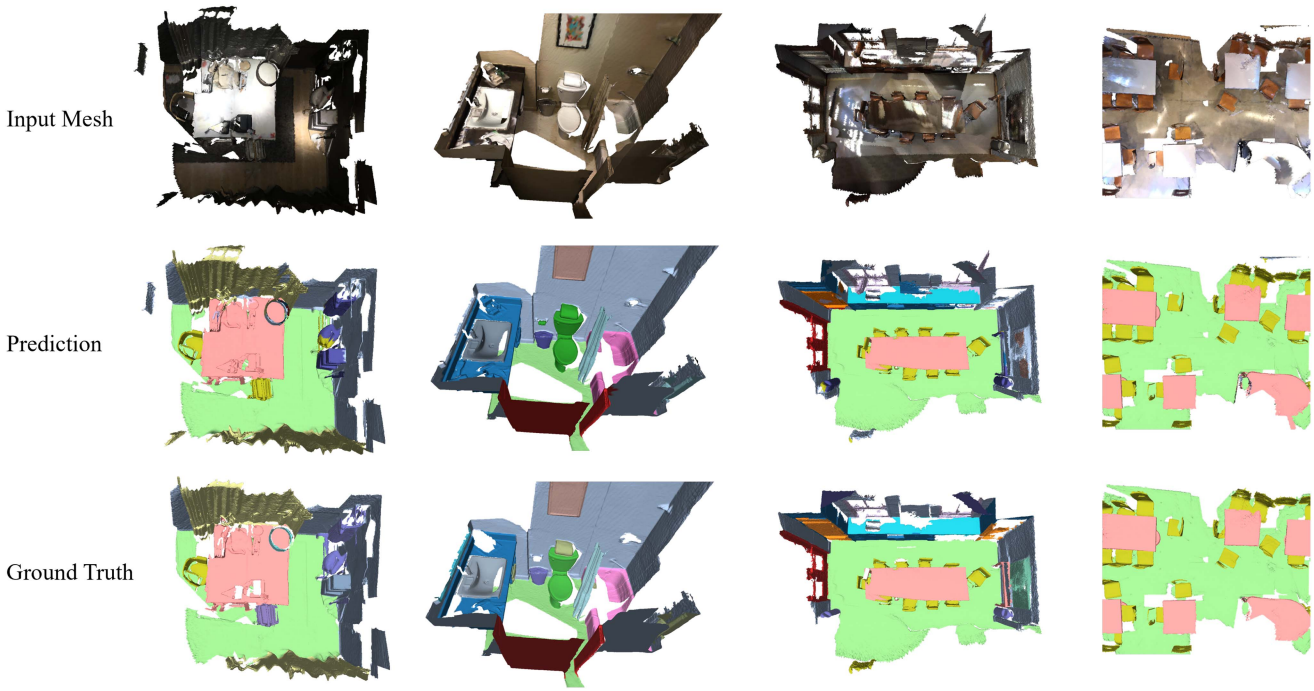

Fig. 8. Example segmentation results from the ScanNet v2 validation set.

TABLE X
MEAN CLASS ACCURACY SCORES ON MATTERPORT3D TEST SET

Method	mAcc	wall	floor	cab	bed	chair	sofa	table	door	wind	shf	pic	cntr	desk	curt	ceil	fridg	show	toil	sink	bath	other
SplatNet [62]	26.7	**90.8**	95.7	30.3	19.9	77.6	36.9	19.8	33.6	15.8	15.7	0.0	0.0	0.0	12.3	75.7	0.0	0.0	10.6	4.1	20.3	1.7
TangentConv [18]	46.8	56.0	87.7	41.5	73.6	60.7	69.3	38.1	55.0	30.7	33.9	50.6	38.5	19.7	48.0	45.1	22.6	35.9	50.7	49.3	56.4	16.6
3DMV [63]	56.1	79.6	95.5	59.7	82.3	70.5	73.3	48.5	64.3	55.7	8.3	55.4	34.8	2.4	**80.1**	**94.8**	4.7	54.0	71.1	47.5	76.7	19.9
TextureNet [17]	63.0	63.6	91.3	47.6	82.4	66.5	64.5	45.5	69.4	60.9	30.5	**77.0**	**42.3**	44.3	75.2	92.3	49.1	66.0	80.1	60.6	86.4	27.5
DCM-Net [8]	66.2	78.4	93.6	**64.5**	89.5	70.0	**85.3**	46.1	**81.3**	**63.4**	43.7	73.2	39.9	**47.9**	60.3	89.3	**65.8**	43.7	**86.0**	49.6	**87.5**	**31.1**
DGNet (Ours)	**68.3**	78.2	**95.8**	63.4	**89.6**	**80.9**	67.5	**65.2**	77.0	62.9	**52.8**	73.4	34.2	45.8	79.5	94.6	40.7	**92.2**	84.1	**62.9**	82.0	11.3

TABLE XI
PERFORMANCES OF LIGHTWEIGHT APPROACHES ON THE SIMPLIFIED HUMANBODY DATASET

Method	Accuracy	Parameters	Time	GPU Memory
MeshCNN	85.39%	2,279,720	201.7 ms	5327 MiB
HodgeNet	85.03%	31,720	167.4 ms	-
DGNet (Ours)	**88.00%**	**28,712**	**6.2 ms**	**3445 MiB**

improvement with fewer parameters. We also measured the iteration time for each sample and GPU memory consumption during training on an AMD EPYC 7302 CPU and a single GeForce RTX 3090 GPU (HodgeNet only executes on the CPU so needs no GPU memory). Our method only takes 5 minutes to train for 150 epochs.

G. Generality

In addition to its strong performance on manifold mesh datasets, DGNet also shows its ability to perform well on benchmarks with imperfect meshes. ShapeNetCore is a well-known low-quality mesh benchmark, containing many non-manifold faces, gaps and self-intersecting faces. Table II shows that DGNet handles ShapeNetCore well, while, most methods such as MeshCNN [6], MeshNet [7], and SubdivNet [15] can not

TABLE XII
MIOU OF DIFFERENT HIERARCHY GENERATION METHODS ON SCANNET

Method	Retention Ratio	mIoU
VC	4/8/16/32/64 cm	58.9
QEM	30%	58.5
QEM	50%	63.4
QEM	70%	64.1
FPS	30%	57.3
FPS	50%	58.7
FPS	70%	61.8
Ours	-	**67.5**

TABLE XIII
ABLATION STUDY FOR HYBRID CONVOLUTION. LEFT: RESULTS ON SCANNET. RIGHT: RESULTS ON MANIFOLD40

Method	mIoU
Geo Only	54.9
Euc Only	65.6
DGNet(Geo+Euc)	**67.5**

Method	Accuracy
Geo Only	91.4%
Euc Only	91.2%
DGNet(Geo+Euc)	**91.7%**

TABLE XIV
MIOU OF DIFFERENT AGGREGATION METHODS IN THE PROPOSED HYBRID CONVOLUTION ON SCANNET

Method	Aggregation size	Time	GPU Memory	mIoU
Euc	1	19.4 ms	15395 MiB	64.2
Euc	5	27.7 ms	15529 MiB	65.6
Euc	10	42.0 ms	15685 MiB	65.1
Geo+Euc	1	34.3 ms	17245 MiB	67.0
Geo+Euc	5	43.0 ms	17431 MiB	67.5
Geo+Euc	10	56.9 ms	17677 MiB	67.0
Geo+Euc	20	82.6 ms	18229 MiB	**67.7**

handle this dataset due to the nature of its data. Furthermore, as shown in Tables IX and X, our method works well for large-scale real scene mesh data, again demonstrating the ability of DGNet to cope with a wide range of kinds of input.

H. Ablation Study

We conducted ablation experiments using the ScanNet v2 validation set and Manifold40 test set to demonstrate the contributions of our dual graph pyramid and hybrid convolution. Unlike Manifold40, ScanNet has many disconnected patches.

1) Dual Graph Pyramids: When building a hierarchical structure from a mesh, previous work typically used a mesh simplification algorithm [8] or Loop subdivison [15]. However, the subdivision method places a very strict requirement on the input meshes, and therefore cannot be applied to ScanNet. Two commonly used algorithms for mesh simplification are based on vertex clustering (VC) [65] or quadric error metrics (QEM) [66]. We also compared our method to the popular, uniform, farthest point sampling algorithm (FPS) [3].

For the VC algorithm, we set the grid size to 4, 8, 16, 32, and 64 cm. For the QEM algorithm, we set the retention ratio for each layer to 30%, 50%, and 70%. Since there is no obvious correspondence between the levels using mesh simplification, we adopted nearest neighbors to establish the mapping relationship, with upsampling and downsampling following [8]. We also replaced our sampling method with FPS, while using the same adjacency construction method. In addition, we set the retention ratio for each layer to 30%, 50% and 70%.

Table XII shows that, for both VC and QEM, geometric errors during simplification lead to a decrease in the quality of network accuracy. With an increasing retention rate, the errors caused by mesh simplification become smaller, with consequently improved network accuracy. FPS samples uniformly in space, resulting in some regions of interest retaining too few faces, and with an increasing retention rate, this situation gradually decreases, resulting in increased network performance.

On one hand, our method does not change the structure of the whole mesh in the process of building the graph hierarchy, but only reconstructs the adjacency relationships of faces. Therefore, our method does not introduce any geometric errors, while mesh simplification changes the tessellation of shapes and brings geometric distortion. On the other hand, our method samples uniformly on the dual graph of the mesh, which not only gives the network a consistent receptive field but also enables oversampling of complex regions of the mesh. In addition, there is a clear mapping between levels of the dual graph pyramid, so our method constructs a many-to-many transmission of features in downsampling and upsampling, enhancing the performance of mesh neural networks using it.

2) Hybrid Convolution: To evaluate our hybrid convolution, we performed experiments using only geodesic features, only euclidean features, or both. In addition, we also conducted experiments to explore the influence of the number of faces aggregated in each voxel, using $m = 1$, 5, 10 or 20 faces.

As Table XIII shows, for meshes in ScanNet, using only geodesic features reduces accuracy greatly, as there is a large number of separate patches in the real scene dataset, making the receptive field of a large number of faces very small. Using only euclidean features also reduces accuracy: segmentation becomes harder. For example, when chairs and tables are close together, their euclidean features are very close and hard to distinguish, but doing so is easy using geodesic features. Meshes in Manifold40 have no disconnected patches, so in principle, each face can interact with any other face in the mesh, so good results can be achieved using only geodesic features.

Table XIV shows that increasing the number of faces aggregated in each voxel also affects our model, gradually improving the results, but at a cost of increasing memory and computation time. We aggregate 5 faces in our other experiments, which gives a good balance between the quality of results and the computational load.

I. Limitations and Future Works

Large kernel convolutions show stronger learning ability when processing 2D images [67], [68], [69]. These networks need to achieve a large receptive field and are usually implemented by dilation convolution and depthwise convolution, which can reduce the amount of computation and parameters. Besides, traditional and transpose convolutions with a stride

greater than 1 are also often used in the upsampling and downsampling processes. Although we have achieved good results by using small kernel convolution, we still want to further improve the scalability of hybrid convolution. Therefore, we will define more general convolution operations which support user-defined kernel size, stride and dilation. Furthermore, we will define the transpose convolution and depthwise convolution.

While our experiments demonstrate that the proposed method outperforms state-of-the-art approaches in many applications, it should be noted that due to the random sampling procedure and the feature conversion between faces and vertices, DGNet may not be well-suited for stability-sensitive tasks, such as physical simulations [70], [71].

VI. Conclusion

In this paper, we have proposed a general dual graph pyramids based approach for processing real-world, imperfect 3D mesh data by neural networks. Our method can allow networks to have a consistent receptive field, and can perform spatial oversampling in complex regions of meshes. We have also designed a hybrid convolution that aggregates geodesic and euclidean features and takes the interactions between disjoint pieces of a surface into account. These two modules enable neural networks to process imperfect meshes and achieve superior performance on various mesh analysis benchmarks.

Acknowledgments

We would like to thank all the reviewers for their valuable comments and thank Meng-Hao Guo for his discussion.

References

[1] Z. Wu et al., "3D shapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.
[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.
[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 4–9, 2017, pp. 5099–5108.
[4] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process. Syst.*, Montréal, Canada, Dec. 3–8, 2018, pp. 828–838.
[5] 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 Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.
[6] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "MeshCNN: A network with an edge," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, 2019.
[7] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao, "MeshNet: Mesh neural network for 3d shape representation," in *Proc. 33rd AAAI Conf. Artif. Intell., AAAI 31st Innov. Appl. Artif. Intell. Conf., 9th AAAI Symp. Educ. Adv. Artif. Intell.*, Honolulu, HI, USA, Jan. 27–Feb. 01 2019, pp. 8279–8286. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33018279
[8] J. Schult, F. Engelmann, T. Kontogianni, and B. Leibe, "DualConvMeshNet: Joint geodesic and euclidean convolutions on 3D meshes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8612–8622.
[9] Z. Hu et al., "VMNet: Voxel-mesh network for geodesic-aware 3D semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 15488–15498.
[10] N. Sharp, S. Attaiki, K. Crane, and M. Ovsjanikov, "DiffusionNet: Discretization agnostic learning on surfaces," *ACM Trans. Graph.*, vol. 41, no. 3, pp. 1–16, 2022.
[11] D. Smirnov and J. Solomon, "HodgeNet: Learning spectral geometry on triangle meshes," *ACM Trans. Graph.*, vol. 40, no. 4, pp. 1–11, 2021.
[12] F. Milano, A. Loquercio, A. Rosinol, D. Scaramuzza, and L. Carlone, "Primal-dual mesh convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 952–963.
[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[15] S.-M. Hu et al., "Subdivision-based mesh convolution networks," *ACM Trans. Graph.*, vol. 41, no. 3, pp. 1–16, 2022.
[16] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, 2008.
[17] J. Huang, H. Zhang, L. Yi, T. Funkhouser, M. Nießner, and L. J. Guibas, "TextureNet: Consistent local parametrizations for learning from high-resolution signals on meshes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4440–4449.
[18] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3d," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3887–3896.
[19] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 922–928.
[20] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3075–3084.
[21] Y.-P. Xiao, Y.-K. Lai, F.-L. Zhang, C. Li, and L. Gao, "A survey on deep geometry learning: From a representation perspective," *Comput. Vis. Media*, vol. 6, no. 2, pp. 113–133, 2020.
[22] R. Socher, B. Huval, B. P. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3D object classification," in *Proc. Adv. Neural Inf. Process. Syst. 26th Annu. Conf. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 3–6, 2012, pp. 665–673.
[23] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 945–953.
[24] B. Shi, S. Bai, Z. Zhou, and X. Bai, "DeepPano: Deep panoramic representation for 3-D shape recognition," *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2339–2343, Dec. 2015.
[25] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3D shape surfaces using geometry images," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2016, pp. 223–240.
[26] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8895–8904.
[27] J. Huang and S. You, "Point cloud labeling using 3D convolutional neural network," in *Proc. IEEE 23rd Int. Conf. Pattern Recognit.*, 2016, pp. 2670–2675.
[28] G. Riegler, A. Osman Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6620–6629.
[29] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 5–10, 2016, pp. 82–90.
[30] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3Dmatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1802–1811.
[31] B. Graham, M. Engelcke, and L. Van Der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9224–9232.
[32] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based convolutional neural networks for 3D shape analysis," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–11, 2017.
[33] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong, "Adaptive O-CNN: A patch-based deep representation of 3D shapes," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–11, 2018.

[34] P.-S. Wang, Y. Liu, and X. Tong, "Deep octree-based CNNs with output-guided skip connections for 3D shape and scene completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 266–267.
[35] P. Wang, Y. Yang, Q. Zou, Z. Wu, Y. Liu, and X. Tong, "Unsupervised 3D learning for shape analysis via multiresolution instance discrimination," in *Proc. 35th AAAI Conf. Artif. Intell. 33rd Conf. Innov. Appl. Artif. Intell. 11th Symp. Educ. Adv. Artif. Intell.*, Feb. 2–9, 2021, pp. 2773–2781. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16382
[36] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6411–6420.
[37] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "PCT: Point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187–199, 2021.
[38] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 16259–16268.
[39] N. Engel, V. Belagiannis, and K. Dietmayer, "Point transformer," *IEEE Access*, vol. 9, pp. 134826–134840, 2021.
[40] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.
[41] A. Lahav and A. Tal, "MeshWalker: Deep mesh understanding by random walks," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 1–13, 2020.
[42] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5115–5124.
[43] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia, "Learning mesh-based simulation with graph networks," in *proc. Int. Conf. Learn. Representations*, 2021, pp. 1–18.
[44] Y. Liang, S. Zhao, B. Yu, J. Zhang, and F. He, "MeshMAE: Masked autoencoders for 3D mesh data analysis," 2022, arXiv:*2207.10228*.
[45] Q. Dong, Z. Wang, J. Gao, S. Chen, Z. Shu, and S. Xin, "Laplacian2mesh: Laplacian-based mesh understanding," 2022, arXiv:*2202.00307*.
[46] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2015, pp. 37–45.
[47] D. Boscaini, J. Masci, E. Rodolà, and M. M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 5–10, 2016, pp. 3189–3197.
[48] H. Maron et al., "Convolutional neural networks on surfaces via seamless toric covers," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 71–1, 2017.
[49] N. Haim, N. Segol, H. Ben-Hamu, H. Maron, and Y. Lipman, "Surface networks via general covers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 632–641.
[50] Y. Yang, S. Liu, H. Pan, Y. Liu, and X. Tong, "PFCNN: Convolutional neural networks on 3D surfaces using parallel frames," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13578–13587.
[51] Y. Chen, J. Zhao, C. Shi, and D. Yuan, "Mesh convolution: A novel feature extraction method for 3D nonrigid object classification," *IEEE Trans. Multimedia*, vol. 23, pp. 3098–3111, 2021.
[52] H. Lei, N. Akhtar, and A. Mian, "Picasso: A cuda-based library for deep learning over 3D meshes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13854–13864.
[53] V. V. Singh, S. V. Sheshappanavar, and C. Kambhamettu, "MeshNet++: A network with a face," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 4883–4891.
[54] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.
[55] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5828–5839.
[56] Y. Wang, S. Asafi, O. Van Kaick, H. Zhang, D. Cohen-Or, and B. Chen, "Active co-analysis of a set of shapes," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 1–10, 2012.
[57] S.-M. Hu, D. Liang, G.-Y. Yang, G.-W. Yang, and W.-Y. Zhou, "Jittor: A novel deep learning framework with meta-operators and unified graph execution," *Sci. China Inf. Sci.*, vol. 63, no. 222103, pp. 1–21, 2020.
[58] A. X. Chang et al., "ShapeNet: An information-rich 3D model repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:*1512.03012*, 2015.
[59] 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, Hoboken, NJ, USA: Wiley, 2017, pp. 49–57.
[60] Z. Lian et al., "SHREC'11 track: Shape retrieval on non-rigid 3D watertight meshes," in *Proc. Eurographics Workshop 3D Object Retrieval*, 2011, pp. 79–88.
[61] M. Savva et al., "SHREC16 track: Largescale 3D shape retrieval from shapenet core55," in *Proc. Eurographics Workshop 3D object Retrieval*, vol. 10, 2016.
[62] H. Su et al., "SplatNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2530–2539.
[63] A. Dai and M. Nießner, "3DMV: Joint 3D-multi-view prediction for 3D semantic scene segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 452–468.
[64] A. X. Chang et al., "Matterport3D: Learning from RGB-D data in indoor environments," in *Proc. Int. Conf. 3D Vis.*, Qingdao, China, Oct. 10–12 , 2017, pp. 667–676. [Online]. Available: https://doi.org/10.1109/3DV.2017.00081
[65] J. Rossignac and P. Borrel, "Multi-resolution 3D approximations for rendering complex scenes," in *Modeling in computer graphics*. Berlin, Germany: Springer, 1993, pp. 455–465.
[66] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proc. 24th Annu. Conf. Comput. Graph. Interactive Techn.*, 1997, pp. 209–216.
[67] M.-H. Guo, C.-Z. Lu, Q. Hou, Z. Liu, M.-M. Cheng, and S.-M. Hu, "SegNeXt: Rethinking convolutional attention design for semantic segmentation," 2022, arXiv:*2209.08575*.
[68] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11976–11986.
[69] M.-H. Guo, C.-Z. Lu, Z.-N. Liu, M.-M. Cheng, and S.-M. Hu, "Visual attention network," 2022, arXiv:*2202.09741*.
[70] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 8459–8468.
[71] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning mesh-based simulation with graph networks," 2020. [Online]. Available: https://arxiv.org/abs/2010.03409

Xiang-Li Li is currently working toward the PhD degree with the Department of Computer Science and Technology, Tsinghua University. His research interests include computer graphics, 3D shape analysis, and computer vision.

Zheng-Ning Liu received the bachelor's and PhD degrees in computer science from Tsinghua University in 2017 and 2022, respectively. He is a research scientist with Fitten Tech Co., Ltd. His research interests include 3D reconstruction, geometric modeling and processing. He has several papers published in journals such as *ACM Transactions on Graphics*, *IEEE Transactions on Visualization and Computer Graphics*, etc.

Tuo Chen is currently working toward the undergraduation degree with Tsinghua University. His research interests include computer graphics and computer vision.

Tai-Jiang Mu received the BS and PhD degrees from the Department of Computer Science and Technology, Tsinghua University in 2011 and 2016, respectively. He is currently an assistant researcher with the Department of Computer Science and Technology, Tsinghua University. His research interests include computer graphics, visual media learning and image processing. He has published more than 20 papers in journals and refereed conferences, such as *ACM Transactions on Graphics*, *IEEE Transactions on Visualization and Computer Graphics*, *The Visual Computer*, *Graphical Models*, *Computational Visual Media*, CVPR, ICRA, etc.

Ralph R. Martin received the PhD degree from Cambridge University in 1983. He is an emeritus professor of Cardiff University. His extensive publications cover solid and surface modeling, intelligent sketch input, geometric reasoning, reverse engineering, and various aspects of computer graphics computer vision and image processing. He is currently the associate editor-in-chief of Computational Visual Media.

Shi-Min Hu (Senior Member, IEEE) received the PhD degree from Zhejiang University in 1996. He is currently a professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He has published more than 100 papers in journals and refereed conferences. He is the editor-in-chief of *Computational Visual Media*, and on editorial boards of several journals, including *Computer Aided Design and Computer & Graphics*. He is a senior member of ACM, and fellow of CCF and SMA.