

Subdivision-Based Mesh Convolution Networks

SHI-MIN HU, Tsinghua University, China
 ZHENG-NING LIU, Tsinghua University, China
 MENG-HAO GUO, Tsinghua University, China
 JUN-XIONG CAI, Tsinghua University, China
 JIAHUI HUANG, Tsinghua University, China
 TAI-JIANG MU, Tsinghua University, China
 RALPH R. MARTIN, Cardiff University, UK and Tsinghua University, China

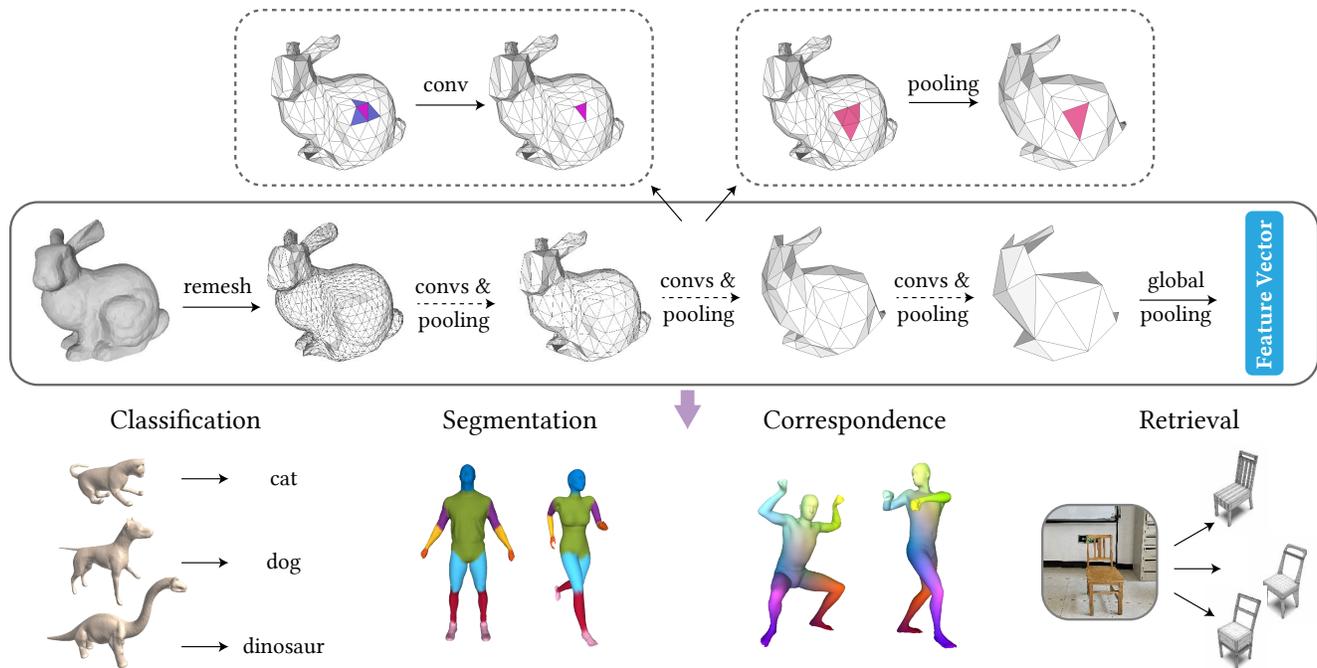


Fig. 1. SubdivNet, a subdivision-based mesh convolution network for deep geometric learning. Given a mesh as input, we construct a hierarchical subdivision structure with a pyramid of regular connectivities, analogous to a 2D image pyramid. This structure permits natural notions of convolution, pooling, and upsampling operation on 3D meshes, which together provide the building blocks of our mesh-based deep neural network. Our network is effective and efficient for mesh-based representation learning in a variety of applications.

Authors' addresses: Shi-Min Hu, Tsinghua University, Beijing, China, shimin@tsinghua.edu.cn; Zheng-Ning Liu, Tsinghua University, Beijing, China, lzhennging@gmail.com; Meng-Hao Guo, Tsinghua University, Beijing, China, gmh20@mails.tsinghua.edu.cn; Jun-Xiong Cai, Tsinghua University, 30 Shuangqing Rd, Haidian Qu, Beijing Shi, China, caijunxiong000@163.com; Jiahui Huang, Tsinghua University, 30 Shuangqing Rd, Haidian Qu, Beijing Shi, China, huang-jh18@mails.tsinghua.edu.cn; Tai-Jiang Mu, Tsinghua University, 30 Shuangqing Rd, Haidian Qu, Beijing Shi, China, taijiang@tsinghua.edu.cn; Ralph R. Martin, Cardiff University, Cardiff, UK, Tsinghua University, 30 Shuangqing Rd, Haidian Qu, Beijing Shi, China, MartinRR@cs.cf.ac.uk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Association for Computing Machinery.
 0730-0301/2021/12-ART \$15.00
<https://doi.org/10.1145/nmmnnnnn.nmmnnnnn>

Convolutional neural networks (CNNs) have made great breakthroughs in 2D computer vision. However, their irregular structure makes it hard to harness the potential of CNNs directly on meshes. A subdivision surface provides a hierarchical multi-resolution structure, in which each face in a closed 2-manifold triangle mesh is exactly adjacent to three faces. Motivated by these two observations, this paper presents *SubdivNet*, an innovative and versatile CNN framework for 3D triangle meshes with Loop subdivision sequence connectivity. Making an analogy between mesh faces and pixels in a 2D image allows us to present a mesh convolution operator to aggregate local features from nearby faces. By exploiting face neighborhoods, this convolution can support standard 2D convolutional network concepts, e.g. variable kernel size, stride, and dilation. Based on the multi-resolution hierarchy, we make use of pooling layers which uniformly merge four faces into one and an upsampling method which splits one face into four. Thereby, many popular 2D CNN architectures can be easily adapted to process 3D meshes. Meshes with arbitrary connectivity can be remeshed to have Loop

subdivision sequence connectivity via self-parameterization, making SubdivNet a general approach. Extensive evaluation and various applications demonstrate SubdivNet’s effectiveness and efficiency.

CCS Concepts: • **Computing methodologies** → **Neural networks; Shape analysis.**

Additional Key Words and Phrases: Geometric Deep Learning, Convolutional Neural Network, Subdivision Surfaces, Mesh Processing

ACM Reference Format:

Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Jun-Xiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph R. Martin. 2021. Subdivision-Based Mesh Convolution Networks. *ACM Trans. Graph.* 1, 1 (December 2021), 17 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

Deep convolutional networks (CNNs) have achieved great success in 2D computer vision, leading to their generalization to a variety of disciplines, including 3D geometry processing. PointNet [Qi et al. 2017a] is a pioneering approach for learning a feature representation of a point cloud; it has led to the development of more powerful networks [Li et al. 2018; Qi et al. 2017b]. 3D geometric learning has also been extended to other forms of 3D data, such as voxels [Klokov and Lempitsky 2017; Wang et al. 2017] and meshes [Hanocka et al. 2019; Lahav and Tal 2020].

In this paper, we consider 3D geometric learning using a mesh representation. Polygonal meshes are one of the most common 3D data representations, with applications in modeling, rendering, animation, 3D printing, etc. Unlike point clouds, meshes contain topological information. Polygon meshes can represent geometric context more effectively than voxels, since they only depict the boundaries of objects, omitting superfluous elements for the interior.

Underpinning the success of 2D CNNs in the image domain is the inherently regular grid structure allowing hierarchical image pyramids, which enable CNNs to explore features of varying sizes by downsampling and upsampling. However, an arbitrary mesh is irregular and lacks the gridded arrangement of pixels in images, making it difficult to define a standard convolution with variable kernel size, stride, and dilation. This prevents 3D geometric learning methods from taking advantage of mature network architectures used in the image domain. Furthermore, the unstructured connectivity between vertices and faces precludes finding a simple fine-to-coarse structure for meshes. One may consider levels of detail (LOD) via mesh simplification, but the mapping of geometry primitives between each level is not well-defined, and one cannot derive an intuitive pooling operator from a LOD hierarchy.

To apply the power of CNNs to meshes, efforts have been made to define special convolution operations on surfaces. Such methods typically attempt to encode or re-sample the neighborhood of a vertex into a regular local domain [Boscaini et al. 2016; Huang et al. 2019; Maron et al. 2017; Masci et al. 2015; Monti et al. 2017; Poulernard and Ovsjanikov 2018; Tatarchenko et al. 2018], where the convolutional operation can be derived. Recently, mesh downsampling schemes have been proposed to dynamically merge regions using edge collapse [Hanocka et al. 2019; Milano et al. 2020], but they do not guarantee a uniformly enlarged receptive field everywhere as in 2D pooling. Although the attention mechanism [Velickovic et al.

2018] can be applied to capture global context by treating a mesh as a graph, doing so comes at a heavy computational cost.

Instead, motivated by image pyramids in 2D CNNs that allow local features to be aggregated into larger-scale features at different levels, we note that subdivision surfaces also construct well-defined hierarchical mesh pyramids. A subdivision surface is a smooth surface produced by refining a coarse mesh. In particular, in Loop subdivision [Loop 1987], each triangle mesh face is split into 4 triangles and then vertex positions are updated to smooth the new mesh (see Fig. 2(a)). As a result, the Loop subdivision scheme gives a 1-to-4 face mapping from the coarse mesh to the finer one. Correspondingly, if a mesh has the same connectivity as a Loop subdivision surface, it has a natural correspondence to a one-level-coarser mesh, and indeed, a carefully constructed Loop subdivision surface may preserve the Loop property over several levels leading to a fine to coarse hierarchy. Furthermore, each face in any closed 2-manifold mesh is exactly surrounded by 3 other faces. The fixed number of face neighbors suggests a regular structure analogous to that of pixels in images, making it suitable for deriving a standard convolution operation on a mesh.

In this paper, we propose *SubdivNet*, which can learn feature representations for meshes with Loop subdivision sequence connectivity. Using the neighbors of faces, we define a novel convolution on mesh faces that supports variable kernel size, stride, and dilation. Thus, ours can operate on a large receptive field. Because of the flexibility of our new convolution on triangle meshes, successful neural networks in the image domain, such as VGG [Simonyan and Zisserman 2015], ResNet [He et al. 2016], and DeepLabv3+ [Chen et al. 2018], can be naturally adapted to meshes.

SubdivNet requires a mesh with subdivision sequence connectivity as input, which may appear excessively limiting. However, any triangle mesh representing a closed 2-manifold with arbitrary genus can be remeshed to have this property via self-parameterization [Lee et al. 1998; Liu et al. 2020]. Thereby, SubdivNet can be used as a general feature extractor for any closed 2-manifold triangle mesh.

SubdivNet achieves state-of-the-art performance on 3D shape analysis, e.g. mesh classification, segmentation, and shape correspondence. Ablation studies verify the effectiveness of the proposed convolution, subdivision-based pooling, and advanced network architectures.

In summary, our work makes the following contributions:

- a general mesh convolutional operation that permits variable kernel size, stride, and dilation analogous to standard 2D convolutions, making it possible to adapt well-known 2D CNNs to mesh tasks,
- SubdivNet, a general mesh neural network architecture based on mesh convolution and subdivision sequence connectivity, with uniform pooling and upsampling, for geometric deep learning, supporting dense prediction tasks,
- demonstrations that SubdivNet provides excellent results for various applications, such as shape correspondence and shape retrieval.

2 RELATED WORK

2.1 3D Geometric Learning

One way of applying deep learning to geometric data is to transform 3D shapes into images, e.g., an unordered set of projections [Su et al. 2015], panoramas [Shi et al. 2015], or geometry images [Sinha et al. 2016], and then run 2D CNNs on them. This family of *indirect* methods is pose-sensitive because an additional view-dependent projection step is involved. Another line of *direct* solutions is to represent the shapes in their intrinsic 3D space, such as volumetric data, whereupon 3D CNNs can be applied [Maturana and Scherer 2015; Wu et al. 2015] or adapted for higher resolution [Klokov and Lempitsky 2017; Liu et al. 2021; Wang et al. 2017]. Recently, point-based learning techniques have emerged [Li et al. 2018; Qi et al. 2017a,b; Wang et al. 2019] due to the ease of acquisition of point cloud data by 3D sensors. Nevertheless, the high computational demand for volumetric data and the absence of topological information for point clouds make current pipelines inefficient. However, methods that learn on *mesh surfaces* overcome the above problems and have shown promise. Readers are referred to recent surveys [Bronstein et al. 2017; Xiao et al. 2020] for a comprehensive review on 3D geometric learning.

2.2 Deep Learning on Meshes

Meshes are composed of three distinct types of geometric primitives: vertices, edges, and faces. We categorize mesh deep learning methods based on which of these are treated as the primary data.

2.2.1 Vertex-based. Certain works perform deep learning on 3D shapes by locally encoding the neighborhood of sampled points into a regular domain, whereupon convolution operations (or kernel functions) can mimic those for images. Masci et al. [2015], Boscaini et al. [2016], Tatarchenko et al. [2018], and Poulenard et al. [2018] parameterize geodesic patches into 2D domains, e.g. the tangent plane, for use with 2D CNNs or point networks. TextureNet [Huang et al. 2019] and PFCNN [Yang et al. 2020] extend geodesic convolution by better handling inconsistent orientations of tangent spaces. Global parameterization is employed by Maron et al. [2017] and Haim et al. [2019] to perform surface convolution. Such methods are usually insensitive to the meshing of shapes due to parameterization. Another series of works applies convolution directly on the mesh structure. Some approaches [Chen et al. 2020; Dai and Nießner 2019; Kostrikov et al. 2018; Monti et al. 2017] employ graph neural networks (GNN) to use vertex connectivity. Pixel2Mesh [Wang et al. 2018] generates meshes from coarse to fine via subdivision, and updates geometry by GNN. Lim et al. [2018] and Gong et al. [2019] propose a spiral convolution pattern within the k -ring neighborhood of a vertex. DiffusionNet [Sharp et al. 2021] and HodgeNet [Smirnov and Solomon 2021] extend the Laplacian operator to learn the surface representation. While these methods can learn the local representation, they are usually less capable of learning multi-scale and contextual information in a mesh.

Closer to our approach are methods with hierarchical design. Dilated kernel parametrization [Yi et al. 2017], and mesh downsampling and upsampling [Ranjan et al. 2018] can be adopted in

the spectral domain to define mesh convolution to aggregate multi-scale information. Schult et al. [2020] combine two kinds of convolutions separately defined on neighbors according to geodesic and Euclidean distance, also exploiting mesh simplification to provide a multi-resolution architecture. Based on subdivision sequence connectivity, our approach offers a more general and standard convolution directly defined on the mesh; as such, it supports variable kernel size, stride, and dilation.

2.2.2 Edge-based. Each edge in a 2-manifold triangle mesh is adjacent to two faces and four ‘next’ edges. This property is exploited by MeshCNN [Hanocka et al. 2019] to define an ordering invariant convolution. PD-MeshNet [Milano et al. 2020] constructs a primal graph and a dual graph from the input mesh, then performs convolutions on these graphs using a graph attention network [Monti et al. 2018; Velickovic et al. 2018]. MeshWalker [Lahav and Tal 2020] extracts shape features by walking along edges rather than exploiting neighborhood structures.

MeshCNN and PD-MeshNet dynamically contract edges to simplify meshes within the network, whereas MeshWalker build the hierarchy using variable-step walks. Unlike our approach, they do not provide a downsampling scheme to uniformly expand the receptive field, a successful strategy in 2D CNNs.

2.2.3 Face-based. Face based methods focus on how to efficiently and effectively gather information from neighboring faces. Xu et al. [2017] propose a rotationally-invariant face based method considering k -ring neighbors for defining convolution on meshes; it is guided by face curvature. MeshSNet, proposed by Lian et al. [2019], adopts graph-constrained mesh-cell nodes to integrate local-to-global geometric features. MeshNet [Feng et al. 2019] learns the spatial and structural features of a face by aggregating its 1-ring neighbors with the help of two mesh convolutional layers. DNF-Net [Li et al. 2020] uses multi-scale embedding and a residual learning strategy to denoise mesh normals on cropped local patches. Hertz et al. [2020] generate geometric textures, using a 3-face convolution and a subdivision-based upsampling similar to Pixel2Mesh [Wang et al. 2018].

SubdivNet uses a regular and uniform downsampling scheme to establish a fine-to-coarse mesh hierarchy. Our convolution also exploits distant neighbors of faces and thus can have a larger receptive field. Compared to Xu et al. [2017], our convolutions efficiently support stride and large dilation, allowing us to better capture long-range features.

2.3 Subdivision Surfaces and Multiresolution Modeling

A subdivision surface is a smooth surface produced by refining a coarse mesh. The best-known mesh subdivision algorithms are Catmull-Clark subdivision [Catmull and Clark 1978] for quad meshes and Loop subdivision [Loop 1987] for triangle meshes. They insert new vertices and edges, split faces, and linearly update vertex positions. Other subdivision schemes, e.g. [Doo and Sabin 1978; Dyn et al. 1990; Kobbelt 2000] and non-linear approaches, e.g. [Liu et al. 2006; Schaefer et al. 2008] have also been proposed.

Multi-resolution modeling, also known as level of detail, aims to construct a sequence of meshes from fine to coarse, and is widely

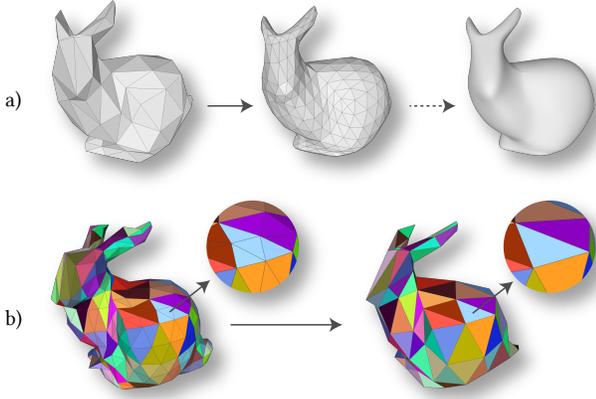


Fig. 2. (a) Loop subdivision. A coarse mesh is iteratively refined by splitting each face into four and smoothing. (b) The 4-to-1 face mapping from a fine mesh to the next coarser level using Loop subdivision sequence connectivity.

applied in mesh compression, editing, and fast rendering. There have been many works on this topic, and we will only consider methods that maintain subdivision sequence connectivity from fine to coarse: we need a whole hierarchy of meshes *all* having Loop subdivision connectivity. MAPS [Lee et al. 1998] is a pioneering work that computes a parameterization of a mesh over a simplified version of the mesh. Then, a new mesh with subdivision sequence connectivity is constructed on the surface of the simplified mesh. Finally, the vertices of the new mesh are projected back to the input faces via the parameterization. This idea is further improved in terms of distortion and smoothness by [Guskov et al. 2002, 2000; Khodakovskiy et al. 2003; Kobbelt et al. 1999]. Liu et al [2020] have also extended the MAPS algorithm to generate multi-resolution meshes for network training.

By use of such a multi-resolution method, any mesh can be remeshed to have subdivision sequence connectivity, making SubdivNet a general method for 3D mesh analysis.

3 SUBDIVNET

3.1 Notation

Before giving details of SubdivNet, it is necessary to define the mathematical notation used throughout this paper.

A triangle mesh $\mathcal{M} = (\mathbf{V}, \mathbf{F})$ is defined by a set of vertices $\mathbf{V} = \{v_i | v_i \in \mathcal{R}^3\}$ and a set of triangular faces $\mathbf{F} = \{f_i | f_i \in \{1, \dots, |\mathbf{V}|\}^3\}$, indicating the triangle's vertices, and hence implicitly, the connectivity. Each face f_i holds an input feature vector e_i , which is to be processed by SubdivNet.

Two faces f_i and f_j are said to be adjacent if they share an edge. The distance $D(f_i, f_j)$ between f_i and f_j is defined as the minimum number of faces traversed by any path from one to the other across edges. The k -ring neighborhood of f_i is then:

$$\mathcal{N}_k(f_i) = \{f_j | D(f_i, f_j) = k\}.$$

We say that a triangle mesh \mathcal{M} has *Loop subdivision connectivity* if it has the same connectivity as a mesh formed by one round of Loop subdivision acting on a coarser mesh.

We say that a triangle mesh \mathcal{M} has *Loop subdivision sequence connectivity* if there exists a sequence of meshes $(\mathcal{M}_0, \dots, \mathcal{M}_L)$, $L \geq 1$, where $\mathcal{M}_L = \mathcal{M}$, satisfying two requirements: (i) all except possibly \mathcal{M}_0 have Loop subdivision connectivity; (ii) all vertices in \mathcal{V}_i are also present in \mathcal{V}_{i+1} , $0 < i < L$. The bunny in Fig. 1 illustrates such a sequence.

We refer to L as the *subdivision depth* of \mathcal{M} , \mathcal{M}_0 as the *base mesh* of \mathcal{M} , and the number of faces of \mathcal{M}_0 as the *base size*. Clearly, the number of faces of \mathcal{M} is $|\mathbf{F}| = 4^L |\mathbf{F}_0|$.

If \mathcal{M} has Loop subdivision sequence connectivity, we can establish a 4-to-1 face mapping from each mesh \mathcal{M}_i , $i > 0$, to mesh \mathcal{M}_{i-1} , which can be regarded as the topological inverse of Loop subdivision (i.e. ignoring vertex geometry updates) (see Fig. 2(b)).

Most common meshes, whether designed by artists or scanned by sensors, lack Loop subdivision sequence connectivity. Hence, we first remesh the input mesh via a self-parameterization to confer this property on it, for a specified base size and subdivision depth, using methods explained in Sec. 4. The remainder of this section will assume that the input mesh has been appropriately remeshed.

3.2 Overview

Given a watertight 2-manifold triangle mesh with Loop subdivision sequence connectivity, we aim to learn a global representation for the 3D shape, or feature vectors on each face for local geometry.

Like a 2D image pyramid, $(\mathcal{M}_L, \dots, \mathcal{M}_0)$ provides a hierarchical structure, or *mesh pyramid*. Through the 4-to-1 face mapping provided by subdivision sequence connectivity, we can also establish an injection of faces from \mathcal{M}_L to \mathcal{M}_0 step by step, allowing feature aggregation from local to global.

Based on this, SubdivNet takes a mini-batch of closed 2-manifold triangle meshes with subdivision sequence connectivity as input. It computes features with convolutions defined on triangle faces, and aggregates long-range feature descriptions of meshes by uniform face downsampling. Because of the regular number of face neighbours, the mesh convolution also supports variable kernel size, dilation, and stride, acting like a standard convolution in the image domain. To support dense prediction tasks, e.g. segmentation, an upsampling operation is provided as the inverse of pooling.

Due to the flexibility and generality of convolution, pooling and upsampling, we can directly adapt well-known networks from the 2D image domain to mesh learning tasks.

3.3 Convolution

This section discusses which faces should be considered in convolution, i.e. the convolution kernel pattern Ω . Then we will discuss how to perform the convolution with such kernels patterns.

3.3.1 Basic Convolution Pattern. A key to defining convolution for a given signal is to specify its neighborhood, or the kernel pattern. Since there are no boundary edges in a 2-manifold triangle mesh, each face on the mesh has exactly 3 adjacent faces. This 3-regular property is analogous to the lattice connectivity of pixels in 2D images, motivating us to define a basic convolution over faces. Formally, for each face f_i , the basic convolution kernel pattern is formed by its 1-ring neighbors $\Omega(f_i) = \mathcal{N}_1(f_i)$, illustrated in Fig 3(a).

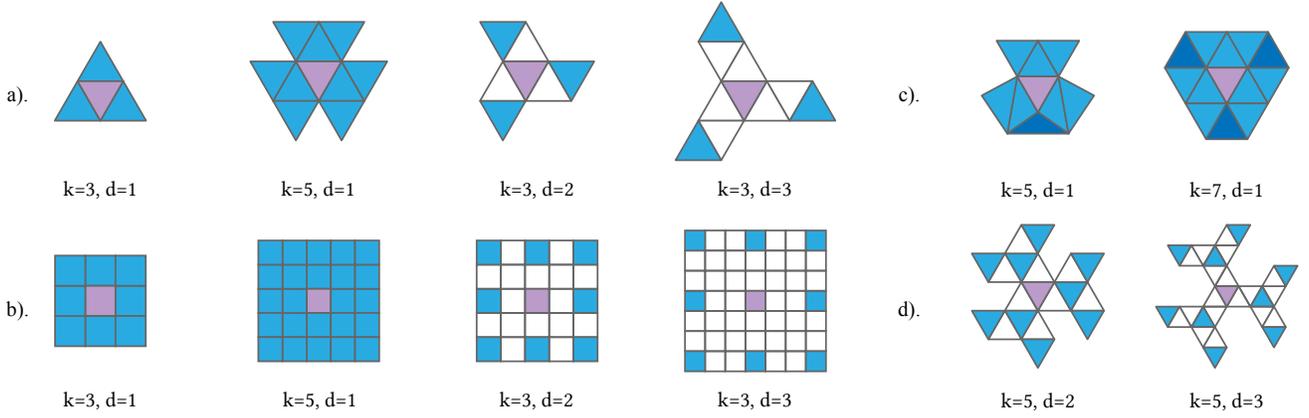


Fig. 3. Mesh convolution kernel pattern. (a) Mesh convolution kernels with different kernel size k and dilation d . (b) Corresponding 2D convolution kernels. (c) Duplication happens for kernel sizes larger than 3: deep blue faces are accessed twice. (d) More complex convolution kernels with larger kernel size and dilation.

3.3.2 Kernel Size. To enable the convolution to have a larger receptive field, convolution in 2D images is designed to support a variable kernel size. This is also critical in shape analysis to learn more discriminative representations for each vertex and face, facilitating tasks such as shape correspondence and segmentation. We consider additional nearby faces to define the pattern of a convolution with variable kernel size k , so

$$\Omega(f_i, k) = \bigcup_{i=1}^{\hat{k}} \mathcal{N}_{\hat{k}}(f_i), \quad \hat{k} = \frac{k-1}{2}, k = 1, 3, 5, \dots \quad (1)$$

In total, there are $3 \times (2^{\hat{k}} - 1)$ faces in the kernel pattern. For instance, Fig. 3(a) depicts a case with a kernel size of 5.

However, when k is greater than 3, adjacent faces may be counted more than once, resulting in fewer triangles in Ω than expected, as seen Fig. 3(c). Avoiding such issues would lead to a complex convolution design, so we simply preserve all duplications of faces and keep $|\Omega|$ fixed. Aside from simplicity, another reason is that in modern networks, a larger 2D convolution kernel is usually substituted by a stack of small kernels. When $k = 3$, no duplication occurs. When $k = 5$, duplication can only exist around vertices whose degree is 4 or less (the degree of most vertices is 6 due to subdivision sequence connectivity). When the kernel size is larger than 7, faces may be accessed more than twice.

3.3.3 Dilation. Dilated convolution, also known as atrous convolution, is a widely used variation, where holes are inserted in the kernel pattern (see Fig 3(b)). Dilation is an efficient strategy to expand the receptive field without consuming more computing or memory resources. To extend this concept to triangle meshes, we define that in a kernel pattern with dilation d , the distance between a face and its nearest face (including the center f_i) is d . In particular, the dilation of the basic convolution pattern is 1. In a 2D image grid, the dilated kernel can be easily obtained by skipping rows and columns. However, such a strategy cannot be readily applied to triangle meshes (see Fig. 4(b)). Instead, we propose a *zig-zag* strategy

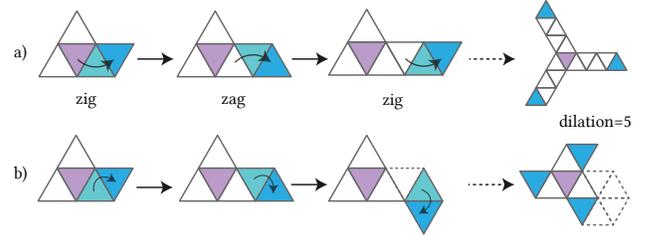


Fig. 4. (a) Zig-zag strategy provides a uniformly dilated convolution from the basic convolution pattern. (b) Alternative scheme folds back, causing the dilation pattern to select triangles too close to the central triangle.

to define a kernel pattern with dilation d , shown in Fig 4(a). Taking a dilated convolution whose kernel size is 3 as an example, we move from faces in $\Omega(f_i, d)$ to neighbors in turn d times, alternately clockwise or counter-clockwise with respect to the last position. Without loss of generality, if we assume that the input ordering of the three vertices in a face is counter-clockwise, then ‘zig’ and ‘zag’ are counter-clockwise and clockwise, respectively. Using the opposite definition is also feasible, which leads to a different, but symmetric pattern.

For kernel sizes k greater than 3, theoretically, we may first find the k -ring neighbors and then perform dilation before the finding $(k+1)$ -ring neighbors (see Fig. 3(d)).

This is not the only way to define dilation, but this formulation is based on face distance, consistently with our approach to kernel size. Another motivation is to ensure that $|\Omega(f_i, d)| = |\Omega(f_i, d=1)|$ as required in 2D image grids. Furthermore, the zig-zag style results in a uniform spatial distribution of elements in Ω , reducing the occurrence of duplicated faces.

One may notice that the proposed dilation is asymmetric. As a result, only elements from three directions are considered when the kernel size is 3 while the information from the other three directions is lost, possibly leading to bias. However, with two or more dilated

convolutions, features for all directions can be aggregated, avoiding potential bias.

3.3.4 Stride. In 2D CNNs, stride determines how densely the convolution is applied to the image. Because a 2D convolution with a stride greater than 1 reduces the resolution of the 2D feature map, it is frequently used for downsampling, acting as a pooling layer with parameters.

Thus, we also define a strided convolution based on the mesh pyramid. When the mesh convolution has a stride, it is only applied to the central face of a group of faces that merge into a single face in the coarser mesh. Since 1-to-4 Loop subdivision is used throughout SubdivNet, the stride size is 2. To support an arbitrary stride size s , one can choose a 1-to- s^2 split subdivision scheme rather than the Loop subdivision scheme when remeshing the input. See Sec. 4 for further discussion of remeshing.

3.3.5 Order-invariant Convolution Computation. The three neighbors of a face are unordered, yet a robust convolution should be ordering invariant. While $\Omega(f_i)$ is an unordered set, we rearrange the set counter-clockwise around f_i , resulting in a sequence $\hat{\Omega}(f_i)$. Therefore, $\hat{\Omega}(f_i)$ is a closed ring (see Appendix A for details). Even so, where ring ordering starts is still ambiguous, but we can remove the ambiguity by computing order-invariant intermediate features. The convolution on a face f_i is defined as,

$$\text{Conv}(f_i) = w_0 e_i + w_1 \sum_{j=1}^n e_j + w_2 \sum_{j=1}^n |e_{j+1} - e_j| + w_3 \sum_{j=1}^n |e_i - e_j|, \quad (2)$$

where $n = |\hat{\Omega}(f_i)|$. e_j ($e_{n+1} = e_1$) is the feature vector on the j th face in $\hat{\Omega}(f_i)$, and (w_0, w_1, w_2, w_3) are learnable parameters. As summation is ordering invariant, the convolution is also insensitive to face ordering.

3.4 Pooling

With the pyramid of input meshes, pooling on triangle meshes is as simple as on a regular 2D image grid, as shown in Fig. 5: four subdivided faces in the finer mesh are pooled to the parent face in the coarser mesh.

MeshCNN [Hanocka et al. 2019] defines pooling via dynamic edge collapse. To maintain a half-edge data structure, edge collapse is executed in sequence, while our pooling approach can be implemented in parallel. On the other hand, edge pooling cannot guarantee that all edges are downsampled once, but our uniform scheme ensures that all faces are involved in pooling. As a result, our approach produces a more spatially-uniform coarse mesh and achieves the same level of feature aggregation everywhere, though the face areas after pooling are not the same.

Like the convolution stride, the pooling operation supports a stride s greater than 2 via a 1-to- s^2 subdivision scheme when pre-processing the input meshes.

3.5 Upsampling

As the reverse of pooling, upsampling is also defined with the help of the mesh pyramid. *Nearest upsampling* simply splits a face into 4 faces; features on split faces are copied from the original face, as shown in Fig. 5.

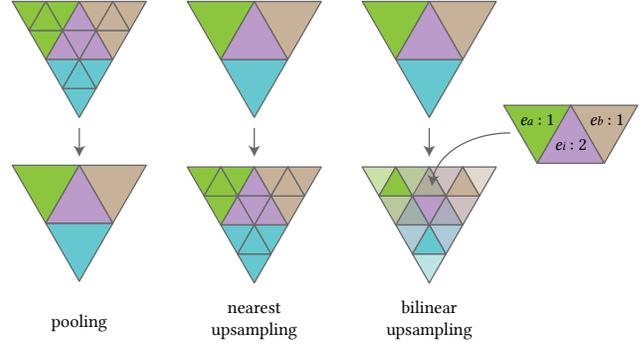


Fig. 5. Pooling and upsampling

In 2D dense prediction tasks, bilinear upsampling is also widely utilized. Therefore, we also define a *bilinear upsampling* based on face distance, to provide smoother interpolation than nearest upsampling. See Fig. 5. The feature of the central subdivided face f_i is equal to the original face's feature, while the feature of the other faces f_j is computed by,

$$e_j = \frac{1}{2} e_i + \frac{1}{4} e_a + \frac{1}{4} e_b, \quad (3)$$

where f_a and f_b are the other two faces adjacent to f_j .

3.6 Propagating Features to Raw Meshes

Because the proposed convolution (without stride) is based on neighborhoods, it can also be applied on raw input meshes lacking subdivision sequence connectivity. We thus propose a *feature propagation layer* to transfer information from the remeshed shape to the original mesh. This layer is similar to k -NN propagation in point cloud methods [Qi et al. 2017b]. For each face f^R in the raw mesh, we find the nearest face f_0^r in the remeshed shape. Let f_i^r ($i = 1, 2, 3$) be the three adjacent faces of f_0^r . Then the feature vector on f^R is obtained by interpolation, weighted by distance:

$$e^R = \sum_{i=0}^3 \lambda_i e_i^r / \sum_{i=0}^3 \lambda_i, \quad \lambda_i = 1/\mathbb{D}(f^R, f_i^r)^2, \quad (4)$$

where $\mathbb{D}(\cdot)$ is the Euclidean distance between face centers.

With feature propagation and further convolution on the raw input, SubdivNet can be end-to-end, making it convenient to incorporate other learning approaches on meshes. The extra layers can also aid in obtaining per-face predictions on raw meshes. However, it is simpler and more practical to project results from the remeshed shape onto the raw mesh by finding the nearest face. Therefore, feature propagation is not used in most of our experiments.

3.7 Network Architecture

Benefiting from the regularity that the proposed convolution offers, we can easily apply popular 2D convolutional networks to 3D meshes. In our experiments, we implement a VGG-style [Simonyan and Zisserman 2015] network for classification, and U-Net [Ronneberger et al. 2015] and DeepLabv3+ [Chen et al. 2018] with a ResNet50 [He et al. 2016] backbone for dense prediction.

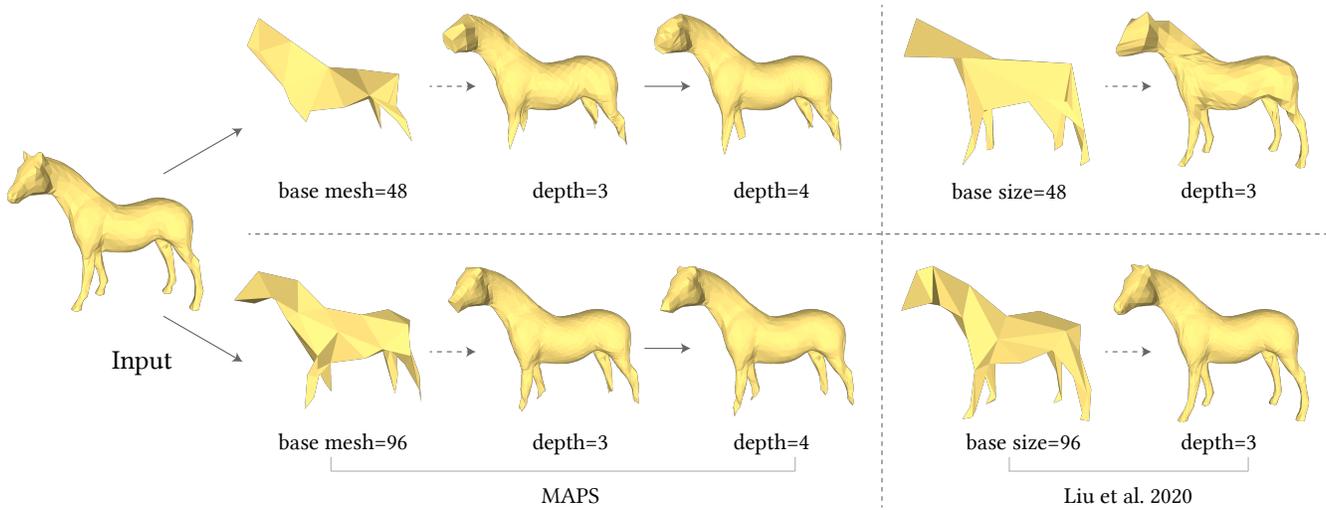


Fig. 6. Remeshing results of MAPS and Liu et al. [2020]. MAPS first decimates the input mesh, and constructs a bijection between the original mesh and the base mesh. Then the base mesh is subdivided and new vertices are projected back onto the input. Limited sampling in the base mesh leads to obvious distortion, e.g. in the limbs. Liu et al.’s approach produces results of higher quality.

DeepLabv3+ provides state-of-the-art performance for 2D image segmentation; its key idea is to extend the receptive field by using dilated convolutions. This mechanism is also helpful for 3D meshes; we provide network details in Appendix A.

Previous work [Hanocka et al. 2019; Milano et al. 2020] has adopted residual design and skip connections in networks, while not including all components of the original ResNet or DeepLabv3+, such as the strided convolution or the dilated convolution.

3.8 Input features

The input feature for each triangular face is a 13-dimensional vector, composed of a 7-dimensional shape descriptor and a 6-dimensional pose descriptor. The components of the shape descriptor are the face area, the three interior angles of the triangle, and the inner products of the face normal with the three vertex normals (characterizing curvature). The pose descriptor gives the position of the face center and the face normal, helping the network to identify faces with similar shapes through position and orientation. Further user-defined features such as color could also be added for specific learning tasks.

4 REMESHING FOR SUBDIVISION CONNECTIVITY

SubdivNet requires meshes with Loop subdivision sequence connectivity as input; however, most available meshes lack this property. We thus must remesh the input to have this property beforehand.

One solution is to use a self-parameterization method, e.g. the MAPS algorithm [Lee et al. 1998] or an improved version [Liu et al. 2020]. The key idea is to establish a mapping between the input mesh and the simplified mesh (the base mesh). Thus, by subdividing the base mesh and back-projection onto the input mesh, we can remesh the input with subdivision sequence connectivity. The output of MAPS occasionally exhibits visible distortion. Liu et al. improve the

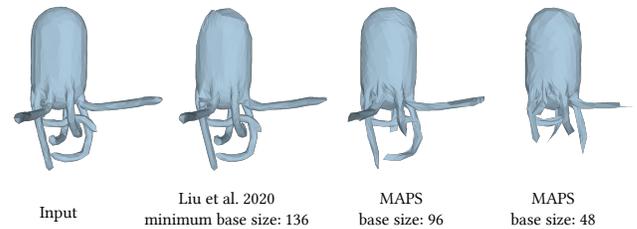


Fig. 7. Octopus mesh from the SHREC11 dataset [Wang et al. 2012]. Liu et al.’s [2020] method cannot produce a lower base mesh size than 136 triangles. However, MAPS can reduce the base size while keeping important features. For example, although shortened, the number of tentacles is unchanged. Here, subdivision depth = 3.

quality of output but cannot reach a low base size when the input mesh is complicated (see Fig. 7). See Appendix B for further details.

In practice, we adopt a simple strategy to switch between remeshing approaches according to task. For tasks where the global shape is more important and local distortion can be tolerated, such as classification, we choose the MAPS algorithm to achieve a higher degree of global feature aggregation, whereas for tasks where local details are more crucial, e.g. fine-grained segmentation, we use Liu et al.’s approach and a larger base size.

Both remeshing algorithms require the input to be manifold and closed, else local parameterizations may fail. More general meshes must be converted to watertight manifolds beforehand, via additional preprocessing (see Appendix C).

5 EXPERIMENTS

The generality and flexibility of our mesh convolution permits SubdivNet to be applied to a wide range of 3D shape analysis tasks.

We have quantitatively evaluated SubdivNet for mesh classification, mesh segmentation, and shape correspondence, comparing it to state-of-the-art alternatives. We have conducted other qualitative experiments to demonstrate its applicability in other areas, such as real-world mesh retrieval. The key components of SubdivNet are also evaluated in an ablation study.

5.1 Data preprocessing and augmentation

As the meshes in the datasets do not have Loop subdivision sequence connectivity, we first remeshed all data in both the training and the test sets. As some datasets are small, we generated multiple remeshed meshes for each input by randomly permuting the order of vertex removal or edge collapse. Augmentation makes the network insensitive to remeshing.

To reduce network sensitivity to size, we scaled each input to fit inside a unit cube and then applied random anisotropic scaling with a normal distribution $\mu = 1$ and $\sigma = 0.1$, following [Hanocka et al. 2019]: for example, certain human body shapes are taller or thinner than others. We also find that some orientations of shapes in the test dataset do not appear in the training dataset, e.g. in the human body dataset [Maron et al. 2017]. Therefore, for such datasets, we also randomly changed the orientation of the input data by rotating it around the three axes with Euler angles of $0, \pi/2, \pi$, or $3\pi/2$.

5.2 Classification

We first demonstrate the capabilities of SubdivNet for mesh classification using three datasets. As in the data augmentation adopted during training, we additionally generated 10 to 20 remeshed shapes for each mesh in the test, and a majority voting strategy was applied to reduce the variance introduced by remeshing.

5.2.1 SHREC11. The SHREC11 dataset [Lian et al. 2011] contains 30 classes with 20 samples per class. Following the setting in [Hanocka et al. 2019], SubdivNet is evaluated using two protocols with 16 or 10 training examples in each class. We report the average accuracy on 3 random splits into training and test sets in Table 1. SubdivNet correctly classifies all test meshes. Even without majority voting, SubdivNet is still comparable to or outperforms other state-of-the-art methods. With voting, accuracy already reaches 100% when accuracy without voting is around 95% in training. This suggests that the proposed method is sufficiently accurate for the SHREC11 mesh classification task.

5.2.2 Cube Engraving. The Cube Engraving dataset [Hanocka et al. 2019] was synthesized by engraving 2D shapes on one random face of a cube. There are 22 categories and 4,381 shapes in the released dataset. SubdivNet is the first method to make no mistakes, as shown in Table 2.

5.2.3 Manifold40. ModelNet40 [Wu et al. 2015], containing 12,311 shapes in 40 categories, is a widely used benchmark for 3D geometric learning. However, most 3D shapes in ModelNet40 are not watertight or 2-manifold, leading to remeshing failures. Therefore, we reconstructed the shapes in ModelNet40 and built a corresponding *Manifold40* dataset, in which all shapes are closed manifolds. See Appendix C for details of Manifold40 and the specific experimental settings.

Method	Split 16	Split 10
GWCNN [Ezuz et al. 2017]	96.6%	90.3%
MeshCNN [Hanocka et al. 2019]	98.6%	91.0%
PD-MeshNet [Milano et al. 2020]	99.7%	99.1%
MeshWalker [Lahav and Tal 2020]	98.6%	97.1%
HodgeNet [Smirnov and Solomon 2021]	99.2%	94.7%
DiffusionNet [Sharp et al. 2021]	-	99.7%
SubdivNet (w/o majority voting)	99.9%	99.5%
SubdivNet	100%	100%

Table 1. Classification accuracy on the SHREC11 dataset [Lian et al. 2011]. We believe that SubdivNet is the first method to achieve perfect performance.

Method	Accuracy
PointNet++ [Qi et al. 2017b]	64.3%
MeshCNN [Hanocka et al. 2019]	92.2%
PD-MeshNet [Milano et al. 2020]	94.4%
MeshWalker [Lahav and Tal 2020]	98.6%
SubdivNet (w/o majority voting)	98.9%
SubdivNet	100.0%

Table 2. Classification accuracy on the Cube Engraving dataset [Hanocka et al. 2019]. SubdivNet is the first method to correctly classify all test meshes.

Method	ModelNet40	Manifold40
PointNet++ [Qi et al. 2017a]	91.7%	87.9%
PCT [Guo et al. 2021]	93.2%	92.4%
SNGC [Haim et al. 2019]	91.6%	-
MeshNet [Feng et al. 2019]	91.9%	88.4%
MeshWalker [Lahav and Tal 2020]	92.3%	90.5%
SubdivNet (w/o majority voting)	-	91.2%
SubdivNet	-	91.5%

Table 3. Classification accuracy on ModelNet40 [Wu et al. 2015] and Manifold40. The first two rows are state-of-the-art point cloud methods with positions and normals as input. Other methods use meshes as input.

We trained and evaluated point cloud methods, MeshNet [Feng et al. 2019], MeshWalker [Lahav and Tal 2020], and SubdivNet on Manifold40; the results are shown in Table 3. Because of the reconstruction error and simplification distortion, Manifold40 is more challenging and the accuracy of all methods tested is lower. SubdivNet again outperforms all mesh-based methods on Manifold40. However, the Transformer-based method PCT [Guo et al. 2021] is more robust to distortion than the hierarchical networks.

5.3 Segmentation

In the mesh segmentation task, SubdivNet is trained to predict labels for every face. As remeshed input is used, the remeshed faces should be appropriately labeled before we can start training. To do so, we

	Method	Accuracy
Note A	Pointnet [Qi et al. 2017a]	74.7%
	Pointnet++ [Qi et al. 2017b]	82.3%
	MeshCNN [Hanocka et al. 2019]	87.8%
	MeshCNN (10000 faces)	65.3%
	PD-MeshNet [Milano et al. 2020]	86.9%
	Toric Cover [Maron et al. 2017]	88.0%
	SNGC [Haim et al. 2019]	91.3%
	PFCNN [Yang et al. 2020]	91.5%
	MeshWalker [Lahav and Tal 2020]	92.7%
	DiffusionNet [Sharp et al. 2021]	91.7%
	SubdivNet (w/o majority voting)	91.1%
SubdivNet	93.0%	
Note B	MeshCNN [Hanocka et al. 2019].	92.3%
	MeshWalker [Lahav and Tal 2020]	94.8%
	DiffusionNet [Sharp et al. 2021]	95.5%
	SubdivNet	96.6%
Note C	PD-MeshNet [Milano et al. 2020]	85.6%
	HodgeNet [Smirnov and Solomon 2021]	85.0%
	DiffusionNet [Sharp et al. 2021]	90.3%
	SubdivNet	91.7%

Table 4. Mesh segmentation accuracy on the human body dataset [Maron et al. 2017]. Note A: these methods use different types of input data; results were projected onto the raw meshes. Notes B, C: these methods were evaluated on the edges and faces of simplified meshes, respectively. See Appendix D for details of the projection and the metrics.

simply adopt a nearest-face strategy to build a mapping between the raw mesh and the remeshed one.

5.3.1 Human Body Segmentation. The human body dataset, labeled by [Maron et al. 2017], contains 381 training shapes from SCAPE [Anguelov et al. 2005], FAUST [Bogo et al. 2014], MIT [Vlasic et al. 2008], Adobe Fuse [Adobe.com 2021], and 18 test shapes from SHREC07 [Giorgi et al. 2007]. The human body is divided into 8 segments. In this case, we used Liu et al.’s method [Liu et al. 2020] to remesh the inputs to ensure lower distortion of details. Majority voting is employed in testing. Table 4 gives the results, which shows that our method outperforms other methods.

The standard input resolution of MeshCNN [Hanocka et al. 2019] is 1,500 faces. To find out whether the performance of MeshCNN is merely limited by the resolution, we additionally trained MeshCNN using 10,000-face inputs to enable a fair comparison.

Some examples of the segmentation results are visualized in Fig. 8. Compared to MeshCNN [Hanocka et al. 2019] and PD-MeshNet [Milano et al. 2020], SubdivNet more accurately segments parts, with more consistent boundaries.

5.3.2 COSEG. We also assessed SubdivNet on the three largest subsets of the COSEG shape dataset [Wang et al. 2012]: tele-aliens, chairs, and vases, which contain in turn 200, 400, and 300 models. They are segmented into only 3 or 4 parts, so we chose the MAPS algorithm as the remeshing method. We discovered that

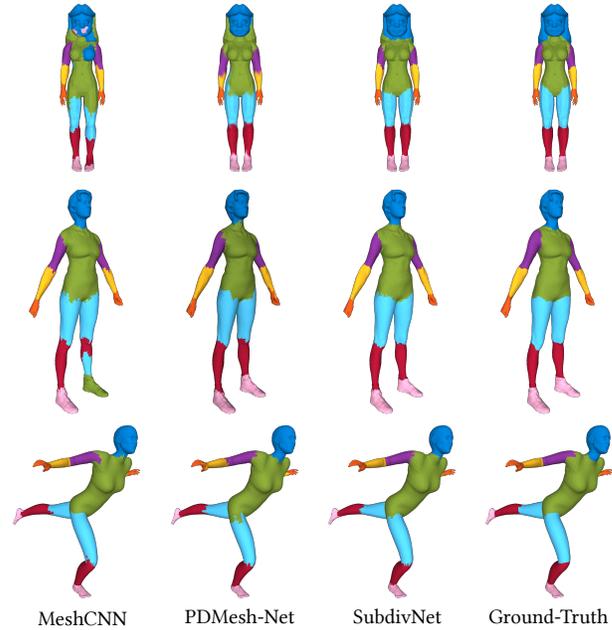


Fig. 8. Segmentation results from the human body dataset [Maron et al. 2017]. SubdivNet correctly classified all body parts, and gave more accurate boundaries.

Method	Vases	Chairs	Tele-aliens
MeshCNN [Hanocka et al. 2019]	85.2%	92.8%	94.4%
PD-MeshNet [Milano et al. 2020]	81.6%	90.0%	89.0%
SubdivNet	96.7%	96.7%	97.3%

Table 5. Mesh segmentation accuracy on the COSEG dataset [Wang et al. 2012]. The training-testing split for vases and chairs differs from [Hanocka et al. 2019]. Both MeshCNN and PD-MeshNet were trained on the new split, and results were projected to the raw meshes in COSEG.

the MeshCNN-generated ids for chairs and vases do not match those in the original COSEG dataset. Therefore, we followed the training-testing split of MeshCNN for tele-aliens, but randomly split the training and tests set for chairs and vases in a 4:1 ratio. SubdivNet was trained on the three datasets independently. Quantitative results are provided in Table 5 and example output is displayed in Fig. 9. Our method achieves a significant improvement over MeshCNN [Hanocka et al. 2019] and PD-MeshNet [Milano et al. 2020]. We also trained SubdivNet on another two random splits of vases and chairs. The mean accuracy and the standard deviation on the two datasets are $97.0\% \pm 0.6\%$ and $95.1\% \pm 1.5\%$, respectively.

5.4 Shape Correspondence

Our method can act as a robust feature extraction backbone for learning fine-grained shape correspondences between two meshes. We demonstrate via human body matching using FAUST [Bogo et al. 2014] and SCAPE [Anguelov et al. 2005] datasets. Specifically,

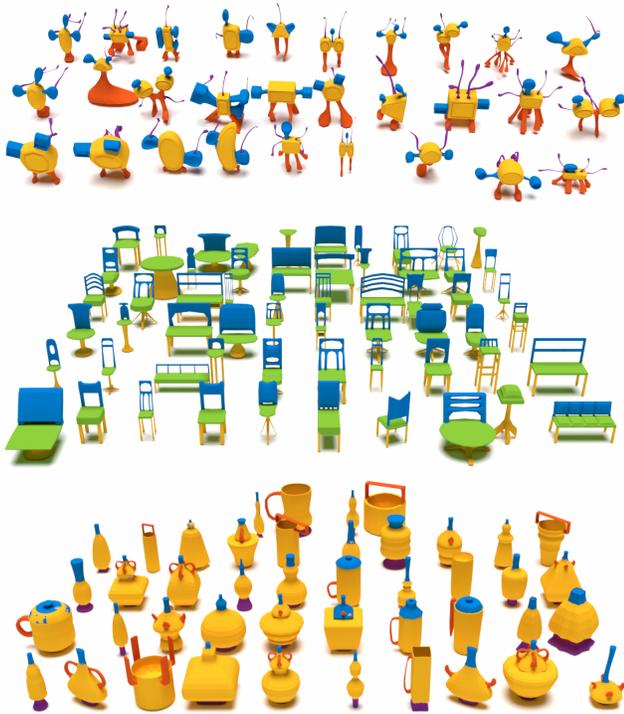


Fig. 9. Gallery of segmentation results for the COSEG dataset.

our network was trained to predict 3-dimensional canonical human coordinates in a similar way to [Mehta et al. 2017], but at mesh level. The set of predicted coordinates is treated as an \mathcal{R}^3 -valued function and the functional coordinates (dimension = 30) are computed based on the spectrum of the Laplace-Beltrami operator on the corresponding mesh. We then build a functional map, a representation for non-rigid shape matching, between the source and target meshes, by solving a linear system as in [Ovsjanikov et al. 2012]. Lastly, the map is refined using ZoomOut [Melzi et al. 2019] and converted back to point-to-point correspondences. Note that due to the scarcity of data, we additionally remeshed our geometries with different subdivision connectivities to augment training data, which helps the model to learn tessellation-invariant robust features.

Our evaluation protocol follows Bogo et al. [2014] and Donati et al. [2020]: The shape correspondence error is calculated as the normalized mean geodesic distance between predicted and ground-truth mapped target positions on the target mesh. The datasets, FAUST and SCAPE, were respectively divided into 80:20 and 51:20 training/test splits. Results from different combinations of the training and test sets are reported in Table 6 and results are visualized in Fig. 10. Our method achieves state-of-the-art matching and shows good generalizability across different datasets, demonstrating the effectiveness of the proposed method on this challenging task.

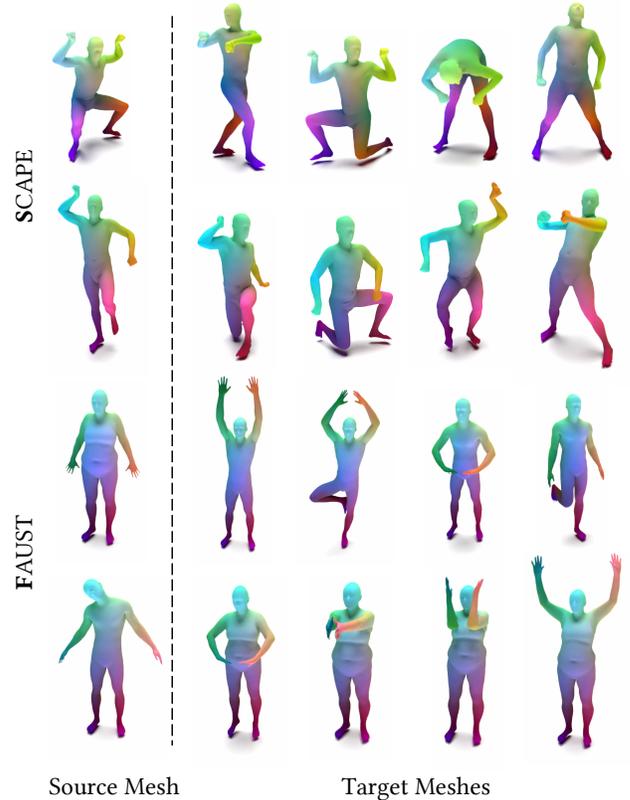


Fig. 10. Shape correspondences learned using different datasets. Each row shows the output matching from the source mesh to the target meshes on the right. Corresponding positions share the same color.

Method	F	S	F on S	S on F
BCICP [Ren et al. 2018]	15.	16.	-	-
ZoomOut [Melzi et al. 2019]	6.1	7.5	-	-
SURFMNet [Roufousse et al. 2019]	7.4	6.1	19.	23.
FMNet [Litany et al. 2017]	5.9	6.3	11.	14.
3D-CODED [Groueix et al. 2018]	2.5	31.	31.	33.
GeomFMaps [Donati et al. 2020]	1.9	3.0	9.2	4.3
Ours	1.9	3.0	10.5	2.6

Table 6. Shape correspondence error ($\times 100$) comparison. ‘F’ and ‘S’ indicate FAUST [Bogo et al. 2014] and SCAPE [Anguelov et al. 2005] datasets. ‘F on S’ means training on FAUST and testing on SCAPE and vice versa.

5.5 Further Evaluation

In this section, the key components of SubdivNet are evaluated on SHREC11 (split 10) [Wang et al. 2012] and the human body segmentation dataset [Maron et al. 2017].

5.5.1 Convolution Patterns and Network Architecture. Table 7 compares many networks with varied convolution patterns and architectures to see if 2D network architectures can aid 3D mesh learning. In

	Network Architecture	Kernel Size	Stride	Dilation	Accuracy
Classification	VGG-like	3	1	1	99.5%
	ResNet50	3,5	1,2	1	99.5%
Segmentation	U-Net [Ronneberger et al. 2015]	3	1	1	89.5%
	DeepLabv3+[Chen et al. 2018]	3, 5	1, 2	1,6,12,18	93.0%
	DeepLabv3+ w/o strided convolution	3	1	1,6,12,18	92.8%
	DeepLabv3+ w/o dilation	3, 5	1, 2	1	92.6%

Table 7. Ablation study on convolution kernel size, stride, dilation, and network architectures, evaluated on the SHREC11 (split 10, without voting) [Wang et al. 2012] and the human body segmentation dataset [Maron et al. 2017]. In the last two networks, the special convolutions are replaced by basic convolutions.

Base Size	Subdivision Depth	Faces	Accuracy
48	3	3072	99.3%
96	3	6144	99.1%
48	4	12288	99.5%

Table 8. Classification accuracy on SHREC11 (split 10) [Wang et al. 2012] using different input size.

Input	Accuracy
shape descriptor only	90.4%
pose descriptor only	92.5%
full input	93.0%

Table 9. Ablation study on input features, evaluated on a human body segmentation dataset [Maron et al. 2017].

2D vision, the performance gap between segmentation networks is more significant than the classification networks (see the CityScape benchmark [Cordts et al. 2016] and ImageNet benchmark [Deng et al. 2009]). Unsurprisingly, network architecture transfer is also more useful in 3D segmentation. Large kernel size and dilation are shown to be effective through the segmentation ablation studies.

5.5.2 Input Resolution. To examine the robustness of our network to input size, we tried several combinations of the base size and subdivision depth of inputs and retrained the network on *Split10*. Table 8 suggests that performances are quite close. Because remeshing distortion is more observable as the input size decreases, the results also show that our network can capture global shape and can tolerate input distortion to some extent.

However, more input faces lead to a heavier network and more training time. On the other hand, if the input size is too small, e.g. a 48-face base mesh with one time subdivision, some categories are quite similar, such as the cats and the dogs. In this case, the network cannot distinguish them well.

5.5.3 Input Features. Unlike image pixels, which have identical size and shape everywhere, shapes and sizes of mesh triangle faces represent local geometry. Thus, shape descriptors for the input features, i.e. areas, angles, and curvatures, are essential to the capability of

Method	Accuracy
DeepLabv3+ with feature propagation	93.06%
DeepLabv3+ without feature propagation	93.03%

Table 10. Ablation study for the feature propagation layer, based on the human body segmentation dataset [Maron et al. 2017].

SubdivNet. Table 9 indicates that both shape and pose are necessary for mesh learning.

5.5.4 Learning on the Raw Meshes. We implemented a variant of DeepLabv3+ with a feature propagation layer and additional convolution layers on the raw meshes. Table 10 shows that learning on the raw meshes slightly improves the segmentation quality. However, the extra layers require a 20% increase in computing time. Details can be found in Appendix A.

5.5.5 Computation Time and Memory Consumption. Our method was implemented with the Jittor deep learning framework [Hu et al. 2020]. Its flexibility enables efficient face neighbor indexing; convolution can be implemented with general matrix multiplication operators. As a result, the proposed network is as efficient as a 2D network.

We measured forward and backward propagation duration as well as GPU memory consumption on a 48-core CPU and a single TITAN RTX GPU. Table 11 shows that SubdivNet is more than 20 times faster than an edge-based approach [Hanocka et al. 2019] using less than a third of the amount of GPU memory, achieving comparable performance to a highly optimized 2D CNN [Chen et al. 2018].

5.6 3D shape retrieval from the real-world

The superior representation power of SubdivNet allows us to effectively extract global shape descriptors for arbitrary 3D meshes. We demonstrate this by retrieving 3D shapes from from partially-observed point clouds captured by an Asus Xtion Pro Live 3D sensor. By jointly embedding both point clouds and meshes into the latent feature space, shape retrieval can be implemented as nearest neighbor search in the Euclidean-structured latent manifold. Specifically, to build such a latent space, we first trained a denoising point cloud variational auto-encoder using the encoder architecture from [Qi

Network	Input Size (faces/pixels)	Time (ms)	GPU Memory (MB)
SubdivNet (DeepLabv3+)	16384	47.4	1221
MeshCNN	10000	1051.2	4090
2D DeepLabv3+	16384	20.1	612

Table 11. Computation time and GPU memory consumption of SubdivNet, MeshCNN [Hanocka et al. 2019], and the 2D DeepLabv3+ [Chen et al. 2018]. The number of layers in DeepLabv3+ was reduced to be the same as for SubdivNet. Numbers are averaged over 1000 data samples.

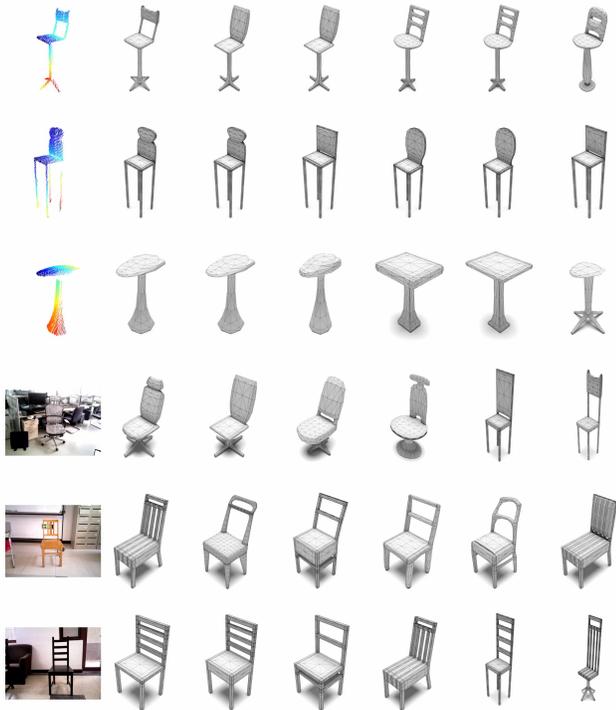


Fig. 11. Shape retrieval results using our method. Left: input point clouds. Other columns: retrieved results. The last three rows are from real-world RGB-D captures, where only the back-projected depth points are used as input. Retrieval results are ordered from left to right by similarity in feature space.

et al. 2017a] and the decoder network from [Fan et al. 2017]. The point clouds were synthesized from the mesh dataset with virtual cameras. Then we extracted the bottleneck features (dimension = 32) corresponding to all the point clouds in our dataset and used them to directly supervise SubdivNet, obtaining a mapping from mesh space to the latent space.

The chair models from the COSEG dataset [Wang et al. 2012] were used to train our network. Evaluation on the synthetic point clouds gives top1, top5, and top10 recall rates of 76.8%, 83.3%, and 88.0%, respectively. Fig. 11 shows retrieval results for both synthetic point clouds and real-world depth scans.

6 LIMITATIONS AND FUTURE WORK

6.1 Convolution

The origins of convolutions in 2D CNNs can be traced back to signal filtering. In our framework, we must take care to ensure that convolution is ordering-invariant, so we first transform the neighborhood features, rather than apply direct signal convolutions, leading to an isotropic kernel for all neighboring faces. Another consequence is that the number of convolution parameters does not increase with kernel size as for 2D convolutions. One possible strategy is to differentiate faces by their distances from the center.

6.2 Subdivision Connectivity

Subdivision connectivity plays a crucial role in SubdivNet, providing a uniform feature aggregation scheme from local to global. We believe it is a key factor in our method surpassing other mesh learning methods. However, remeshing is necessary to apply our method to an arbitrary mesh. As discussed in Sec. 4, both remeshing approaches used have limitations, with a trade-off between mesh quality and base mesh size. Processing imperfect meshes, polygon soups, objects with borders, and large-scale scenes with flaws requires more thought. Adaptive remeshing [Lee et al. 1998] may be more helpful than the current uniform remeshing. We also note that majority voting improves the final performance, showing that the differences between the remeshed shape and the raw mesh affect the results to some extent. Better methods of remeshing [Sharp et al. 2019] are needed, or better, ways of downsampling without needing remeshing at all.

6.3 Applications

This paper demonstrates SubdivNet’s effectiveness on single shape analysis, but the current network cannot be directly applied on large-scale scenes due to the limitation of the remeshing technique. Yet the proposed convolution is promising for being integrated into a scene network, because it does not rely on subdivision connectivity. Apart from the applications in the paper, our ideas could also potentially be applied to traditional geometric problems, such as mesh smoothing and denoising, deformation, registration of multiple meshes, etc. They could also be employed in specific areas that require human knowledge or professional skill. For example, we could learn the natural right way up for a mesh, or choose the best orientation of a mesh for 3D printing [Ezair et al. 2015].

7 CONCLUSIONS

This work has presented a novel deep learning framework, SubdivNet, for 3D geometric learning on meshes. The core of SubdivNet is a general and flexible mesh convolution using a mesh pyramid structure for effective feature aggregation. We first utilize self-parameterization to remesh the input mesh to have Loop subdivision sequence connectivity. That allows a well-defined, uniform mesh hierarchy to be constructed over the input shape. We then use mesh convolution operators which support user-specified kernel size, stride, and dilation. Pooling and upsampling are also naturally supported by subdivision connectivity. This enables the direct application of well-known 2D image CNNs to mesh learning.

Our evaluations indicate that SubdivNet surpasses existing mesh learning approaches in both accuracy and efficiency.

REFERENCES

- Adobe.com. 2021. *Animate 3D characters for games, film, and more*. Retrieved January 24, 2021 from <https://www.mixamo.com>
- Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: shape completion and animation of people. *ACM Trans. Graph.* 24, 3 (2005), 408–416.
- Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. 2014. FAUST: Dataset and Evaluation for 3D Mesh Registration. In *CVPR*. 3794–3801.
- Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael M. Bronstein. 2016. Learning shape correspondence with anisotropic convolutional neural networks. In *NIPS*. 3189–3197.
- Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.* 34, 4 (2017), 18–42.
- E. Catmull and J. Clark. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10, 6 (1978), 350–355.
- Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *ECCV*. 801–818.
- Y. Chen, J. Zhao, C. Shi, and D. Yuan. 2020. Mesh Convolution: A Novel Feature Extraction Method for 3D Nonrigid Object Classification. *IEEE Transactions on Multimedia* (2020), 1–1. <https://doi.org/10.1109/TMM.2020.3020693>
- François Chollet. 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In *CVPR*. 1800–1807.
- Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*. Eurographics, 129–136.
- Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*. 3213–3223.
- Angela Dai and Matthias Nießner. 2019. Scan2Mesh: From Unstructured Range Scans to 3D Meshes. In *CVPR*. 5574–5583.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*. 248–255.
- Nicolas Donati, Abhishek Sharma, and Maks Ovsjanikov. 2020. Deep Geometric Functional Maps: Robust Feature Learning for Shape Correspondence. In *CVPR*. 8589–8598.
- Daniel Doo and Malcolm Sabin. 1978. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design* 10, 6 (1978), 356–360.
- Nira Dyn, David Levine, and John A. Gregory. 1990. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graph.* 9, 2 (1990), 160–169.
- Ben Ezair, Fady Massarwi, and Gershon Elber. 2015. Orientation analysis of 3D objects toward minimal support volume in 3D-printing. *Computers & Graphics* 51 (2015), 117–124.
- Danielle Ezuz, Justin Solomon, Vladimir G. Kim, and Mirela Ben-Chen. 2017. GWCNN: A Metric Alignment Layer for Deep Shape Analysis. *Comput. Graph. Forum* 36, 5 (2017), 49–57.
- Haoqiang Fan, Hao Su, and Leonidas J. Guibas. 2017. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *CVPR*. 605–613.
- Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. 2019. MeshNet: Mesh Neural Network for 3D Shape Representation. In *AAAI*. 8279–8286.
- Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In *SIGGRAPH*. 209–216.
- Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. 2007. Shape retrieval contest 2007: Watertight models track. *SHREC competition* 8, 7 (2007).
- Shunwang Gong, Lei Chen, Michael M. Bronstein, and Stefanos Zafeiriou. 2019. SpiralNet++: A Fast and Highly Efficient Mesh Convolution Operator. In *ICCV Workshops*. 4141–4148.
- Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 2018. 3D-CODED: 3D Correspondences by Deep Deformation. In *ECCV*. 230–246.
- Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Taijiang Mu, Ralph R. Martin, and Shi-Min Hu. 2021. PCT: Point cloud transformer. *Computational Visual Media* 7, 2 (2021), 187–199.
- Igor Guskov, Andrei Khodakovskiy, Peter Schröder, and Wim Sweldens. 2002. Hybrid meshes: multiresolution using regular and irregular refinement. In *Symposium on Computational Geometry*. 264–272.
- Igor Guskov, Kiril Vidimce, Wim Sweldens, and Peter Schröder. 2000. Normal meshes. In *SIGGRAPH*. 95–102.
- Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. 2019. Surface Networks via General Covers. In *ICCV*. 632–641.
- Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: a network with an edge. *ACM Trans. Graph.* 38, 4 (2019), 90:1–90:12.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.
- Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. 2020. Deep geometric texture synthesis. *ACM Trans. Graph.* 39, 4 (2020), 108.
- Shi-Min Hu, Dun Liang, Guo-Ye Yang, Guo-Wei Yang, and Wen-Yang Zhou. 2020. Jitter: a novel deep learning framework with meta-operators and unified graph execution. *Science China Information Sciences* 63, 12 (2020), 222103:1–222103:21.
- Jingwei Huang, Hao Su, and Leonidas J. Guibas. 2018. Robust Watertight Manifold Surface Generation Method for ShapeNet Models. *CoRR abs/1802.01698* (2018).
- Jingwei Huang, Haotian Zhang, Li Yi, Thomas A. Funkhouser, Matthias Nießner, and Leonidas J. Guibas. 2019. TextureNet: Consistent Local Parameterizations for Learning From High-Resolution Signals on Meshes. In *CVPR*. 4440–4449.
- Andrei Khodakovskiy, Nathan Litke, and Peter Schröder. 2003. Globally smooth parameterizations with low distortion. *ACM Trans. Graph.* 22, 3 (2003), 350–357.
- Roman Klokov and Victor S. Lempitsky. 2017. Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. In *ICCV*. 863–872.
- Leif Kobbelt. 2000. 3-subdivision. In *SIGGRAPH*. 103–112.
- Leif Kobbelt, Jens Vorsatz, Ulf Lابسik, and Hans-Peter Seidel. 1999. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces. *Comput. Graph. Forum* 18, 3 (1999), 119–130.
- Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. 2018. Surface Networks. In *CVPR*. 2540–2548.
- Alon Lahav and Ayellet Tal. 2020. MeshWalker: deep mesh understanding by random walks. *ACM Trans. Graph.* 39, 6 (2020), 263:1–263:13.
- Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence C. Cowsar, and David P. Dobkin. 1998. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *SIGGRAPH*. 95–104.
- Xianzhi Li, Ruihui Li, Lei Zhu, Chi-Wing Fu, and Pheng-Ann Heng. 2020. DNF-Net: a Deep Normal Filtering Network for Mesh Denoising. *CoRR abs/2006.15510* (2020).
- Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution On X-Transformed Points. In *NeurIPS*. 828–838.
- Chunfeng Lian, Li Wang, Tai-Hsien Wu, Mingxia Liu, Francisca Durán, Ching-Chang Ko, and Dinggang Shen. 2019. MeshSNet: Deep Multi-scale Mesh Feature Learning for End-to-End Tooth Labeling on 3D Dental Surfaces. In *MICCAI*. 837–845.
- Zhouhui Lian, Alexander Godil, Benjamin Bustos, Mohamed Daoudi, Jeroen Hermans, Shun Kawamura, Yukinori Kurita, Guillaume Lavoué, Hien Van Nguyen, Ryutarou Ohbuchi, Yuki Ohkita, Yuya Ohishi, Fatih Porikli, Martin Reuter, Ivan Sipiran, Dirk Smeets, Paul Suetens, Hedi Tabia, and Dirk Vandermeulen. 2011. SHREC '11 Track: Shape Retrieval on Non-rigid 3D Watertight Meshes. In *3DOR@Eurographics*. Eurographics Association, 79–88.
- Isaak Lim, Alexander Dielen, Marcel Campen, and Leif Kobbelt. 2018. A Simple Approach to Intrinsic Correspondence Learning on Unstructured 3D Meshes. In *ECCV Workshops*. 349–362.
- Or Litany, Tal Remez, Emanuele Rodolà, Alexander M. Bronstein, and Michael M. Bronstein. 2017. Deep Functional Maps: Structured Prediction for Dense Shape Correspondence. In *ICCV*. 5660–5668.
- Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. 2020. Neural subdivision. *ACM Trans. Graph.* 39, 4 (2020), 124.
- Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.* 25, 3 (2006), 681–689.
- Zheng-Ning Liu, Yan-Pei Cao, Zheng-Fei Kuang, Leif Kobbelt, and Shi-Min Hu. 2021. High-Quality Textured 3D Shape Reconstruction with Cascaded Fully Convolutional Networks. *IEEE Transactions on Visualization and Computer Graphics* 27, 1 (2021), 83–97.
- Charles Loop. 1987. Smooth Subdivision Surfaces Based on Triangles. *Masters Thesis, Department of Mathematics, University of Utah* (January 1987).
- Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G. Kim, and Yaron Lipman. 2017. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.* 36, 4 (2017), 71:1–71:10.
- Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. 2015. Geodesic Convolutional Neural Networks on Riemannian Manifolds. In *ICCV Workshops*. 832–840.
- Daniel Maturana and Sebastian A. Scherer. 2015. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *IROS*. IEEE, 922–928.
- Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Trans. Graph.* 36, 4 (2017), 1–14.
- Simone Melzi, Jing Ren, Emanuele Rodolà, Abhishek Sharma, Peter Wonka, and Maks Ovsjanikov. 2019. ZoomOut: spectral upsampling for efficient shape correspondence. *ACM Trans. Graph.* 38, 6 (2019), 155:1–155:14.

Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. 2020. Primal-Dual Mesh Convolutional Neural Networks. In *NeurIPS*, Vol. 33. 952–963.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *CVPR*. 5425–5434.

Federico Monti, Oleksandr Shchur, Aleksandar Bojchevski, Or Litany, Stephan Günnemann, and Michael M. Bronstein. 2018. Dual-Primal Graph Convolutional Networks. *CoRR* abs/1806.00770 (2018). arXiv:1806.00770

Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph.* 31, 4 (2012), 1–11.

Adrien Poulernard and Maks Ovsjanikov. 2018. Multi-directional geodesic neural networks via equivariant convolution. *ACM Trans. Graph.* 37, 6 (2018), 236:1–236:14.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*. 77–85.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NIPS*. 5099–5108.

Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. 2018. Generating 3D Faces Using Convolutional Mesh Autoencoders. In *ECCV*. 704–720.

Jing Ren, Adrien Poulernard, Peter Wonka, and Maks Ovsjanikov. 2018. Continuous and orientation-preserving correspondences via functional maps. *ACM Trans. Graph.* 37, 6 (2018), 248:1–248:16.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, Vol. 9351. 234–241.

Jean-Michel Roufousse, Abhishek Sharma, and Maks Ovsjanikov. 2019. Unsupervised Deep Learning for Structured Shape Matching. In *ICCV*. 1617–1627.

Scott Schaefer, E. Vouga, and Ron Goldman. 2008. Nonlinear subdivision through nonlinear averaging. *Computer Aided Geometric Design* 25, 3 (2008), 162–180.

Jonas Schult, Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. 2020. DualConvMesh-Net: Joint Geodesic and Euclidean Convolutions on 3D Meshes. In *CVPR*. 8609–8619.

Nicholas Sharp, Souhaib Attaiqi, Keenan Crane, and Maks Ovsjanikov. 2021. Diffusion-Net: Discretization Agnostic Learning on Surfaces. arXiv:2012.00888 [cs.CV]

Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019. Navigating intrinsic triangulations. *ACM Trans. Graph.* 38, 4 (2019), 55:1–55:16.

Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. 2015. DeepPano: Deep Panoramic Representation for 3-D Shape Recognition. *IEEE Signal Process. Lett.* 22, 12 (2015), 2339–2343.

Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*. <https://arxiv.org/abs/1409.1556>

Ayan Sinha, Jing Bai, and Karthik Ramani. 2016. Deep Learning 3D Shape Surfaces Using Geometry Images. In *ECCV*. 223–240.

Dmitriy Smirnov and Justin Solomon. 2021. HodgeNet: learning spectral geometry on triangle meshes. *ACM Trans. Graph.* 40, 4 (2021), 166:1–166:11.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In *ICCV*. 945–953.

Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. 2018. Tangent Convolutions for Dense Prediction in 3D. In *CVPR*. 3887–3896.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*. <https://openreview.net/forum?id=rjXmpikCZ>

Daniel Vlastic, Ilya Baran, Wojciech Matusik, and Jovan Popovic. 2008. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.* 27, 3 (2008), 97.

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *ECCV*. 52–67.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: octree-based convolutional neural networks for 3D shape analysis. *ACM Trans. Graph.* 36, 4 (2017), 72:1–72:11.

Yunhai Wang, Shmulik Asafi, Oliver van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2012. Active co-analysis of a set of shapes. *ACM Trans. Graph.* 31, 6 (2012), 165:1–165:10.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.* 38, 5 (2019), 146:1–146:12.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*. 1912–1920.

Yun-Peng Xiao, Yu-Kun Lai, Fang-Lue Zhang, Chungpeng Li, and Lin Gao. 2020. A survey on deep geometry learning: From a representation perspective. *Computational Visual Media* 6, 2 (2020), 113–133.

Haotian Xu, Ming Dong, and Zichun Zhong. 2017. Directionally Convolutional Networks for 3D Shape Segmentation. In *ICCV*. 2717–2726.

Yuqi Yang, Shilin Liu, Hao Pan, Yang Liu, and Xin Tong. 2020. PFCNN: Convolutional Neural Networks on 3D Surfaces Using Parallel Frames. In *CVPR*. 13575–13584.

Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas. 2017. SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation. In *CVPR*. 6584–6592.

A NETWORK IMPLEMENTATION

Here, we show in detail how the proposed mesh convolution defined above can be integrated into 2D network architectures to provide solutions for general 3D tasks such as mesh classification and segmentation. The code is publicly available at <https://xxx.xxx.xxx>.

Convolution Neighborhood Indexing. When the kernel size > 3 , the neighborhood can be found by depth-first search (DFS). Because a face is exactly adjacent to three neighbors, the search process results in a binary DFS tree. The rearrangement is then obtained by in-order traversal of the binary tree. As neighbors can be indexed in parallel, the process is efficient.

Classification Network. We implemented a VGG-like network, which simply has two blocks of basic convolution, batch normalization, and ReLU layers at each resolution. Max-pooling is used for downsampling. Experimentally, we find this simple convolutional network provides sufficient performance. Thus, we do not choose a more sophisticated architecture, e.g. ResNet.

DeepLabv3+. Because the raw training meshes (before remeshing) typically have far fewer triangles than pixels in a 2D image, we simply use ResNet50 [He et al. 2016] as a feature extractor instead of inception [Chollet 2017], both for efficiency and to avoid overfitting. The kernel size and stride of the first convolution are also lowered to 5 and 2, respectively, and we reduce the number of downsampling layers to 3. One key component of DeepLabv3+ is the atrous spatial pyramid pooling (ASPP) that stacks multiple dilated convolutions to enlarge the receptive field. We use dilations in our experiments of 1, 6, 12, and 18. The purple boxes in Fig. 12 depict the network architecture.

DeepLabv3+ with Feature Propagation. With the feature propagation layer and convolutions on the raw mesh (see the dashed orange box in Fig. 12), we can achieve a complete end-to-end pipeline. The extra layers increase computation by about 20%.

Input shape. For the classification task, the base mesh size is 48, similar to the 7×7 output feature map of the last convolution in VGG and ResNet. The subdivision depth is set to 4, resulting in the finest mesh having 12288 faces. For dense prediction tasks, the base size and subdivision depth are 256 and 3, respectively, reaching a similar number of faces of raw meshes in human body segmentation to balance the prediction quality and computational efficiency. We find these hyper-parameters work well for evaluation on public datasets. However, for meshes with higher resolution, one may choose a larger subdivision depth for better results.

Training. The Adam optimizer is employed to train the networks. In all experiments, we trained the network 4 times on the training set and reported the best results on the test set.

B FURTHER REMESHING DETAILS

To construct the subdivision sequence connectivity, the MAPS algorithm [Lee et al. 1998] establishes a bijective map between the raw mesh and the decimated version. In detail, MAPS iteratively removes the maximum independent set of vertices. When a vertex

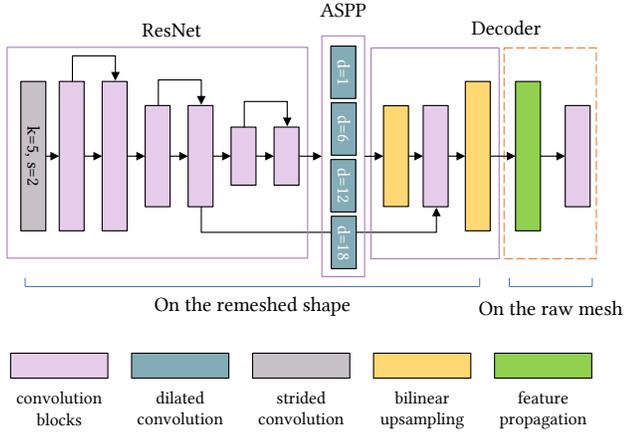


Fig. 12. DeepLabv3+ architecture. If the layers in the dashed orange box are enabled, the network can output per-face features on the raw mesh.

is removed, MAPS first re-triangulates the 1-ring neighbors, and calculates a conformal map over the local region between the before and after states. The removed vertices are also parameterized on the decimated mesh. After the raw mesh has been simplified into a base mesh, a global parameterization is constructed. Then Loop subdivision [Loop 1987] without vertex update is applied to the base mesh d times, where d is the subdivision depth, and the vertices of the subdivided mesh are projected onto the raw mesh using the global parameterization. Because the global parameterization links the raw mesh and the simplified mesh, this idea is also called *self-parameterization*. One obvious advantage of MAPS is that it supports any genus as long as the decimation process does not break the topology.

Recently, Liu et al. [2020] proposed a modified MAPS method, utilizing edge collapse based decimation, e.g. qslim [Garland and Heckbert 1997], rather than vertex removal, which improves the decimation quality.

In practice, we find both methods have limitations. In MAPS, the order of vertex removal is crucial. For example, repeated removal of limb vertices in a horse mesh will lead to insufficient sampling of the limbs in the output, and ultimately the hooves cannot be fully reconstructed. This causes significant distortion if the mesh contains small but important details (see Fig. 6). Liu et al. [2020] tackle the issue with a better decimation algorithm and prohibit collapses that cause poor triangle quality. However, doing so restricts the lowest base size the algorithm can reach (see Fig. 7).

More importantly, *UV flip* may occur, so that a triangle face on the original mesh cannot be mapped to a triangular region in the parameter domain. Sampling on flipped *UV* triangles may lead to remeshing failure. In detail, when removing a vertex, MAPS first flattens the vertex and its one-ring neighbors. The flattening, or local parameterization, may be invalid because of *UV flip* (Fig. 13(a) shows a case). Because global parameterization is the composition of a sequence of local parameterizations, *UV* triangle flip is more probable when the base size is lower. Fortunately, *UV flip* does not occur if the three vertices lie on the same face of the current decimated

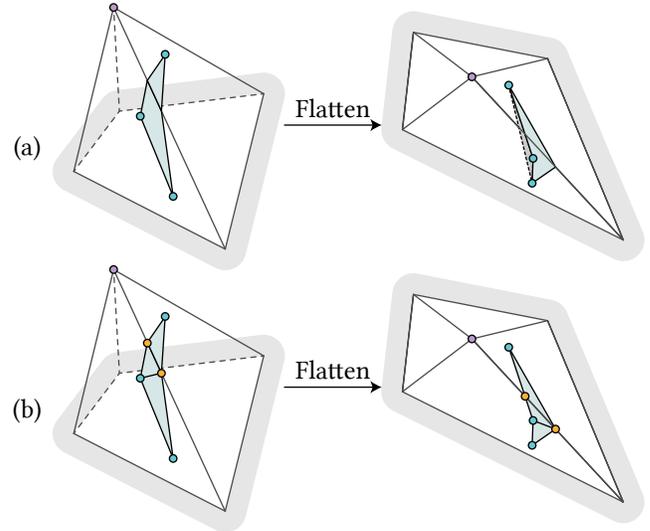


Fig. 13. *UV Flip*. (a) Example of *UV flip* after local parameterization. The three blue vertices form a face in the input mesh, already removed in earlier steps. Now they are parameterized in the current simplified mesh. (b). Splitting the blue triangle into three triangles with orange vertices prevents the problem.



Fig. 14. Examples from Manifold40.

mesh. Thus, to avoid *UV flip*, during decimation and parameterization, we split problematic triangles along with the triangulation of the current decimated mesh. For example, in Fig. 13(b), the triangle with three blue vertices is divided into three smaller triangles as it crosses an edge of the simplified mesh.

However, Liu et al. [2020] do not solve this issue. Instead, if *UV flip* occurs after collapsing an edge and parameterizing the related region, they simply abandon this edge collapse operation and move to the next candidate edge to be collapsed. However, as more vertices of the raw mesh are parameterized to the simplified mesh, *UV flip* may become inevitable if the input mesh is over-decimated. However, in 2D CNNs, the size of the feature map of the last layer is often very small, e.g. 7×7 pixels in ResNet [He et al. 2016]. Thus, for some inputs, Liu et al.'s method cannot meet our requirements.

C MANIFOLD40

We employed a reconstruction algorithm [Huang et al. 2018] to make the models in ModelNet40 watertight. This method constructs an octree for the shape, extracts isosurfaces, and projects vertices onto the original shape. Small and isolated components are cleaned after reconstruction. Sometimes, we found that non-manifold vertices may occur; we split these vertices using MeshLab [Cignoni et al. 2008]. Following the datasets contributed by [Hanocka et al. 2019], all meshes were simplified to contain exactly 500 faces. Simplifying meshes also speeds up the remeshing process. Fig. 14 illustrates some examples from Manifold40.

We found some shapes in Manifold40 have a genus larger than 20. Because the decimation process of remeshing does not change the topology of shapes, it is almost impossible to reach a base size of 48 for such shapes. Therefore, we chose a more tolerant strategy: the base size of most meshes is enlarged to 96, and shapes with complicated topology are decimated as little as possible. As a result, except for 11 training samples, the base size of all other meshes was between 96 and 192. We simply discarded those 11 samples when training SubdivNet. To avoid heavy demands on computational resources, the depth was reduced to 3 from 4.

Using variable base sizes leads to variable input sizes. To incorporate them in a conventional batch-based training scheme, we padded meshes with empty faces to ensure all inputs have the same size in a mini-batch. Because the global pooling layer after convolutions does not restrict the mesh to have a fixed number of faces, SubdivNet can be trained and evaluated with variable input sizes.

D EVALUATION METRICS FOR MESH SEGMENTATION

In the segmentation experiments, different inputs and evaluation metrics are employed by the approaches compared. The original human body segmentation contains up to 30k faces in a mesh. However, in COSEG, the number of faces ranges from hundreds to thousands. Both human body segmentation and COSEG datasets offer per-face labels on meshes. MeshCNN [Hanocka et al. 2019] simplifies the meshes to 1500 faces, and projects the segmentations onto the edges on simplified meshes. PD-MeshNet [Milano et al. 2020] and HodgeNet [Smirnov and Solomon 2021] also employ simplified datasets but use face labels. To fairly compare SubdivNet with other approaches, we report the performance of SubdivNet using three evaluation metrics.

Per-face Accuracy. The metric is the overall accuracy on faces of the original meshes before simplification. It is also the default metric presented in this paper. For rows marked by Note A in Table 4, we mapped other forms of segmentations to the original meshes. In detail, for point cloud methods [Qi et al. 2017a,b], we uniformly sampled 4096 points on the mesh surface. Then the segmentations on point clouds are projected to faces on meshes by finding the nearest point. For PD-MeshNet, the face label on the original mesh is obtained by finding the nearest face center in the simplified mesh. For MeshCNN, we first generate per face segmentation on the simplified meshes with edge labels, weighed by edge lengths. Then the nearest query strategy is used.

Per-edge Soft Accuracy on Simplified Meshes. The metric is from MeshCNN, which collects the overall accuracy on edges with a soft

criterion: an edge’s prediction is true if it equals any of its neighbor’s ground truth labels. We directly projected our segmentation results to the edges of MeshCNN’s simplified meshes by querying the nearest edge center. Accuracy is calculated by public code for MeshCNN. Rows marked Note B in Table 4 use this metric. Table 12 presents an additional experiment on COSEG under this metric.

Per-face Hard Accuracy on Simplified Meshes. The metric is the overall accuracy on faces of the simplified meshes. The segmentation results of SubdivNet are also directly projected onto the simplified meshes. Rows marked Note C in Table 4 use this metric.

Method	HumanBody	COSEG Vases	COSEG Chairs	COSEG Tele-aliens
MeshCNN [Hanocka et al. 2019]	92.3%	92.7%	98.1%	97.6%
MeshWalker [Lahav and Tal 2020]	94.8%	-	-	99.1%
SubdivNet	96.6%	98.1%	99.5%	99.4%

Table 12. Mesh segmentation accuracy using MeshCNN’s metric. Vases and chairs use a train-test split different from that originally used by MeshCNN.