Directionally Convolutional Networks for 3D Shape Segmentation

Haotian Xu, Ming Dong, Zichun Zhong
Department of Computer Science, Wayne State University
Detroit, MI, USA
htxu@wayne.edu, mdong@wayne.edu, zichunzhong@wayne.edu

Abstract

Previous approaches on 3D shape segmentation mostly rely on heuristic processing and hand-tuned geometric descriptors. In this paper, we propose a novel 3D shape representation learning approach, Directionally Convolutional Network (DCN), to solve the shape segmentation problem. DCN extends convolution operations from images to the surface mesh of 3D shapes. With DCN, we learn effective shape representations from raw geometric features, i.e.*, face normals and distances, to achieve robust segmentation. More specifically, a two-stream segmentation framework is proposed: one stream is made up by the proposed DCN with the face normals as the input, and the other stream is implemented by a neural network with the face distance histogram as the input. The learned shape representations from the two streams are fused by an element-wise product. Finally, Conditional Random Field (CRF) is applied to optimize the segmentation. Through extensive experiments conducted on benchmark datasets, we demonstrate that our approach outperforms the current state-of-the-arts (both classic and deep learning-based) on a large variety of 3D shapes.*

1. Introduction

Segmentation over 3D shapes, also known as compositional part-based reasoning on 3D shapes, plays an important role in computer graphics and computer vision. It has been applied to various applications, such as 3D modeling [32], 3D object detection [17, 27], 3D scene understanding [13], and human pose estimation [24]. In the past few years, many methods have been proposed to segment 3D shapes into semantic parts. Among these approaches, they either rely on heuristic processing and hand-engineering geometric features [2, 19], or apply co-segmentation schemes based on geometric characteristics of 3D shapes [26, 14]. More recently, (convolutional) neural networks have been applied to 3D shape segmentation [31, 9].

Inspired by the remarkable success of applying Convolutional Neural Network (CNN) in image recognition tasks, a few approaches have been proposed to extent convolution to graphs [5, 8, 7], most of which operate convolutions in the spectral domain - taking convolution as a linear operator in the Fourier space of a graph. However, as mentioned in [7], a convolution filter defined in the spectral domain is not naturally localized and the translations are very costly. For approaches that define convolution in the spatial domain, they require relatively weak regularity assumptions on the graph and utilize the advantage of graphs, *i.e.*, having localized neighborhoods. However, the method in [5] only works for a given domain as eigenbases vary arbitrarily from shape to shape.

In this paper, we propose Directionally Convolutional Network (DCN) that extends convolution operations from images to the surface mesh in the spatial domain. As a special case of graphs, polygon meshes inherit the advantage of being natural to define localized neighborhoods. Furthermore, we introduce a two-stream framework combining proposed DCN and a neural network (NN) for segmentation of 3D shapes. Instead of fusing the two streams by a simple concatenation, we take our framework as an approximation of a directed graph and combine the probabilities inferred by the two streams with an element-wise product. Finally, Conditional Random Field (CRF) is applied to optimize the surface mesh segmentation. The main contributions of this paper are summarized as follows:

- By defining rotation-invariant convolution and pooling operation on the surface of 3D shapes, we learn effective shape representations from raw geometric features, *i.e.*, face normals and distances, to achieve robust segmentation of 3D shapes.
- Based on the proposed DCN, we introduce a two-stream framework (shown in Fig. 1) to classify each face of a given mesh into predefined semantic parts. Our approach achieves state-of-the-art segmentation results on a large variety of 3D shapes.

In the rest of the paper, we first review related work in Section 2. Then, we describe details of DCN in Section 3.

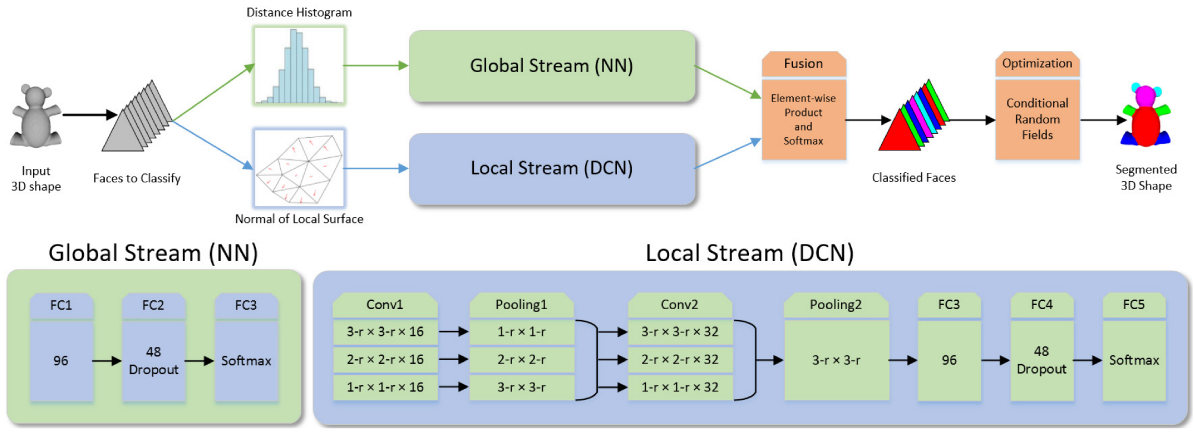

Figure 1: Workflow of our two-stream framework for 3D shape segmentation. Given a 3D shape, we learn representation for each face in the mesh and classify it into the predefined semantic parts. Shape representations are learned with the proposed two-stream framework (DCN and NN) from raw geometric descriptors: face normals and distance histogram. The learned features are fused with an element-wise product and optimized by CRF for segmentation.

In Section 4, we show our two-stream segmentation framework. In Section 5, we compare our approach with the current state-of-the-art methods and report our results on benchmark datasets. At last, we conclude in Section 6.

2. Related Work

2.1. 3D Shape Segmentation

Nowadays, 3D shape segmentation and labeling are widely used in computer graphics and computer vision fields. We can generally divide existing methods into the following categories.

Traditional feature-based approaches. Some early studies aim to manually design a single geometric descriptor that is effective in mesh segmentation [2, 19]. However, a single descriptor is insufficient to deal with various kinds of 3D shapes.

Co-segmentation approaches. In order to address the aforementioned limitation, data-driven approaches are utilized to extract common geometric features, including unsupervised co-segmentation methods [26, 14], and (semi-)supervised methods [12, 30]. These learning-based approaches generally outperform single geometric features. However, the simple combinations of geometric features are still not robust enough to describe complicated 3D shapes in many cases.

CNN-based approaches. Recently, neural networks have been popularly employed in 3D model analysis, due to their capabilities in extracting effective representations from low-level features. Xie et al. [31] proposed a shallow network to learn high-level features for segmentation, but this approach does not offer better performance than standard shallow classifiers [11]. Guo et al. [9] and Shu et al. [25] utilized CNNs to learn high-level features from hand-engineering descriptors. These approaches simply concatenate hand-tuned features and lack geometric spatial coherence.

Kalogerakis et al. [11] proposed a view-based deep architecture for 3D shape segmentation and achieved state-of-the-art performance. However, their approach suffers from strong requirements on view selection, *i.e.*, minimizing occlusions, covering shape surface, and guaranteeing each part of shapes is visible in at least three views.

Comparing to the aforementioned methods, our approach only relies on the two most fundamental geometric descriptors for 3D shape representation learning: one preserves local precision and the other preserves global spatial consistency. Instead of combining the same kind of feature at different scales as in [22], we combine two different kinds of features. Furthermore, DCN defines convolution and pooling operations on 3D shape surfaces directly and thus has clearer geometric coherence than previous methods.

2.2. Convolutional Networks on Graphs

Inspired by the great success of CNNs in computer vision tasks, several approaches have been proposed to extend convolutional networks from images to arbitrarily structured graphs [5, 10, 21, 3, 7, 4].

Bruna et al. [5], Henaff et al. [10] and Defferrard et al. [7] proposed spectral networks which utilize spectral graph theory to define graph convolution as multiplication of a filter and graph node values in the Fourier space. The spatial network proposed in [5] is based on a hierarchical clustering of a graph. However, this approach does not have an efficient strategy of weight sharing across different locations of

the graph [5]. By contrast, the proposed DCN operates convolution and pooling on the surface mesh of 3D shapes in the spatial domain and thus does not require strong regularity assumptions on the input shape structure [5]. Moreover, it is natural to define a face and its neighbors in the mesh as a cluster for convolution filters, which provides efficient weight sharing.

Some works also defined convolution on the surface mesh of 3D shapes for shape correspondence. Masci et al. [21] took geometry vector as input and had to compute the convolution for all possible filter rotations due to angular coordinate ambiguity. Boscaini et al. [3] took local SHOT descriptor as input. In contrast to these earlier works, our method learns the shape representation (layer by layer and coarse to fine) from the most fundamental low-level geometric features. Furthermore, the proposed DCN is rotation-invariant and does not have the angular coordinate ambiguity problem in [21].

3. Directional Convolution and Pooling

In this section, we present the details of directional convolution and pooling methods defined on surface meshes of 3D shapes, and how to generalize to cloud points.

3.1. Mesh Face Normal and Curvature

In geometry, curvatures can effectively represent the local shape variations. The local directions of minimum and maximum curvatures indicate the slowest and steepest variation of the surface normal, respectively. In this subsection, we define the fundamental low-level geometric features on each face based on surface normal and curvature.

Face normal: For each mesh face f_i, let $\{\mathbf{v}_{f_{i_1}}, \mathbf{v}_{f_{i_2}}, \mathbf{v}_{f_{i_3}}\}$ denote its vertices. The face f_i's normal can be represented using the cross product of two edge vectors as follows:

$$\mathbf{n}_{f_i} = (\mathbf{v}_{f_{i_2}} - \mathbf{v}_{f_{i_1}}) \times (\mathbf{v}_{f_{i_3}} - \mathbf{v}_{f_{i_1}}). \quad (1)$$

As mentioned in [23], we can estimate the local shape variations and properties over a local region by using the differences in face normals, which is similar to estimate the curvatures of a face.

Face curvature: The mesh vertex curvature magnitudes and directions are computed based on [1]. We can use the average value of three vertex's curvatures (including magnitude and direction) to approximate the minimum and maximum curvatures on mesh face f_i as follows:

$$k_i = \frac{(k_{i_1} + k_{i_2} + k_{i_3})}{3} \text{ and } \mathbf{d}_i = \frac{(\mathbf{d}_{i_1} + \mathbf{d}_{i_2} + \mathbf{d}_{i_3})}{3}, \quad (2)$$

where magnitude k_i and direction $\mathbf{d}_i$ can be used to represent the minimum and maximum curvatures on face f_i, respectively.

3.2. Directional N-ring Face Neighbors

In order to define a convolution on the surface mesh, we need to define a robust rotation-invariant face neighboring mechanism at first. There are two aspects needed to be defined in the concept of *directional n-ring face neighbors*: (1) The set of n-ring face neighbors: the nth ring of a face i is the set of faces that are at distance $n-1$ from f_i in the given mesh, where the distance n is the minimum number of edges between two faces. (2) The order of n-ring face neighbors: the order of neighbors is important in convolutions since filter weights are adaptive according to their significance.

As mentioned above, local directions of curvatures indicate the local shape variation. No matter how the model rotates, the local shape geometry is invariant. So, we choose the curvature direction as a guidance to define the order of face neighbors. For face f_i, we traverse the neighbors ring by ring based on the direction of maximum curvature counter-clockwise, and the first face neighbor for each ring is the one having the minimal angle difference between two vectors, *i.e.*, the maximum curvature direction and the vector $\mathbf{c}_{ij}$ defined by the centroids of faces f_i and f_j. The angle between two vectors can be computed by using the geometric definition of dot product, *i.e.*, $\theta_{ij} = \cos^{-1}\left(\frac{\mathbf{d}_i^{max} \cdot \mathbf{c}_{ij}}{\|\mathbf{d}_i^{max}\| \|\mathbf{c}_{ij}\|}\right)$. Fig. 2 illustrates the first nth rings of neighbors of face i under the defined order on a 3D hand mesh model.

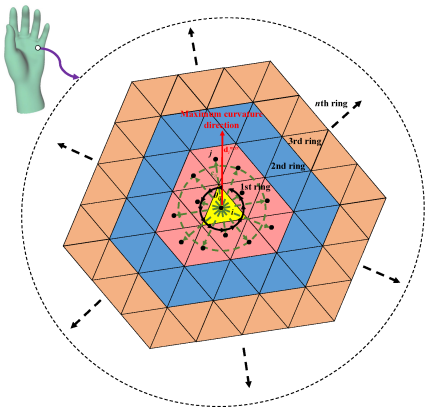

Figure 2: The illustration of the first nth rings of neighbors of face f_i under the defined order on a 3D Hand mesh model. Yellow triangle is face f_i, pink triangles are 1st ring neighbors, blue triangles are 2nd ring neighbors, and brown triangles are 3rd ring neighbors, etc.

3.3. Directional Convolution on Mesh

Once directional n-ring face neighbors are defined, we can present the definition of *directional convolution* of the

feature ϕ with a kernel w on mesh face f_i as follows:

$$(\phi * w)(i) = \frac{1}{K} \sum_{j \in N_n(i)} w(j)\phi(j), \tag{3}$$

where ϕ can be a scale or vector function based on the mesh face features, such as normal, curvature, shape diameter, etc. In this paper, we only use the face normal vectors as the feature. Face normals are computed in Eq. (1). The kernel w weighs the participation of neighbouring faces f_j, which will be learned during the optimization of DCN. K is the normalization factor, *i.e.*, $K = \sum_{j \in N_n(i)} w(j)$. $N_n(i)$ is the set of neighbors of face f_i. n is the index of ring for face neighbors. The order of neighbors is computed as in Section 3.2.

We define the filter size as n-r$\times n$-r, which means that a face and its first n rings of neighbors are convolved by the filter. If $n = 0$, then only one face is convolved by the convolution filter. Since neighboring face number in n-ring of different faces varies, we choose the average neighbor number as filter size for n-ring, and pad zeros for faces without enough neighbors (or omit redundant neighbors).

3.4. Pooling on Mesh

Classic pooling layers in CNN make use of the natural multi-scale clustering of grid: they input all the feature maps over a cluster, and output a single feature for that cluster [5]. On surface mesh, we define a cluster as a face and its 1- to n-ring neighbors. Thus, given such a cluster, the pooling is manipulated by a downsampling strategy of *a cluster of faces to 1* and denoted as n-r$\times n$-r pooling. For max pooling, the maximum value of feature maps in the cluster is taken as output. Similarly, the mean normal value is taken as the output for average pooling.

3.5. Generalization to Cloud Points

Although meshes and point clouds are two different representations of 3D objects, we can easily modify the proposed method to segment 3D point clouds. Specifically, we can first use principal component analysis to compute the local point normal and curvatures[1]. Then, we define the point neighbors by finding the k nearest points. Finally, we can employ the proposed directional convolution on point clouds, same as on meshes.

4. 3D Segmentation with DCN

In this section, we describe our shape segmentation approach in detail (see Fig. 1). First, we compute normals and distances for faces in a given 3D shape. Second, we feed these raw features to the proposed two-stream framework with DCN and NN, and then fuse the two streams by an element-wise product and softmax. Finally, the segmentation is optimized by CRF.

[1] http://pointclouds.org/documentation/tutorials/normal_estimation.php

4.1. Input Features

In our approach, we aim to learn an effective 3D shape representation, robust for a large variety of shapes. Two types of input geometric features, face normals and distance histogram, are utilized in order to ensure local precision and global spatial consistency, respectively.

4.1.1 Face normal as local features

Normal is one of the most fundamental geometric features to describe the shape of a surface mesh. We select face normals to ensure the local precision of the segmentation. To capture the local shape information of a surface at a higher level of details, we extract a patch of the target face and its first n rings of neighbors. Generally, taking a very small patch as the input is insufficient to accurately describe the local geometry. A larger patch will help but typically leads to inefficiency in computing. Empirically, we choose $n = 6$. The normals of faces in the patch are used as the local input features of the surface patch centered on the target face.

4.1.2 Distance histogram as global features

For segmentation over the same category of 3D shapes, semantic parts consistently preserve the same relative positions in all the models. Thus, including global information is likely to yield improvements. Although simply increasing the size of local patches would cover larger part of a 3D shape, it is computationally inefficient. Another strategy is to use the coordinates of face centroids, but this scheme is not shift-, scale-, or rotation-invariant. In this paper, we use normalized histograms of the pairwise face distances to ensure the global spatial consistency.

To define pairwise distances, we first denote an input 3D shape dataset of M models as $D = \{S_1, S_2, \ldots, S_M\}$. For a 3D shape S_m with N_m faces to segment, we denote each face as $f_i^{S_m}$, where $m \in [1, M]$ and $i \in [1, N_m]$. We build a dual graph $S_m^{'}$ with N_m vertices, in which each vertex corresponds to a face of S_m and two vertices are connected by an edge if and only if the two corresponding faces share at least one vertex in S_m. The pairwise distance between two faces $f_i^{S_m}$ and $f_j^{S_m}$ is denoted as $d_{i,j}$, which is the shortest distance between corresponding vertices $\mathbf{v}_i^{S_m^{'}}$ and $\mathbf{v}_j^{S_m^{'}}$ in the dual graph $S_m^{'}$. Since our input is "water-tight" polygon meshes, every two vertices in the same 3D shape are connected by one or more edges. Thus, the existence of the pairwise distance between every two faces in the same shape is guaranteed.

In this way, we can get $N_m - 1$ pairwise distances for face $f_i^{S_m}$. Then, a histogram is computed based on these distances. Empirically, we choose a 50-bin histogram. Finally, we perform L_2 normalization on the 50 elements of each histogram, making the distance histogram insensitive to the total number of faces in a 3D shape. Unlike coordinates, normalized distance histograms are robust to scaling and invariant to shifting and rotation.

4.2. Two-stream Framework with DCN and NN

In the proposed two-stream segmentation framework, the local stream is a DCN with the face normal as the input, and the global stream is a neural network with the distance histogram as the input. Then we fuse the two streams with an element-wise product and softmax.

Local stream DCN. The architecture of the proposed DCN is design based on the idea of multi-scale and multi-level feature ensembling. Limited by the size of experimental datasets, the network consists of only two convolution and pooling layers and three fully connected layers.

Specifically, an input patch to DCN contains a center face and its 1- to 6-ring neighbors. To get multi-scale features, we first employ three sizes of convolution filters, *i.e.*, 3-r×3-r, 2-r×2-r, and 1-r×1-r, of stride 1 in layer Conv1. A sliding max pooling of stride 1 [18] is separately applied for the three feature maps in layer Pooling1. Since neighbors that are closer to the target face carry more information and less noise, we only keep the center and first 3 neighboring rings of three feature maps and concatenate them together. In Conv2 and Pooling2, we employ three sizes of filters of stride 1 and average pooling to flatten the feature map. Finally, three fully connected layers with dropout and softmax are used and output classification probabilities P_{local}. P_{local} is a real-valued vector of size C, where C is the number of predefined classes.

Global stream NN. The global stream is implemented by a three-layer neural network with dropout and softmax, which takes the distance histogram as input. We denote the output softmax scores of the global stream as P_{global}, which represents the probabilities of an input face belonging to segmentation classes. Similar to P_{local}, P_{global} is the vector of probabilities inferred based on distance histograms.

Fusion in a graphical-model style. Assuming that the local feature and global feature of the same face are independent, we take our two-stream segmentation framework as a directed graphical model. That is, the probability of segmentation classes P_{seg} can be computed by an element-wise product:

$$P_{seg} = \sigma(P_{local} \circ P_{global}), \tag{4}$$

where $\sigma(\cdot)$ denotes the softmax function.

4.3. Mesh Label Optimization with CRF

Given P_{seg} for each triangle in a test 3D shape S_m with N_m faces, we employ CRF [16] to refine the labels by incorporating the constraint on label consistency. The energy function is

$$E(\mathbf{x}) = \sum_i \xi_i(x_i) + \lambda \sum_{i,j} \xi_{ij}(x_i, x_j), \tag{5}$$

where x is the label assignment for faces and λ is a non-negative constant. In particular, we set $\lambda = 50$ as in [9]. Here, we define the unary item $\xi_i(x_i)$ as

$$\xi_i(x_i) = -\log P(x_i), \tag{6}$$

where $P(x_i)$ is the ith element in P_{seg} of face f_i. Thus, assigning f_i to a class with low probability will result in a high penalty.

We define the pairwise item $\xi_{ij}(x_i, x_j)$ as

$$\xi_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } i = j \\ -\log(\theta_{ij}/\pi) d_{ij} & \text{otherwise} \end{cases}, \tag{7}$$

where θ_{ij} and d_{ij} are the dihedral angle and distance between triangle face f_i and f_j, respectively. With this pairwise item, we penalize the smoothness between the labels of adjutant face pairs.

5. Experimental Results

5.1. Datasets and Experimental Setups

In this section, we compare our approach with current state-of-the-arts on the segmentation of a large variety of shapes. The following benchmark datasets are employed to evaluate our approach: Princeton Segmentation Benchmark (PSB) [6] (19 categories, 20 meshes per category) and four categories from the Shape COSEG Dataset [29], including Iron (18 meshes), Lamp (20 meshes), Candelabra (28 meshes), and Guitar (44 meshes). We also perform our experiments on two large categories, Vases (300 meshes) and Chairs (400 meshes).

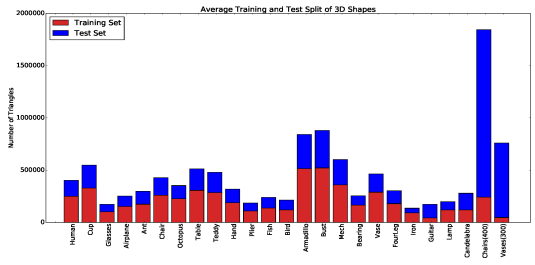

Figure 3: The average training and testing triangle face split for all categories of 3D shapes.

We followed the experimental setup of [9]. For small categories from PSB and the Shape COSEG, we take 12 meshes for training and the remaining for testing. For the two large categories, we take 20 meshes as the training set for Vases and 50 meshes for Chairs. For each category, we repeat our approach three times and report the average accuracy and standard deviation. The average number of faces used in the training and testing dataset is shown in Fig. 3.

In our experiments, we compared our approach with several state-of-the-arts (both classic and deep learning-based), including Sidi et al. [26], Kalogerakis et al. [12], Wang et al. [30], Guo et al. [9], and shapePFCN [11]. Besides, we also combined the two streams at penultimate layers as an alternative approach (named *Ours-early* in Tables).

Our network is implemented using Python and Theano [28]. Adam [15] with learning rate 10^{-4} is applied for optimization. Our deep learning framework runs on a GTX 980Ti GPU, and it takes about 20 minutes to train a model with 20K-30K faces, excluding the pre-processing time of computing the face normals and distance histograms.

5.2. Directional vs. Non-directional Convolutions

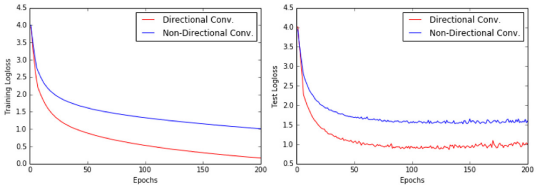

Figure 4: Training logloss (left) and test logloss (right) on 3D Human shapes with directional convolution (red curves) and non-directional convolution (blue curves).

First, we show that directional convolution is necessary for shape representation learning. In Fig. 4, we compare the training and test loss of directional convolutions with non-directional convolutions on one exemplar category: Human. For the latter one, the input triangle faces are randomly shuffled. Clearly, the network with the directional convolution converges faster and to a lower error in both the training (12 meshes) and the testing (8 meshes) datasets.

5.3. Segmentation Accuracy

We compare the segmentation accuracy between our approaches and the current state-of-the-arts. Following [9], the accuracy is computed as the percentage of area of correctly classified faces over area of all the surface. The performance of existing methods is based on publicly reported results in the literature. Best results are marked in bold.

The segmentation performance on the small datasets is shown in Table 1. Our early fusion approach (Ours-early) performs 0.0026% weaker than late fusion approach (Ours). On average, our framework gains an improvement of 1.13% against the best-performing previous approach. When it comes to each category, our approaches outperform all existing methods on 15 out of 23 objects. In the remaining 8 categories, the accuracies of our method are only a bit less than the prior-best. For the two large datasets, *i.e.*, Chairs and Vases, their segmentation accuracies are listed in Table 2. From the table, we can see that our approaches outperform previous approaches significantly, which benefits from the high learning capacity of our two-stream framework with DCN and NN.

5.4. Visualization of DCN Kernels and Feature Maps

To visualize kernels learned in DCN, we convolve the trained filters in layer Conv1 over the surfaces of all 3D shapes in the corresponding category and find out the patches with strongest response for each filter (see Fig. 5). Clearly, 1) filters of smaller size tend to focus on the fine details of a shape, *e.g.*, steep surface changes; 2) filters of larger size tend to focus on the main trend of the shape, the matched patches are more smooth.

Moreover, to better understand what the proposed two-stream segmentation framework learns from face normals and distance histograms, we randomly select $10,000$ patches from each 3D shape category and feed them into the trained network. Then, we extract the 48-dimensional feature maps of layer FC4 in the local stream and FC2 in the global stream. We also apply element-wise multiplication on extracted local and global feature maps and use the product as final feature maps. The three types of feature maps for category Teddy and Ant are visualized by t-SNE [20], which embeds high-dimensional feature maps in a 2D space while preserving the pairwise distance of the instances. As shown in Fig. 6, the clusters from different parts overlap with each other with global features, indicating a high similarity. That is mainly due to the symmetry of Teddy and Ant. By contrast, the clusters with local features are better separated. Combining global and local features together, the t-SNE clusters are best separated in both categories, which clearly indicates that our framework learned effective shape representations for segmentation.

5.5. Segmentation Examples

Fig. 7 demonstrates some exemplar segmentation results of the proposed framework. In this figure, our approach performs well on shapes with rotation and different poses. However, the edges between different parts are not smooth, which means that some faces locating at the edges are challengingly segmented. Apparently, if two neighboring faces with similar face normal and distance histogram are located at the edge of two separate parts, our approach has difficulty classifying them correctly.

Table 1: Mesh segmentation accuracy on PSB and four additional categories from Shape COSEG. The performance of previous methods is from their papers, and "-" means the result was not reported.

Category	Kalogerakis[12]	Wang[30]	Guo[9]	ShapePFCN[11]	Ours-early	Ours
Human	0.9320	0.5560	0.9122	0.9380	0.9317(±0.0397)	**0.9408**(±0.0088)
Cup	0.9960	0.9960	0.9973	0.9370	0.9973(±0.0002)	**0.9979**(±0.0003)
Glasses	0.9720	-	0.9760	0.9630	0.9792(±0.0338)	**0.9869**(±0.0020)
Airplane	0.9610	-	0.9667	0.9250	0.9739(±0.0049)	**0.9766**(±0.0078)
Ant	0.9880	-	0.9880	0.9890	**0.9898**(±0.0012)	**0.9898**(±0.0008)
Chair	0.9840	**0.9960**	0.9867	0.9810	0.9941(±0.0048)	0.9935(±0.0051)
Octopus	0.9840	-	0.9879	0.9810	0.9891(±0.0013)	**0.9934**(±0.0007)
Table	0.9930	**0.9960**	0.9955	0.9930	0.9952(±0.0005)	0.9959(±0.0003)
Teddy	0.9810	-	0.9824	0.9650	**0.9911**(±0.0019)	0.9908(±0.0022)
Hand	0.8870	-	**0.8871**	0.8870	0.8796(±0.0034)	0.8861(±0.0028)
Plier	0.9620	-	0.9622	0.9570	0.9710(±0.0031)	**0.9714**(±0.0054)
Fish	0.9560	-	0.9564	0.9590	0.9688(±0.0025)	**0.9705**(±0.0016)
Bird	0.8790	-	0.8835	0.8630	0.9037(±0.0102)	**0.9039**(±0.0096)
Armadillo	0.9010	-	0.9227	0.9330	0.9361(±0.0020)	**0.9382**(±0.0012)
Bust	0.6210	-	0.6984	0.6640	0.7644(±0.0328)	**0.7898**(±0.0266)
Mech	0.9050	0.9130	0.9560	**0.9790**	0.9698(±0.0011)	0.9660(±0.0012)
Bearing	0.8660	-	0.9246	0.9120	0.9429(±0.0027)	**0.9470**(±0.0036)
Vase	0.8580	**0.9050**	0.8911	0.8570	0.8916(±0.0078)	0.8931(±0.0089)
FourLeg	0.8620	0.5430	0.8702	**0.8950**	0.8803(±0.0077)	0.8742(±0.0083)
Iron	-	-	**0.9737**	0.8770	0.9689(±0.0018)	0.9714(±0.0022)
Guitar	-	-	0.9715	0.9790	0.9847(±0.0043)	**0.9932**(±0.0037)
Lamp	-	-	0.9628	0.9090	0.9774(±0.0024)	**0.9789**(±0.0007)
Candelabra	-	-	0.9447	**0.9630**	0.9603(±0.0113)	0.9546(±0.0048)
Average	0.9204	0.8436	0.9409	0.9263	0.9496	**0.9522**

Table 2: Mesh segmentation accuracy on large datasets.

Category	Sidi[26]	Kim[14]	Guo[9]	Ours-early	Ours
Chairs(400)	0.8020	0.9120	0.9252	0.9518(±0.0014)	**0.9573**(±0.0013)
Vases(300)	0.6990	0.8560	0.8854	0.9034(±0.0042)	**0.9086**(±0.0060)

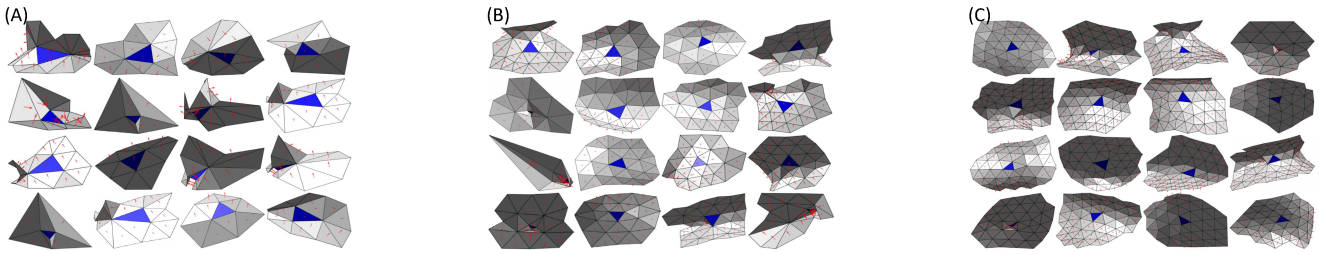

Figure 5: Strongest responses of convolution filters in Conv1 of DCN. The center face of each patch is marked blue and normal of each face is drawn in red. A: Best matches of 1-r×1-r filters. B: Best matches of 2-r×2-r filters. C: Best matches of 3-r×3-r filters.

Another way to provide an intuitive understanding of the segmentation results is to visualize probability maps of layer FC3 in the global stream and of layer FC5 in the local stream. For a better comparison, we also include segmentations from the two-stream framework in the figure (last column in Fig. 8). For the global stream, the segmentation suffers from the symmetry of the shapes. Taking shape Human as an example, faces near ankle are classified into

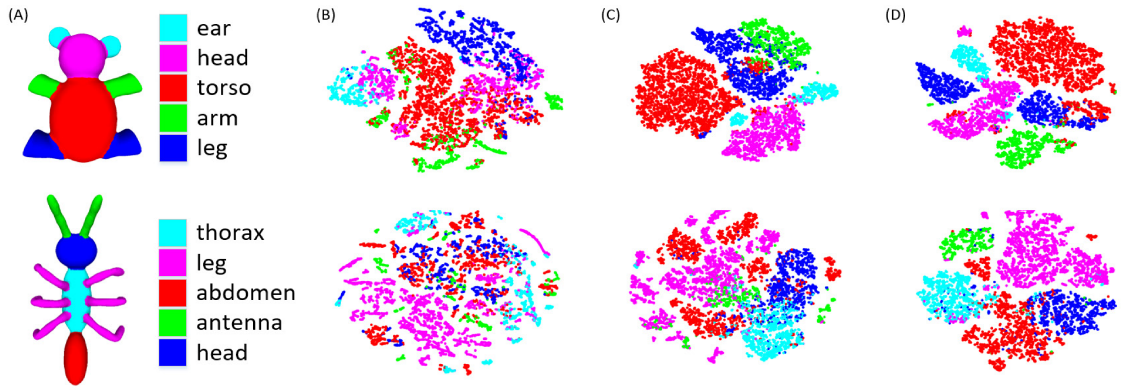

Figure 6: t-SNE visualization of global and local representations learned by proposed framework with NN and DCN. A: Segmentations of our framework. B: Feature maps learned in FC2 of global stream (NN). C: Feature maps learned in FC3 of local stream (DCN). D: The element-wise product of feature maps from local and global steams. Best viewed in color.

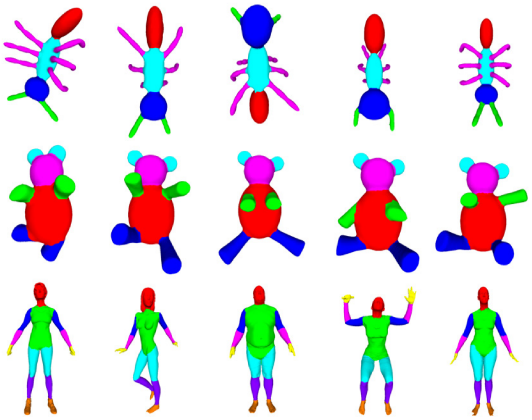

Figure 7: Visualization of segmentation on category Ant, Teddy, and Human.

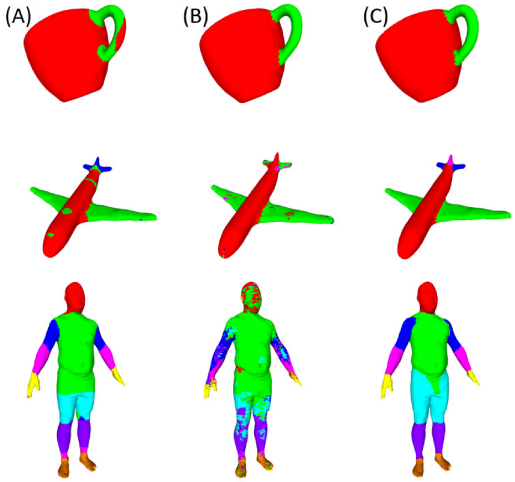

Figure 8: Visualization of segmentation results inferred by different streams. A: Segmentations from the global stream NN. B: Segmentations from the local stream DCN. C: Segmentations from the combination of the two streams.

lower arm. That is because they are all near the end of body and are not differentiable based on distance histograms. For the local stream DCN, it performs well when face normals of patches belonging to different classes are quite distinct (*e.g.*, category Cup). However, when the surfaces of different classes are similar or change smoothly, the performance is weak (*e.g.*, category Human). That is because it is difficult to tell lower arm from upper arm with a small patch of surface. In this case, the global stream plays a more important role. By fusing the two streams, we make the best use of local and global input features for segmentation.

6. Conclusion

In this paper, we proposed a novel 3D shape representation learning approach, Directionally Convolutional Network (DCN). Based on a two-stream segmentation framework, we learn effective shape representations from raw geometric features, *i.e.*, face normals and distances, for robust segmentation. Our method achieved the state-of-the-art results on a large variety of 3D shapes in benchmark datasets.

Limited by the size of datasets, our approach is based on patches, which is time-consuming. In the future, we plan to integrate Fully Convolutional Networks and CRF to build an end-to-end learning framework and apply our method on larger 3D shape datasets.

Acknowledgment This work was partially supported by US National Science Foundation (NSF) under grant CNS-1637312 and ACI-1657364, and by Ford Motor Company University Research Program under grant 2015-9186R.

References

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 485–493. ACM, 2003. 3

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522, 2002. 1, 2

[3] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 3189–3197, 2016. 2, 3

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

[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014. 1, 2, 3, 4

[6] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. In *ACM Transactions on Graphics (ToG)*, volume 28, page 73. ACM, 2009. 5

[7] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3837–3845, 2016. 1, 2

[8] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015. 1

[9] K. Guo, D. Zou, and X. Chen. 3D mesh labeling via deep convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 35(1):3, 2015. 1, 2, 5, 6, 7

[10] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 2

[11] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3D shape segmentation with projective convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 6, 7

[12] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics (TOG)*, 29(4):102, 2010. 2, 6, 7

[13] B.-s. Kim, P. Kohli, and S. Savarese. 3D scene understanding by voxel-CRF. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1425–1432, 2013. 1

[14] V. G. Kim, W. Li, N. J. Mitra, S. Chaudhuri, S. DiVerdi, and T. Funkhouser. Learning part-based templates from large collections of 3D shapes. *ACM Transactions on Graphics (TOG)*, 32(4):70, 2013. 1, 2, 7

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015. 6

[16] J. Lafferty, A. McCallum, F. Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML*, volume 1, pages 282–289, 2001. 5

[17] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3D object detection with RGBD cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1417–1424, 2013. 1

[18] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2016. 5

[19] R. Liu, H. Zhang, A. Shamir, and D. Cohen-Or. A part-aware surface metric for shape analysis. In *Computer Graphics Forum*, volume 28, pages 397–406. Wiley Online Library, 2009. 1, 2

[20] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 6

[21] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. 2, 3

[22] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3376–3385, 2015. 2

[23] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, pages 486–493. IEEE, 2004. 3

[24] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013. 1

[25] Z. Shu, C. Qi, S. Xin, C. Hu, L. Wang, Y. Zhang, and L. Liu. Unsupervised 3D shape segmentation and co-segmentation via deep learning. *Computer Aided Geometric Design*, 43:39–52, 2016. 2

[26] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Transactions on Graphics (TOG)*, 30(6):1, 2011. 1, 2, 6, 7

[27] S. Song and J. Xiao. Deep sliding shapes for amodal 3D object detection in RGB-D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016. 1

[28] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. 6

[29] Y. Wang, S. Asafi, O. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31(6):165, 2012. 5

[30] Y. Wang, M. Gong, T. Wang, D. Cohen-Or, H. Zhang, and B. Chen. Projective analysis for 3D shape segmentation.

ACM Transactions on Graphics (TOG), 32(6):192, 2013. 2, 6, 7

[31] Z. Xie, K. Xu, L. Liu, and Y. Xiong. 3D shape segmentation and labeling via extreme learning machine. In *Computer graphics forum*, volume 33, pages 85–95. Wiley Online Library, 2014. 1, 2

[32] T. Xue, J. Liu, and X. Tang. Example-based 3D object reconstruction from line drawings. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 302–309. IEEE, 2012. 1

[33] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.