3D Mesh Labeling via Deep Convolutional Neural Networks

KAN GUO, DONGQING ZOU, and XIAOWU CHEN
State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University

This article presents a novel approach for 3D mesh labeling by using deep Convolutional Neural Networks (CNNs). Many previous methods on 3D mesh labeling achieve impressive performances by using predefined geometric features. However, the generalization abilities of such low-level features, which are heuristically designed to process specific meshes, are often insufficient to handle all types of meshes. To address this problem, we propose to learn a robust mesh representation that can adapt to various 3D meshes by using CNNs. In our approach, CNNs are first trained in a supervised manner by using a large pool of classical geometric features. In the training process, these low-level features are nonlinearly combined and hierarchically compressed to generate a compact and effective representation for each triangle on the mesh. Based on the trained CNNs and the mesh representations, a label vector is initialized for each triangle to indicate its probabilities of belonging to various object parts. Eventually, a graph-based mesh-labeling algorithm is adopted to optimize the labels of triangles by considering the label consistencies. Experimental results on several public benchmarks show that the proposed approach is robust for various 3D meshes, and outperforms state-of-the-art approaches as well as classic learning algorithms in recognizing mesh labels.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: 3D mesh labeling, deep convolutional neural networks, geometry features

ACM Reference Format:

Kan Guo, Dongqing Zou, and Xiaowu Chen. 2015. 3D mesh labeling via deep convolutional neural networks. ACM Trans. Graph. 35, 1, Article 3 (December 2015), 12 pages.
DOI: http://dx.doi.org/10.1145/2835487

1. INTRODUCTION

In computer graphics, mesh understanding plays an important role to build and process 3D models. To effectively understand a mesh, a key step is to identify the probable object part that each triangle on the mesh belongs to. This step, denoted as mesh labeling, aims to infer the inherent characteristics of the mesh and can be applied to various applications such as mesh editing [Yu et al. 2004; Chen et al. 2015a], modelling [Chen et al. 2013, 2015b] and deformation [Yang et al. 2013]. As a consequence, mesh labeling is now becoming one of the hottest yet challenging research areas.

In the past few decades, many approaches have been proposed for mesh labeling (e.g., Kalogerakis et al. [2010], Huang et al. [2013], and Wang et al. [2013]). In these approaches, a triangle on a mesh is usually characterized by heuristically designed geometry features. For example, Shapira et al. [2010] utilized shape diameter function as a signature to partition a 3D object. They defined a context-aware distance measure through the calculated signatures and found part analogies among numerous objects. Subsequently, Sidi et al. [2011] proposed an unsupervised cosegmentation method, which involved a predefined feature set to achieve per-object segmentation first. After that, they embedded the initial segments into a common space by using diffusion maps to obtain the final cosegmentation of the mesh set. Generally speaking, these approaches can achieve promising performance in many cases. However, each class of geometric features usually works only for very limited types of 3D meshes. Due to the limited feature space and variety, such heuristic features often lack the ability to generate accurate and consistent mesh labels for various types of meshes. Therefore, a key problem in mesh labeling is to construct powerful features that can provide accurate labeling results for various types of meshes with enhanced generalization ability.

To address this problem, a feasible solution is to *learn* an effective and robust mesh representation from data, which has been proved to be effective in classical recognition tasks [Lyu and Simoncelli 2008; Zeiler et al. 2011]. Kalogerakis et al. [2010] proposed a data-driven approach to label the mesh by optimizing the Conditional Random Field (CRF). By using labeled meshes for training, they successfully learned different types of segmentations for different tasks without human interaction. Xie et al. [2014] proposed a fast method for 3D shape segmentation and labeling. They employed a set of feature descriptors and the Extreme Learning Machine (ELM) to train a neural networks classifier for mesh labeling. However, the feature combination strategies learnt from CRF and ELM are usually somewhat simple, which may prevent these approaches from being used in complex labeling tasks.

According to the mechanisms of human vision system, it is suggested that a deep structure can be used to extract compact, effective and robust feature representations from the input data [Deng 2004; Baker et al. 2009]. Inspired by that, we resort to the deep learning algorithm to extract powerful representations for mesh labeling. In this article, we propose to learn mesh representations from a large pool of geometric features with the deep Convolutional Neural Networks (CNNs). As shown in Figure 1, CNNs can be viewed as a deep architectures composed of multiple nonlinear transformations [Bengio 2009]. In our approach, a large pool of classical geometric features are first extracted to characterize each triangle on the mesh. To facilitate the training process and utilize the convolutional

This work was partially supported by NSFC (61325011&61532003), 863 Program (2013AA013801), and SRFDP (20131102130002).
Authors' address: State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing, China. X. Chen (corresponding author) email: chen@buaa.edu.cn.

DOI: http://dx.doi.org/10.1145/2835487

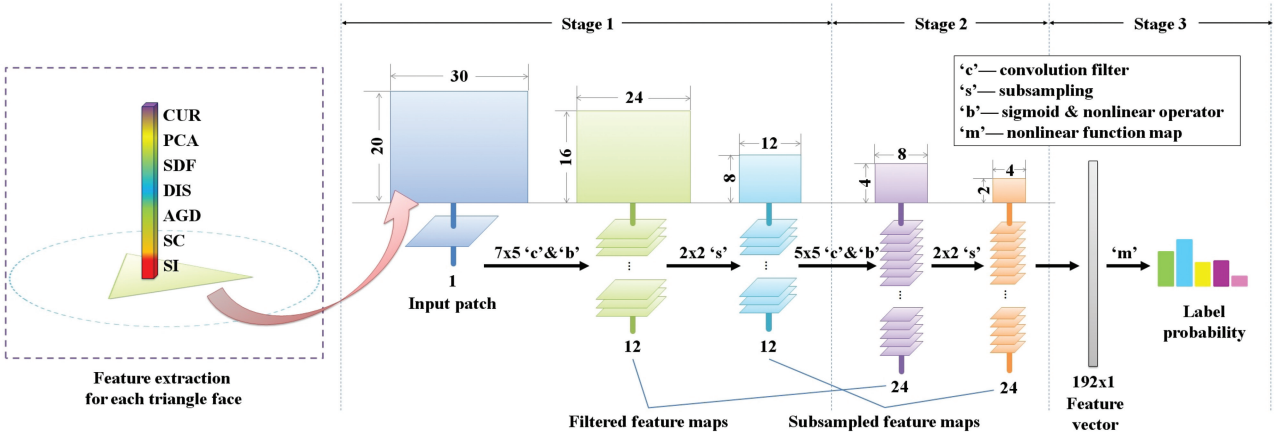


Fig. 1. The architecture of the trained CNNs with three stages. Each of the first two stages contains a filter bank module, a nonlinearity module, and a spatial pooling module for subsampling. The last stage involves a nonlinear function map which is used to obtain the label probability of each triangle on the mesh.

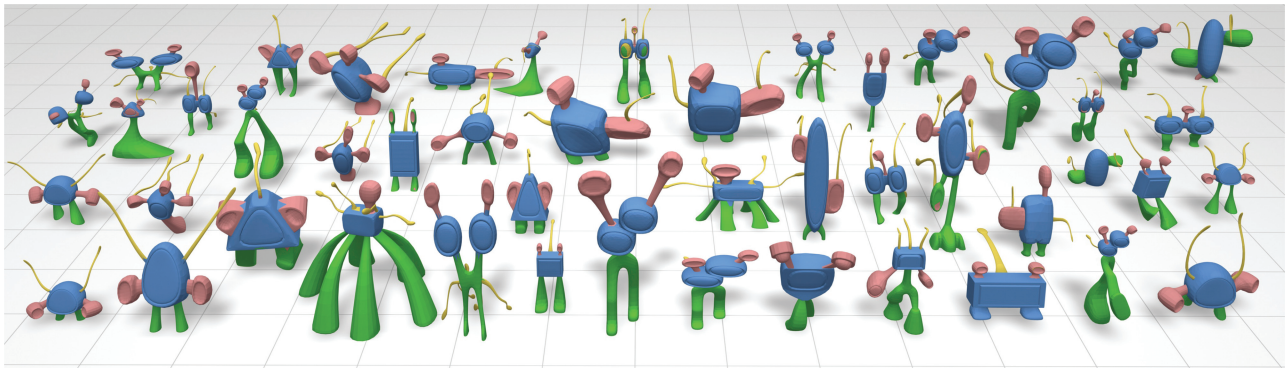


Fig. 2. Labeling results predicted by the deep CNNs that are trained on randomly selected meshes in the same class and tested on the new meshes.

characteristics of CNNs, we reorganize these features into a 2D feature matrix and treat it as the input to the CNNs. Given the features and labels of massive triangles, we can thus train CNNs so as to obtain a compact, effective, and robust mesh representation. Meanwhile, the trained CNNs also generate a label vector for each triangle that describes its probabilities of belonging to various object parts. In this way, the trained CNNs serve as a bridge with a deep structure that links the pool of geometric features and mesh labels. Experimental results show that our approach is effective and robust in labeling 3D meshes (as shown in Figure 2).

The main contributions of our article include: (1) A novel mesh representation is learned by nonlinearly combining and hierarchically compressing various geometry features with the deep CNNs. Compared with the heuristically designed geometric features, the compact mesh representation learned from data can adapt to various 3D models in mesh labeling. (2) Extensive experiments are conducted on public benchmarks. Experimental results show that our approach outperforms state-of-the-arts in many categories.

The rest of this article is organized as follows: Section 2 reviews related work and Section 3 introduces how to train deep CNNs for mesh labeling. In Section 4, we visualize the learned mesh representations. Then we demonstrate impressive performance in the experiments in Section 5. Finally, the article is concluded in Section 6.

2. RELATED WORK

Recently, mesh segmentation and labeling have attracted much research interest in the field of computer graphics. In some early studies such as Katz and Tal [2003], Ben-Chen and Gotsman [2008], Huang et al. [2009], Shapira et al. [2010], Zhang et al. [2012], and Au et al. [2012], a major concern is to find a single feature that is effective for mesh label identification. For example, Curvature features, PCA features, Shape diameter and Distance from medial surface. However, the meshes in different 3D models may vary remarkably. A single feature is often insufficient to process all kinds of scenarios.

To address this problem, several cosegmentation approaches [Huang et al. 2011; Sidi et al. 2011; Hu et al. 2012; van Kaick et al. 2013; Kim et al. 2013] were proposed, which aimed to extract common geometric subsets and their correspondence from each category of 3D shapes. These methods were unsupervised and their usage were restricted by the knowledge presented in unlabeled data. On the contrary, supervised or semisupervised approaches [Kalogerakis et al. 2010; Lv et al. 2012; Wang et al. 2012, 2013; Xie et al. 2014] were also proposed to tackle this problem. For example, Kalogerakis et al. [2010] presented a method to segment and label 3D meshes by combining various geometric features with the Conditional Random Field [Lafferty et al. 2001] and JointBoost

classifier [Torralba et al. 2007]. Wang et al. [2013] first projected 3D shapes to 2D space and the labeling results in 2D space were then projected back to 3D shapes for mesh labeling. In this process, bi-class symmetric Hausdorff (BiSH) distance was used for 3D mesh labeling. Recently, Xie et al. [2014] proposed a supervised approach for fast 3D shape segmentation and labeling. They employed the Extreme Learning Machine to train a neural networks as the mesh label classifier. In many cases, these learning-based approaches outperform the heuristic approaches that utilize only simple geometric features. However, the simple combination of geometric features (e.g., linear model and shallow neural network) are often insufficient to be generalized to complex meshes.

As discussed previously, one of the key issues in mesh labeling is to find *effective and robust mesh representation*, and a feasible solution is to learn such representation from various geometric features that have been proved to be successful in certain mesh labeling tasks. Recent studies show that deep neural networks have impressive capabilities in extracting effective and biologically plausible representations from low-level features [Bengio 2009; Hinton 2010]. As a special case, CNNs [Krizhevsky et al. 2012; Farabet et al. 2013] have been proved to be effective for nonlinear combination and compression of raw features and can apply to various applications [Socher et al. 2012; Bruna et al. 2014]. Toward this end, we propose to "learn" the mesh representation from a large pool of classical geometric features with deep CNNs so as to address the problem of mesh labeling.

3. MESH LABELING VIA CNNS

In this section, we will present how to learn compact and effective mesh representations by using deep CNNs. First, we extract a large pool of geometric features to form a 2D feature matrix so as to characterize each triangle on the mesh. Second, we present the architecture of the deep CNNs that are used to learn mesh representations from massive feature matrices. Third, we show the details of training the CNNs. Finally, we make a brief description of the mesh label optimizing process.

3.1 Geometric Feature Extraction

In our approach, we aim to learn a compact and effective mesh representation from low-level features. Thus, we first extract seven types of geometric features that are widely used in existing studies, including: curvature (CUR) [Gal and Cohen-Or 2006], PCA feature (PCA) [Kalogerakis et al. 2010], shape diameter function (SDF) [Shapira et al. 2010], distance from medial surface (DIS) [Liu et al. 2009], average geodesic distance (AGD) [Hilaga et al. 2001], shape context (SC) [Belongie et al. 2002], and spin image (SI) [Johnson and Hebert 1999]. While the features are calculated with face areas and different scales in consideration. These features can well describe the characteristics of each triangle on a mesh from multiple perspectives.

Given these features, an intuitive solution is to concatenate them into a high-dimensional feature vector (i.e., 600 components in total). However, as suggested in Hu et al. [2012] and Wang et al. [2012], such simple concatenation will degrade the performance of clustering and may lead to over-fitting due to the high-dimensional descriptor space. To address this problem and to fully utilize the convolutional property of CNNs, we reorganize these 600 components to form a 30×20 feature matrix (as shown in Figure 1). In this manner, low-level geometric features can be nonlinearly combined and hierarchically compressed through various convolutional operations in CNNs. In experiments, we will demonstrate that the ordering of features as well as the size of feature matrix only slightly change the accuracy of mesh labeling.

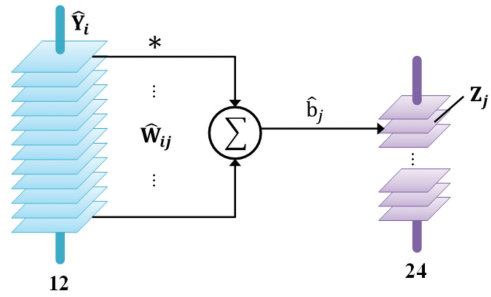


Fig. 3. In the second stage, each output feature map is obtained by applying twelve 5×5 convolution kernels on all first-stage feature maps.

3.2 Networks Structure

The feature matrix of each face, denoted as $\mathbf{X}$, can be used as the input of CNNs. As shown in Figure 1, the structure of CNNs consists of three major stages. In the first stage, twelve 7×5 convolution kernels $\{\mathbf{W}_i\}_{i=1}^{12}$ and bias values $\{b_i\}_{i=1}^{12}$ are applied to the input feature matrix as:

$$\mathbf{Y}_i = \mathbf{W}_i * \mathbf{X} + b_i,\ i = 1, \ldots, 12, \tag{1}$$

where $*$ indicates the convolution operation and the same bias b_i is added to all the components of $\mathbf{W}_i * \mathbf{X}$. After the convolution, twelve 24×16 maps are generated by passing $\{\mathbf{Y}_i\}_{i=1}^{12}$ through a nonlinearity module with a sigmoid activation function (i.e., $\phi(x) = (1 + \exp(-x))^{-1}$). Note that the nonlinearity is operated on each element of $\mathbf{Y}_i$. After the nonlinear mapping, we down-sample each dimension of the maps by a factor of 2 to generate twelve 12×8 feature maps, denoted as $\{\widehat{\mathbf{Y}}_i\}_{i=1}^{12}$.

In the second stage, we first generate 24 new feature maps from the 12 input feature maps (as shown in Figure 3). In this process, each 8×4 map $\mathbf{Z}_j$ is obtained by applying 12 different 5×5 kernels $\{\widehat{\mathbf{W}}_{ij}\}_{i=1}^{12}$ and biases $\widehat{b}_j$ to the input feature maps $\{\widehat{\mathbf{Y}}_i\}_{i=1}^{12}$:

$$\mathbf{Z}_j = \sum_{i=1}^{12} \widehat{\mathbf{W}}_{ij} * \widehat{\mathbf{Y}}_i + \widehat{b}_j,\ j = 1, \ldots, 24. \tag{2}$$

After that, the same nonlinearity module and subsampling operation is conducted, which eventually produce 24 down-sampled 4×2 feature maps. For the sake of simplification, we reorganize these feature maps into a compact descriptor $\mathbf{v}$ which is a column vector with 192 components.

In the third stage, we aim to infer the label of a triangle from the descriptor $\mathbf{v}$. Therefore, we assume that there exist a matrix $\mathbf{M}$ and bias vector $\mathbf{b}$ that can map the descriptor $\mathbf{v}$ to label indicator $\mathbf{p}$:

$$\mathbf{p} = \phi\left(\mathbf{M}\mathbf{v} + \mathbf{b}\right). \tag{3}$$

Suppose there exist N_c candidate labels for each triangle, the size of matrix $\mathbf{M}$ becomes $N_c \times 192$. Both $\mathbf{b}$ and $\mathbf{p}$ have N_c components, and each component of $\mathbf{p}$ falls in the range $[0, 1]$. In this manner, the deep CNNs can be viewed as a multiclass classifier. Suppose that the ground-truth label of the triangle is denoted as a vector $\mathbf{g}$ with 1 at only one component and 0 elsewhere, the objective of network training is to optimize all parameters (i.e., convolution kernels, biases, mapping matrix, etc.) to approximate $\mathbf{g}$ with $\mathbf{p}$.

From the extraction of $\mathbf{v}$ and the computation of $\mathbf{p}$, we can see that each component of $\mathbf{v}$ is actually generated by nonlinearly combining

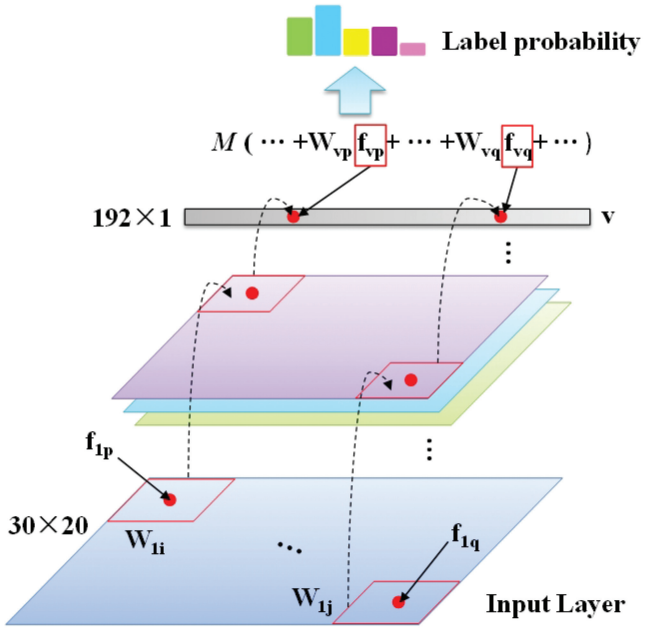


Fig. 4. The procedure of feature weights learning. For any two features, our method hierarchically combines them together by learning a set of weights in different layers of CNNs. Specially, for any two features (i.e., f_{1p}, f_{1q}) in the input layer, there are two or more corresponding high features (i.e., f_{vp}, f_{vq}) in the feature vector layer. These high features are further fused with a set of weights, which in fact equals to fuse the original features by learning a set of weights.

(i.e., sigmoid function) and hierarchically compressing (i.e., down-sampling) a subset of the input features. Thus each feature actually characterizes specific attributes of a triangle on the mesh. As shown in Figure 4, various attributes of the triangle are combined to infer the final label of a triangle. In this process, any two components from the input feature matrix can be nonlinearly combined, no matter how the low-level geometric features are organized in the feature matrix. For the sake of simplification, we rewrite the feature $\mathbf{v}$ as a function of $\mathbf{X}$:

$$\mathbf{v} = \mathcal{N}\left(\mathbf{X} \,\middle|\, \left\{\mathbf{W}_i, \widehat{\mathbf{W}}_{ij}, b_i, \widehat{b}_j\right\}_{i,j}\right). \tag{4}$$

3.3 Networks Training

Given the structure of the deep networks, we have to optimize its parameters for mesh labeling. Suppose we have T triangles and their ground-truth labels, we can train these parameters by minimizing the following energy function:

$$\begin{aligned} \Pi^* &= \arg\min_{\Pi} \sum_{t\in T} \left|\mathbf{g}_t - \phi\left(\mathbf{M}\mathbf{v}_t + \mathbf{b}\right)\right|^2, \\ \text{where } \Pi &= \left\{\left\{\mathbf{W}_i, \widehat{\mathbf{W}}_{ij}, b_i, \widehat{b}_j\right\}_{i,j}, \mathbf{M}, \mathbf{b}\right\}. \end{aligned} \tag{5}$$

In our implementation, the minimization of (5) mainly consists of iterative feed-forward and back-propagation operations. Note that we initialize all convolution kernels (i.e., $\mathbf{W}_i$ and $\widehat{\mathbf{W}}_{ij}$) and $\mathbf{M}$ with small random values and set $b_i, \widehat{b}_j, \mathbf{b}$ to 0.

Feed-forward. In the feed-forward process, we pass the input geometric features through the deep CNNs to update the label indicator $\mathbf{p}_t$. As shown in (5), the prediction error is computed as the $\ell 2$ distance between $\mathbf{g}_t$ and $\mathbf{p}_t$.

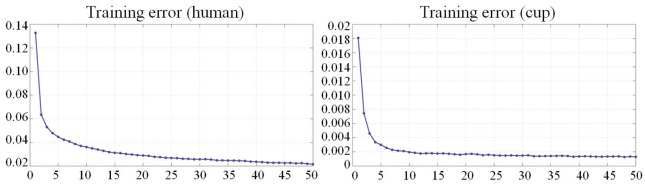


Fig. 5. Training error convergence curves. One iteration error is obtained by averaging the errors of all the training triangular faces.

Back-propagation. In the back-propagation process, the objective is to propagate backward the prediction error and reduce it by tuning the parameters layer by layer. Here we use the gradient descent algorithm to update the parameters layer by layer:

$$\pi \leftarrow \pi - \alpha \cdot \nabla_\pi \sum_{t\in T} \left|\mathbf{g}_t - \phi\left(\mathbf{M}\mathbf{v}_t + \mathbf{b}\right)\right|^2, \quad \forall \pi \in \Pi, \tag{6}$$

where ∇ represents the gradient and α is the learning rate (we empirically set $\alpha = 0.95$). The feed-forward and back-propagation processes are iteratively conducted until convergence or a predefined number of iterations is reached (e.g., 50). Figure 5 shows two examples to illuminate the convergency of our method. Our method can achieve convergency through a few of iterations. In general, for complex meshes (e.g., human), the convergence rate is slower than the simple ones (e.g., cup), as shown in Figure 5.

3.4 Mesh Label Optimization

Given the trained CNNs, we can easily obtain a label indicator for each triangle on a testing mesh. Intuitively, the indicator can roughly depict the probabilities that a triangle belongs to various parts of the mesh. However, inconsistency may arise among adjacent triangles, leading to unsatisfactory visual effects. To address this problem, we employ the multilabel graph-cut method [Boykov et al. 2001] to refine the labels by incorporating the constraint on label consistency. Let $\mathbb{T}$ be the set of triangles on a mesh, we use t to represent a triangle with label indicator $\mathbf{p}_t$ and $\mathbb{N}_t$ be its neighboring triangles. Thus, the labels $\{l_t | t \in \mathbb{T}\}$ of these triangles can be optimized by solving:

$$\min_{\{l_t, t\in\mathbb{T}\}} \sum_{t\in\mathbb{T}} \xi_U(\mathbf{p}_t, l_t) + \lambda \sum_{t\in\mathbb{T}, v\in\mathbb{N}_t} \xi_S(\mathbf{p}_t, \mathbf{p}_v, l_t, l_v), \tag{7}$$

where λ is a nonnegative constant to balance the influences of the terms (we empirically set $\lambda = 50$). The first term is defined as:

$$\xi_U(\mathbf{p}_t, l_t) = -\log(\mathbf{p}_t(l_t)), \tag{8}$$

where $\mathbf{p}_t(l_t)$ denotes the l_tth component of $\mathbf{p}_t$. This implies that assigning a label to triangle t that corresponds to a small value in $\mathbf{P}_t$ will lead to a large penalty.

Moreover, the second term penalizes the smoothness between the labels of adjacent triangles:

$$\xi_S(\mathbf{p}_t, \mathbf{p}_v, l_t, l_v) = \begin{cases} 0, & \text{if } l_t = l_v \\ -log(\theta_{tv}/\pi)\varphi_{tv}, & \text{otherwise} \end{cases}. \tag{9}$$

where φ_{tv} and θ_{tv} are the distance and dihedral angle between the triangles t and v, respectively. With the smooth term, two adjacent triangles are likely to have consistent labels. By incorporating (8) and (9) into (7), we can efficiently refine the mesh labels using the graph cuts algorithm as in Boykov et al. [2001].

Fig. 6. Visualization of the normalized feature distance between each triangle and the human part it belongs to. The features used in the computation are: 1st row: geometric features $\mathbf{X}$ (input of the first stage); 2nd row: feature $\widehat{\mathbf{Y}}_i$ (output of the first stage and also the input of the second stage); 3rd row: feature $\mathbf{v}$ (output of the second stage and also the input of the third stage); 4th row: label indicator $\mathbf{p}$ (output of the third stage).

4. VISUALIZATION AND ANALYSIS

After training the CNNs, we try to visualize the learned features to provide an intuitive impression on their effectiveness. Given a 3D human model with multiple parts, we assume that a triangle from the model can be characterized by the features obtained after each stage of CNNs (i.e., $\mathbf{X}$, $\widehat{\mathbf{Y}}_i$, $\mathbf{v}$, and $\mathbf{p}$). Thus, the Euclidean distance (normalized to $[0, 1]$) between the feature of a triangle and the average feature of the model part it belongs to can provide an intuitive impression on the effect of the learned features. As

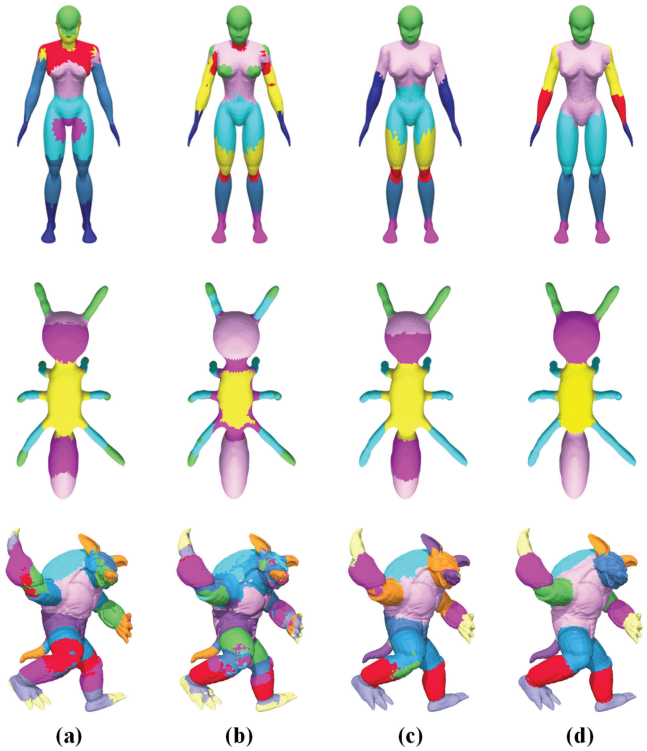


Fig. 7. Column (a)-(d) visualize the K-means results of (a) geometric features $\mathbf{X}$ (input of the first stage); (b) features $\{\widehat{\mathbf{Y}}_i\}$ (output of the first stage and also the input of the second stage); (c) features $\mathbf{v}$ (output of the second stage and also the input of the third stage); (d) label indicator $\mathbf{p}$ (output of the third stage).

shown in Figure 6, the learned features gradually becomes more similar after various stages of CNNs. This implies that the features generated by CNNs can be effective for mesh labeling.

Another way of visualization is to cluster all the triangles by using the K-means algorithm. As shown in Figure 7, the clustering results gradually become better by using the features obtained from later stages of CNNs. Furthermore, we can see that the trained CNNs can even distinguish triangles from object parts with similar geometric features but different labels (e.g., the antennas and legs of the ant in Figure 7). Moreover, for the parts with the same labels but different geometric features, the trained CNNs can also correctly identify them (e.g., the hands of the armadillo in Figure 7).

In addition to "Human," "Ant," and "Armadillo," we also show the visualization of several other categories using the feature vectors mapped in the stage 3 of CNNs, as shown in Figure 8. Even though without using the last mapping and smooth processes, the consistent and comparable results show the effectiveness of the deep structures. Such qualitative analysis and visualization demonstrate that the learned features are effective for mesh labeling.

5. EXPERIMENTS

In this section, we conduct extensive experiments to show the effectiveness of our approach. These experiments are conducted on the Princeton Segmentation Benchmark (PSB) [Chen et al. 2009] (19 categories, 20 meshes per category) and 4 additional categories including Lamp (20 meshes), Candelabra (28 meshes), Iron (18 meshes) and Guitar (44 meshes) from van Kaick et al. [2011],

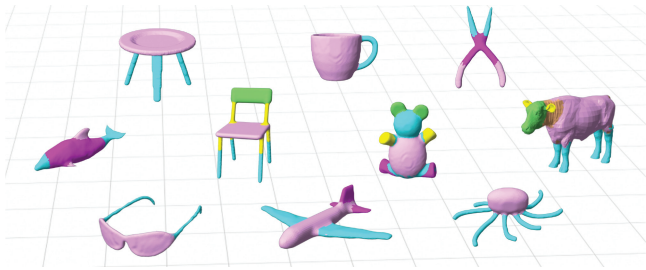


Fig. 8. More results of several other categories using the original feature vectors directly with the mapping in stage 3.

Sidi et al. [2011], and Wang et al. [2012]. We also test our method on two large categories, namely Chair (400 meshes) and Vase (300 meshes), to further evaluate the proposed method. For each category, N meshes are randomly selected for training and the rest meshes are used for testing. In our experiments, we set $N = 6$, $N = 12$, and $N = 19$ (i.e., SB6, SB12, and SB19) for categories from PSB and 4 small additional categories. In particular, we set $N = 50$ and $N = 20$ for two large categories. On the PSB and four additional categories, we compare our method to three mesh labeling models [Kalogerakis et al. 2010; Wang et al. 2013; Xie et al. 2014] and classic learning methods (i.e., SVM [Chang and Lin 2011] and JointBoost [Torralba et al. 2007]). On each category, our approach is repeated five times to compute the average performance. And for SB19 especially, we use leave-one-out cross validation.

As in Kalogerakis et al. [2010], the accuracy is computed as:

$$Accuracy = \sum_{t\in\mathbb{T}} a_t \mathbf{g}_t(l_t) / \sum_{t\in\mathbb{T}} a_t, \tag{10}$$

where a_t is the area of the triangle t and l_t is the predicted label. $\mathbf{g}_t(l_t)$ denotes the l_tth component of $\mathbf{g}_t$, which equals to 1 if the prediction is correct.

The experiments on accuracy comparisons are shown in Table I and Table II and some visual comparisons are shown in Figure 9. We can see that our approach outperforms Kalogerakis et al. [2010] and Wang et al. [2013] on most of the object categories. On average, our method obtains 91.34%, 93.39%, 94.96% accuracy according to "SB6," "SB12," and "SB19", respectively. As shown in Table I, when six meshes are used for training, our approach outperforms [Kalogerakis et al. 2010] on 17 categories. For the remaining two categories, we get the same accuracy on "Ant" and comparable accuracy on "Human". Our method also outperforms Wang et al. [2013] on five categories out of the seven categories ever reported, especially on articulated objects. When the number of training meshes increases to 12, our approach outperforms Kalogerakis et al. [2010] on 17 categories and Wang et al. [2013] on 4 categories. In Table II, we can see that our approach outperforms Kalogerakis et al. [2010] and Xie et al. [2014] on all categories in the SB19 experiment, and outperforms Wang et al. [2013] on five categories. Generally speaking, the performance of our method becomes better with more training meshes. Figure 10 shows variances and standard deviations of our repeated tests. From the figure, we can see that the variances and standard deviations of most meshes are small, which demonstrates the robustness of our proposed method.

Comparison with classical learning methods. We also compare our method to SVM [Chang and Lin 2011] and JointBoost [Torralba et al. 2007]. We first fine-tune the parameters of these two methods for label classification, followed by the same graph-based label optimization process as in our approach. As shown in Table I, our method performs significantly better than these two methods. The

Table I. Labeling Accuracy on Princeton Benchmark and Four Additional Categories (SB6/SB12)

	SVM SB6	JointBoost SB6	ToG[2010] SB6	ToG[2013] SB6	Ours SB6	Ours 2 features	ToG[2010] SB12	ToG[2013] SB12	Ours SB12
Human	48.43%	70.35%	**89.40%**	51.10%	88.05%	86.26%	**93.20%**	55.60%	91.22%
Cup	94.11%	93.12%	99.10%	97.50%	**99.49%**	98.94%	99.60%	99.60%	**99.73%**
Glasses	95.92%	93.59%	96.10%	-	**96.78%**	96.42%	97.20%	-	**97.60%**
Airplane	80.43%	91.16%	95.50%	-	**95.56%**	94.27%	96.10%	-	**96.67%**
Ant	90.38%	92.81%	98.70%	-	98.70%	97.88%	98.80%	-	98.80%
Chair	81.38%	95.67%	97.80%	97.90%	**97.93%**	96.10%	98.40%	**99.60%**	98.67%
Octopus	97.04%	96.26%	98.60%	-	**98.61%**	98.12%	98.40%	-	**98.79%**
Table	90.16%	97.37%	99.10%	**99.60%**	99.11%	98.28%	99.30%	**99.60%**	99.55%
Teddy	91.01%	85.68%	93.30%	-	**98.00%**	96.82%	98.10%	-	**98.24%**
Hand	64.10%	68.55%	82.40%	-	**83.98%**	83.25%	88.70%	-	**88.71%**
Plier	92.04%	81.55%	94.30%	-	**95.01%**	93.51%	96.20%	-	**96.22%**
Fish	87.05%	90.76%	95.60%	-	**96.22%**	94.77%	95.60%	-	**95.64%**
Bird	81.49%	81.80%	84.20%	-	**87.51%**	87.17%	87.90%	-	**88.35%**
Armadillo	54.28%	72.06%	84.00%	-	**90.00%**	87.93%	90.10%	-	**92.27%**
Bust	51.88%	51.99%	53.90%	-	**63.39%**	61.60%	62.10%	-	**69.84%**
Mech	81.87%	75.73%	88.90%	90.20%	**90.56%**	89.52%	90.50%	91.30%	**95.60%**
Bearing	67.16%	77.10%	84.80%	-	**90.08%**	87.13%	86.60%	-	**92.46%**
Vase	74.19%	75.81%	77.00%	**89.70%**	80.74%	73.85%	85.80%	**90.50%**	89.11%
FourLeg	71.90%	75.94%	85.00%	59.10%	**85.81%**	82.10%	86.20%	54.30%	**87.02%**
Average	78.67%	82.49%	89.35%	83.59%	**91.34%**	89.68%	92.04%	84.36%	**93.39%**
Iron	83.96%	89.81%	-	-	91.85%	82.84%	-	-	97.37%
Guitar	87.57%	92.69%	-	-	96.95%	96.22%	-	-	97.15%
Lamp	59.26%	44.16%	-	-	91.14%	90.08%	-	-	96.28%
Candelabra	83.91%	71.24%	-	-	94.28%	91.21%	-	-	94.47%

ToG[2010] is short for Kalogerakis et al. [2010], and ToG[2013] is short for Wang et al. [2013].

Table II. Labeling Accuracy on Princeton Benchmark (SB19)

	ToG[2010]	ToG[2013]	CGF[2014]	Ours
Human	93.60%	63.80%	88.61%	**94.39%**
Cup	99.60%	99.10%	98.32%	**99.79%**
Glasses	97.40%	-	96.79%	**98.01%**
Airplane	96.30%	-	94.49%	**97.74%**
Ant	98.80%	-	95.98%	**98.84%**
Chair	98.50%	**99.20%**	97.41%	98.53%
Octopus	98.40%	-	98.73%	**98.87%**
Table	99.40%	99.60%	99.31%	**99.63%**
Teddy	98.10%	-	97.74%	**98.53%**
Hand	90.50%	-	83.47%	**92.21%**
Plier	97.00%	-	95.96%	**97.01%**
Fish	96.70%	-	95.68%	**96.74%**
Bird	92.50%	-	95.68%	**96.12%**
Armadillo	91.90%	-	87.86%	**93.57%**
Bust	67.20%	-	61.36%	**70.67%**
Mech	94.60%	94.60%	87.49%	**95.90%**
Bearing	95.20%	-	90.84%	**96.05%**
Vase	87.20%	**91.90%**	89.52%	90.11%
FourLeg	88.70%	67.90%	91.44%	**91.44%**
Average	93.77%	88.01%	91.93%	**94.96%**

ToG[2010] is short for Kalogerakis et al. [2010], ToG[2013] is short for Wang et al. [2013], and CGF[2014] is short for Xie et al. [2014].

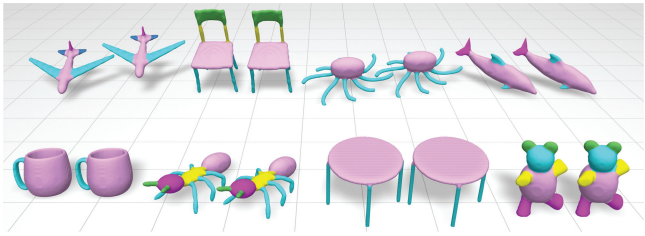


Fig. 9. Comparisons between the results with all geometric features and those with two selected geometric features. For each group, the left is the labeling results with all features, and the right is the one with two features.

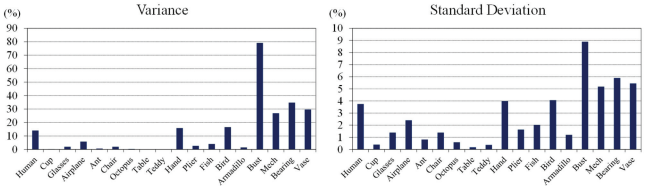


Fig. 10. The variances and standard deviations of the repeated tests for Princeton Benchmark (SB19).

main reason is that the features learned in CNNs can better represent a mesh, which is very important if the training instances are very limited. In this case, the training instances are often too sparse to train the SVM and the boosting classifier. To clearly show the superiority of our method and compare the features learnt by CNNs and those learnt by JointBoost, we visualize the label results before graph-cut smooth, as shown in Figure 11. As a linear classifier, JointBoost can be treated as a shallow network (one layer). When the training process involves large numbers of meshes, containing tens of thousands of triangles, the feature space would become large and complex. Linear dividing may fail in many cases. As shown in Figure 11, for meshes with large geometry variations (i.e.,

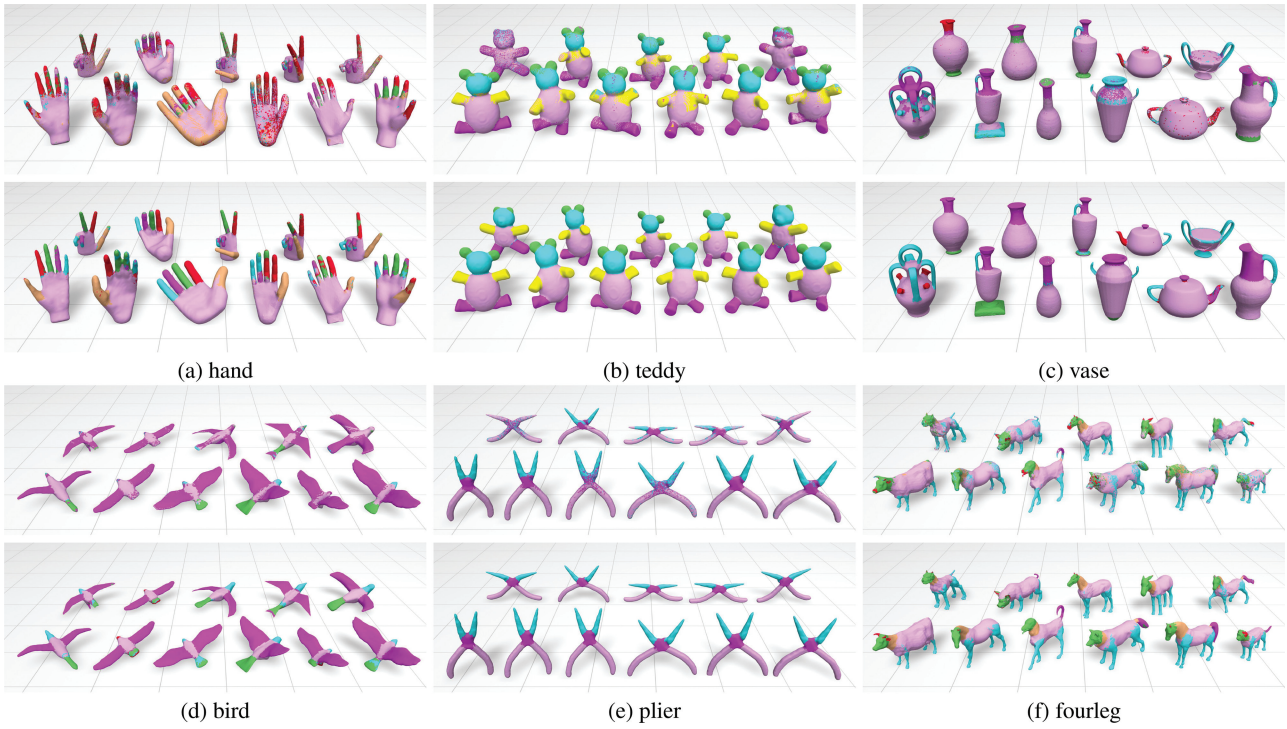


Fig. 11. Visual comparison between JointBoost results and our method before smooth. The second rows show the results produced by our method, and the first rows show the results by JointBoost [Torralba et al. 2007].

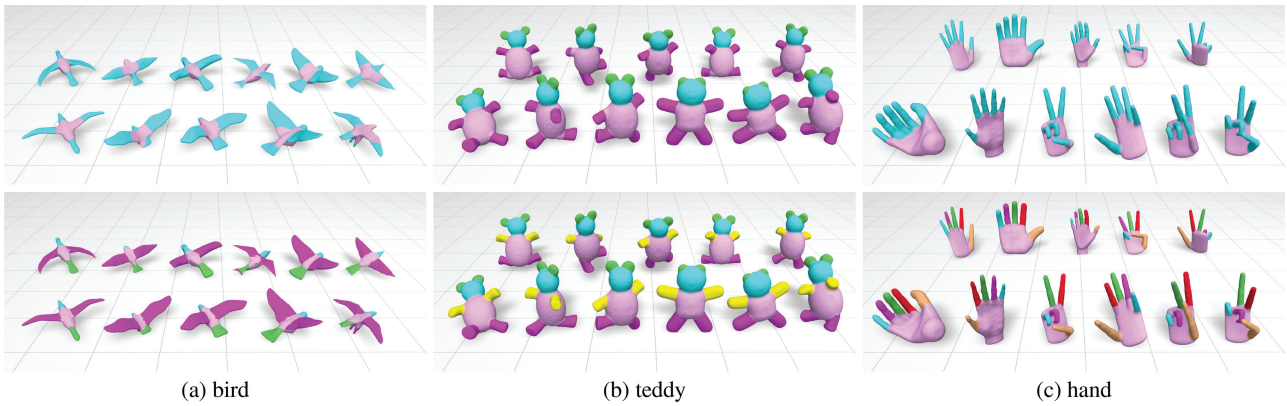


Fig. 12. Comparisons with mesh cosegmentation. The second row shows the results produced by our method, and the first row shows the results from Hu et al. [2012].

hand and vase), JointBoost method could not ensure the correctness and consistency. In contrast, our method can well represent these meshes. Through multilayer convolutions and nonlinear mapping, the feature space becomes much easier to be divided.

Comparison with the cosegmentation method. We also make a comparison with Hu et al. [2012], which is a cosegmentation method on 3D meshes. As shown in Figure 12, our method can get better viewing effects than that of Hu et al. [2012], while the meshes can be labeled at a finer level in our approach. Moreover, our method could achieve more accurate segment boundaries, such as the wings of the birds, the four limbs of the bears, and the fingers of the hands shown in Figure 12.

Performance with less raw features. To illustrate the robustness of our method, we conduct an additional experiment to test the performance of our approach when fewer raw features are used. We remove five raw features from the feature pool and train/test the CNNs again on the PSB. In SB6, the performance of our approach still outperforms SVM and JointBoost and becomes comparable to Kalogerakis et al. [2010] with only two raw features (SC and DIS). As shown in the seventh column of Table I, the performance of our method is even comparable to the results achieved by using all features (see Figure 9 for some examples). Additionally, to show the influence of the labeling correctness by different low-level features, we remove one raw feature in turn and train/test the CNNs.

Table III. Experiments of Different Raw Features for Networks

	RCUR	RPCA	RSC	RAGD	RSDF	RDIS	RSI
Human	87.36%	87.37%	86.02%	88.07%	86.82%	84.00%	87.42%
Cup	97.54%	98.74%	99.29%	99.13%	99.48%	99.13%	99.43%
Glasses	95.90%	96.49%	96.05%	96.03%	96.27%	96.04%	96.03%
Airplane	94.19%	95.30%	92.80%	95.05%	95.31%	94.19%	95.46%
Ant	97.22%	97.84%	96.79%	98.36%	98.32%	98.38%	98.25%
Chair	95.01%	96.91%	94.72%	95.93%	96.61%	95.00%	96.13%
Octopus	98.67%	98.64%	98.48%	98.64%	98.55%	98.59%	98.63%
Table	98.38%	99.20%	95.45%	99.45%	99.30%	99.02%	99.28%
Teddy	97.62%	98.04%	97.92%	97.31%	97.77%	97.57%	98.00%
Hand	80.95%	80.54%	80.67%	83.67%	82.92%	81.05%	80.57%
Plier	93.08%	94.24%	93.85%	94.84%	93.22%	94.87%	93.39%
Fish	95.45%	95.99%	95.45%	95.70%	95.50%	95.50%	95.52%
Bird	88.90%	88.53%	88.14%	88.20%	89.50%	87.68%	88.49%
Armad.	87.75%	89.99%	87.80%	90.23%	89.25%	89.34%	88.23%
Bust	55.13%	59.07%	53.70%	57.16%	63.22%	60.79%	62.43%
Mech	88.08%	89.17%	87.65%	88.24%	88.77%	88.80%	88.35%
Bearing	79.33%	87.12%	74.00%	79.84%	83.67%	85.28%	89.49%
Vase	77.26%	78.71%	81.27%	81.67%	80.27%	80.54%	81.54%
FourLeg	83.10%	85.30%	83.62%	83.10%	82.02%	84.02%	84.73%
Average	88.99%	90.38%	88.61%	90.03%	90.36%	89.99%	90.60%

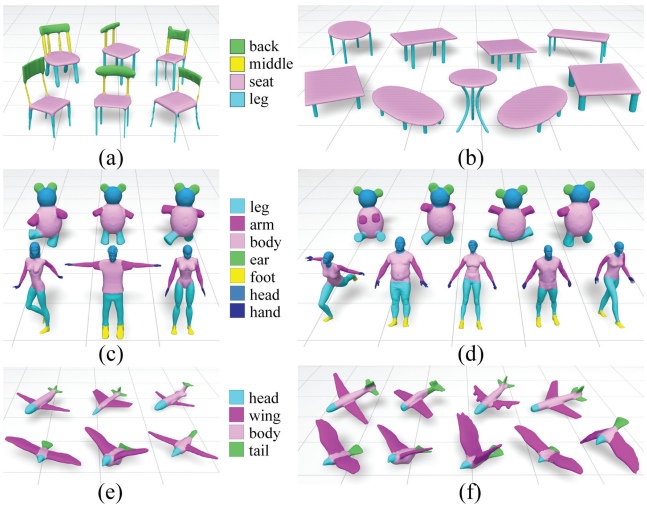


Fig. 13. Labeling results for crossing and merging categories. The left ones are used for training and the right ones for testing.

As shown in Table III, the accuracy of each category fluctuates differently. Generally speaking, the performance of less raw features is comparable to the results by all features. This proves that our method is robust and effective when it is trained on various geometric features.

Reshape input features into different sizes and orders. Another concern that may arise is the size and order of 2D feature matrix. To show that our approach is robust when using various sizes of feature matrices, we reorganize the 30×20 feature matrix into 20×30, 40×15, 15×40, 60×10, and 10×60 to test the performance of our approach. The labeling accuracies on 19 categories in SB6 are shown in Table IV. Moreover, to show that our approach is robust to various orders of feature matrices, we randomly rearrange the 30×20 feature matrix and repeat this operation by ten times, obtaining 10 new feature matrices to test the performance of our approach. The variation of labeling accuracies on 19 categories in SB6 varies

Table IV. Experiments of Different Reshaping Size for Networks

	30×20	20×30	40×15	15×40	60×10	10×60
Human	88.05%	87.67%	87.56%	87.67%	88.18%	88.24%
Cup	99.49%	99.25%	99.58%	99.59%	99.37%	99.60%
Glasses	96.78%	96.73%	96.82%	96.78%	96.82%	96.76%
Airplane	95.56%	95.67%	95.04%	95.02%	95.33%	95.02%
Ant	98.70%	98.50%	98.38%	98.32%	98.24%	98.37%
Chair	97.93%	97.81%	98.03%	97.99%	98.07%	97.63%
Octopus	98.61%	98.56%	98.74%	98.72%	98.62%	98.70%
Table	99.11%	99.12%	98.75%	98.81%	98.92%	98.85%
Teddy	98.00%	98.00%	97.87%	97.89%	97.88%	98.11%
Hand	83.98%	83.86%	84.02%	83.97%	83.75%	83.67%
Plier	95.01%	95.12%	95.13%	94.85%	95.10%	95.11%
Fish	96.22%	96.20%	96.24%	96.10%	96.08%	96.36%
Bird	87.51%	87.62%	87.14%	87.13%	87.17%	87.58%
Armadillo	90.00%	90.11%	89.83%	89.80%	89.90%	89.83%
Bust	63.39%	63.02%	63.23%	63.08%	63.25%	63.13%
Mech	90.56%	90.13%	90.29%	90.23%	90.32%	90.29%
Bearing	90.08%	89.94%	90.19%	90.21%	90.21%	89.95%
Vase	80.74%	80.44%	80.61%	80.62%	80.33%	80.41%
FourLeg	85.81%	85.52%	85.38%	85.48%	85.61%	85.56%
Average	91.34%	91.22%	91.20%	91.17%	91.22%	91.22%

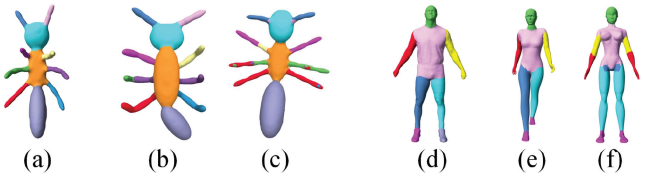


Fig. 14. Experiments for distinguishing similar parts within the models. (a) and (d) are the training samples, and (b), (c), (e), and (f) are the test ones.

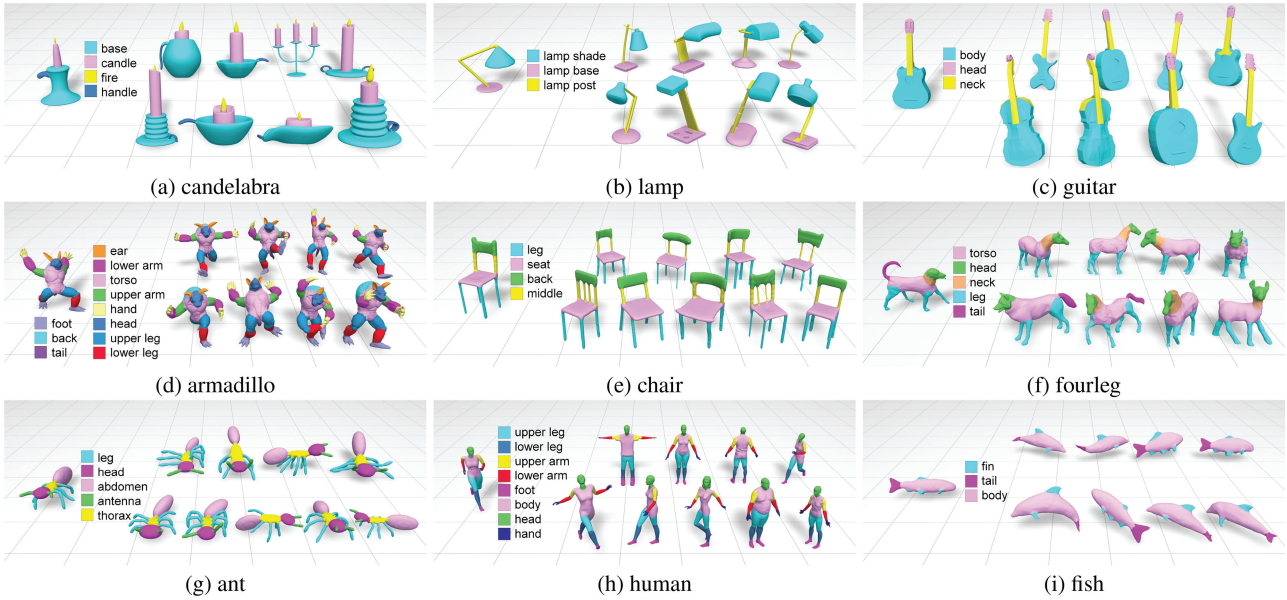
from 1.0407×10^{-6}(Octopus) to 7.4515×10^{-4}(FourLeg). We can see that the accuracies only slightly changes even when the sizes and orders of feature matrices change remarkably.

Time cost. Our implementation runs on a single thread on Xeon E5-2670 2.60GHz CPU. It takes about 4 hours to train the model with 20K-30K faces with six labels and 2 minutes testing a mesh. In comparison, Kalogerakis et al. [2010] takes 8 hours for training on Xeon E5355 2.66GHz CPU.

More results. The results when the deep CNNs are trained on one category and applied to other categories are shown in Figures 13(a) and (b). We also conduct an experiment to train the deep CNNs on a training set consisting of blended categories, such as humans and teddies, birds and planes. When the trained CNNs are applied to other meshes, the label can be also correctly identified (as shown in Figures 13(c) and (d), and (e) and (f)). These impressive results prove that our approach has impressive generalization ability.

Our method is able to distinguish those similar parts from each other. As shown in Figure 14, two antennas and six legs of an ant model are labeled individually as in Figure 14(a), similarly the arms, legs and feet of human model as in Figure 14(d). However, the results are not desirable when the test meshes are absolutely symmetrical. It is because that the geometric features of those symmetric parts are almost the same.

More labeling results produced by our method are shown in Figure 15. We also compare the proposed approach with various other methods on challenging datasets [Wang et al. 2012], as shown in Figure 16. And from Table V, we can see that the proposed



(a) candelabra (b) lamp (c) guitar

(d) armadillo (e) chair (f) fourleg

(g) ant (h) human (i) fish

Fig. 15. More results of our method.

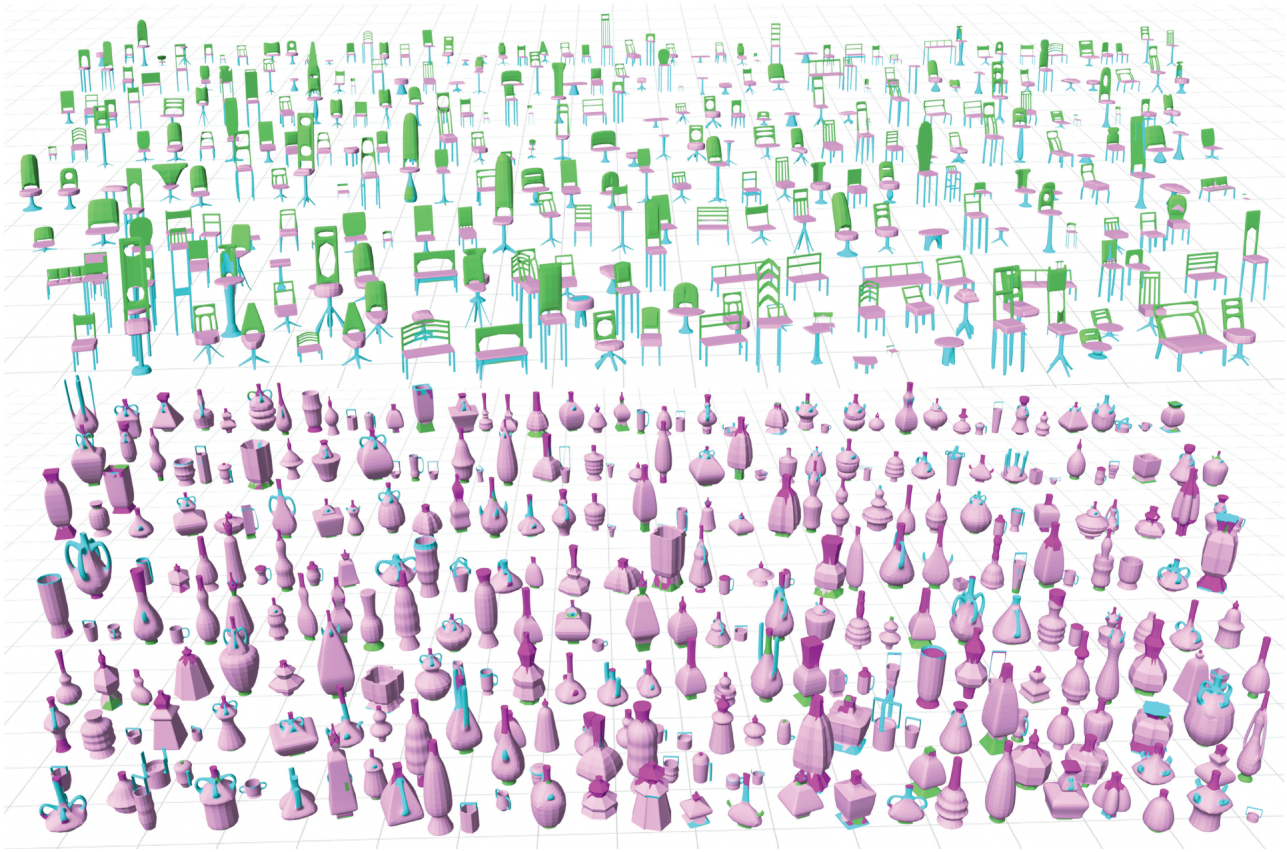


Fig. 16. Large chair set: 400 shapes, 50 shapes for training. Large vase set: 300 shapes, 20 shapes for training.

Table V. Labeling Accuracy of Big Datasets

	Sidi [ToG 2011]	Kim-Auto [ToG 2013]	Ours
Chairs(400)	80.20%	91.20%	92.52%
Vases(300)	69.90%	85.60%	88.54%

approach is superior to state-of-the-art techniques [Sidi et al. 2011; Kim et al. 2013] on these two datasets.

6. CONCLUSION AND DISCUSSION

We present a 3D mesh labeling method based on the deep convolutional neural networks, and demonstrate the powerful ability of our method in 3D shape labeling. Classical representative geometric features are extracted first. We feed our designed deep convolutional neural networks with these features to learn the parameters of the network, yielding high-level features which are more representative than the raw limited-dimension geometric features. For a new 3D mesh, we then extract the high-level features with our learnt networks, and label each triangle face. Finally, a well-known optimization algorithm is applied to improve the local smoothness of triangle labels.

Our method is a purely supervised learning algorithm which is applied to a classical convolutional neural architecture. Generally speaking, better accuracy of labeling can be obtained in a supervised way in comparison with an unsupervised algorithm under the same labeling framework. We believe the accuracy will be improved if we pretrain our convolutional networks using some unsupervised methods. Several recent works [Kavukcuoglu et al. 2009, 2010] have shown that an unsupervised pre-train step fed to a convolutional net would lead to a good starting point before supervised refinement.

There are many other efficient deep architectures, such as deep belief networks [Lee et al. 2009], which have a good ability of object recognition. It would be an interesting task to apply other models to 3D mesh labeling in the future. We choose convolutional neural networks for several reasons. Convolutional neural networks have a special structure shared by local weights, so as to reduce the complexity of the network. And our training samples are triangles, which have more than tens of thousands in the train sets, thus the overfitting can be excluded. The networks architecture we use is simple, otherwise with the growth of the layer number, the parameters would rapidly increase so as to produce overfit, especially with limited training meshes. Meanwhile, it would be an interesting attempt for other complex networks. Moreover, we choose basic sigmoid as activation function rather than hyperbolic tangent function, since the latter is much more fluctuated and not easy to convergence in our experiments.

Last but not least, although the method proposed gets a good performance, there still remain some interesting problems for further discussion. Our method can achieve better performance based on datasets with few label categories. However, the accuracy of almost all existing 3D mesh labeling methods, including ours of course, is in fact still low if the number of the categories is large. To recognize more categories is to recognize tiny parts in a 3D mesh, which is a difficult problem far from being solved.

ACKNOWLEDGMENTS

We would like to thank all the reviewers for their valuable comments and suggestions. We are also grateful to Jia Li for proofreading, and Kai Xing, Han Zhang for data processing.

REFERENCES

O. K.-C. Au, Y. Zheng, M. Chen, P. Xu, and C.-L. Tai. 2012. Mesh segmentation with concavity-aware fields. *IEEE TVCG* 18, 7, 1125–1134.

J. M. Baker, L. Deng, J. Glass, S. Khudanpur, C.-H. Lee, N. Morgan, and D. O. Shaughnessy. 2009. Developments and directions in speech recognition and understanding, part 1. *IEEE Signal Processing Magazine* 26, 3, 75–80.

S. Belongie, J. Malik, and J. Puzicha. 2002. Shape matching and object recognition using shape contexts. *IEEE TPAMI* 24, 4, 509–522.

M. Ben-Chen and C. Gotsman. 2008. Characterizing shape using conformal factors. In *Proc. Eurographics 3DOR*. 1–8.

Y. Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning* 2, 1, 1–127.

Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE TPAMI* 23, 11, 1222–1239.

J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. 2014. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203.

C.-C. Chang and C.-J. Lin. 2011. LIBSVM: A library for support vector machines. *ACM TIST* 2, 27:1–27:27. Software available at www.csie.ntu.edu.tw/∼cjlin/libsvm.

X. Chen, A. Golovinskiy, and T. Funkhouser. 2009. A benchmark for 3D mesh segmentation. *ACM Trans. Graph.* 28, 3, 73:1–73:12.

X. Chen, Y. Guo, B. Zhou, and Q. Zhao. 2013. Deformable model for estimating clothed and naked human shapes from a single image. *The Visual Computer* 29, 11, 1187–1196.

X. Chen, J. Li, Q. Li, B. Gao, D. Zou, and Q. Zhao. 2015a. Image2scene: Transforming style of 3D room. In *Proceedings of ACM MM*. 321–330.

X. Chen, B. Zhou, F. Lu, L. Wang, L. Bi, and P. Tan. 2015b. Garment modeling with a depth camera. *ACM Trans. Graph.* 34, 6.

L. Deng. 2004. Switching dynamic system models for speech articulation and acoustics. In *Proceedings of the IMA Workshop*. Springer, 115–134.

C. Farabet, C. Couprie, L. Najman, and Y. Lecun. 2013. Learning hierarchical features for scene labeling. *IEEE TPAMI* 35, 8, 1915–1929.

R. Gal and D. Cohen-Or. 2006. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.* 25, 1, 130–150.

M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. 2001. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proc. SIGGRAPH*. 203–212.

G. Hinton. 2010. A practical guide to training restricted Boltzmann machines. *Momentum* 9, 1, 926.

R. Hu, L. Fan, and L. Liu. 2012. Co-segmentation of 3D shapes via subspace clustering. *CGF* 31, 5, 1703–1713.

Q. Huang, V. Koltun, and L. Guibas. 2011. Joint shape segmentation with linear programming. *ACM Trans. Graph.* 30, 6, 125:1–125:12.

Q.-X. Huang, H. Su, and L. Guibas. 2013. Fine-grained semisupervised labeling of large shape collections. *ACM Trans. Graph.* 32, 6, 190:1–190:10.

Q.-X. Huang, M. Wicke, B. Adams, and L. Guibas. 2009. Shape decomposition using modal analysis. *CGF* 28, 2, 407–416.

A. E. Johnson and M. Hebert. 1999. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE TPAMI* 21, 5, 433–449.

E. Kalogerakis, A. Hertzmann, and K. Singh. 2010. Learning 3D mesh segmentation and labeling. *ACM Trans. Graph.* 29, 4, 102:1–102:12.

S. Katz and A. Tal. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.* 22, 3, 954–961.

K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. Lecun. 2009. Learning invariant features through topographic filter maps. In *Proc. CVPR*. 1605–1612.

K. Kavukcuoglu, M. Ranzato, and Y. Lecun. 2010. Fast inference in sparse coding algorithms with applications to object recognition. arXiv preprint arXiv:1010.3467.

V. G. Kim, W. Li, N. J. Mitra, S. Chaudhuri, S. Diverdi, and T. Funkhouser. 2013. Learning part-based templates from large collections of 3D shapes. *ACM Trans. Graph.* 32, 4, 70.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*. 1106–1114.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*. 282–289.

H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc. ICML*. 609–616.

R. Liu, H. Zhang, A. Shamir, and D. Cohen-Or. 2009. A part-aware surface metric for shape analysis. *CGF* 28, 2, 397–406.

J. Lv, X. Chen, J. Huang, and H. Bao. 2012. Semi-supervised mesh segmentation and labeling. *CGF* 31, 7, 2241–2248.

S. Lyu and E. P. Simoncelli. 2008. Nonlinear image representation using divisive normalization. In *Proc. CVPR*. 1–8.

L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. 2010. Contextual part analogies in 3D objects. *IJCV* 89, 2–3, 309–326.

O. Sidi, O. Van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or. 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. Graph.* 30, 6, 126:1–126:10.

R. Socher, B. Huval, B. P. Bath, C. D. Manning, and A. Y. Ng. 2012. Convolutional-recursive deep learning for 3D object classification. In *Proc. NIPS*. 665–673.

A. Torralba, K. P. Murphy, and W. T. Freeman. 2007. Sharing visual features for multiclass and multiview object detection. *IEEE TPAMI* 29, 5, 854–869.

O. van Kaick, A. Tagliasacchi, O. Sidi, H. Zhang, D. Cohen-Or, L. Wolf, and G. Hamarneh. 2011. Prior knowledge for part correspondence. *CGF* 30, 2, 553–562.

O. van Kaick, K. Xu, H. Zhang, Y. Wang, S. Sun, A. Shamir, and D. Cohen-Or. 2013. Co-hierarchical analysis of shape structures. *ACM Trans. Graph.* 32, 4, 69:1–69:10.

Y. Wang, S. Asafi, O. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen. 2012. Active co-analysis of a set of shapes. *ACM Trans. Graph.* 31, 6, 165:1–165:10.

Y. Wang, M. Gong, T. Wang, D. Cohen-Or, H. Zhang, and B. Chen. 2013. Projective analysis for 3D shape segmentation. *ACM Trans. Graph.* 32, 6, 192:1–192:12.

Z. Xie, K. Xu, L. Liu, and Y. Xiong. 2014. 3D shape segmentation and labeling via extreme learning machine. *CGF* 33, 5, 85–95.

Y. Yang, W. Xu, X. Guo, K. Zhou, and B. Guo. 2013. Boundary-aware multidomain subspace deformation. *IEEE TVCG* 19, 10, 1633.

Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.* 23, 3, 644–651.

M. D. Zeiler, G. W. Taylor, and R. Fergus. 2011. Adaptive deconvolutional networks for mid and high level feature learning. In *Proc. ICCV*. 2018–2025.

J. Zhang, J. Zheng, C. Wu, and J. Cai. 2012. Variational mesh decomposition. *ACM Trans. Graph.* 31, 3, 21:1–21:14.

Received March 2015; revised August 2015; accepted October 2015