Computer Aided Geometric Design 43 (2016) 39–52

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Unsupervised 3D shape segmentation and co-segmentation via deep learning

Zhenyu Shu[a], Chengwu Qi[b], Shiqing Xin[c,*], Chao Hu[a], Li Wang[a], Yu Zhang[a], Ligang Liu[d]

[a] School of Information Science and Engineering, Ningbo Institute of Technology, Zhejiang University, Ningbo, 315100, PR China
[b] School of Electronic Information Engineering, Taiyuan University of Science and Technology, Taiyuan, 030024, PR China
[c] School of Information Science and Engineering, Ningbo University, Ningbo, 315211, PR China
[d] Graphics & Geometric Computing Laboratory, School of Mathematical Sciences, University of Science and Technology of China, Anhui, 230026, PR China

ARTICLE INFO

Article history:
Available online 18 February 2016

Keywords:
3D shapes
Segmentation
Co-segmentation
Deep learning
High-level features

ABSTRACT

In this paper, we propose a novel unsupervised algorithm for automatically segmenting a single 3D shape or co-segmenting a family of 3D shapes using deep learning. The algorithm consists of three stages. In the first stage, we pre-decompose each 3D shape of interest into primitive patches to generate over-segmentation and compute various signatures as low-level shape features. In the second stage, high-level features are learned, in an unsupervised style, from the low-level ones based on deep learning. Finally, either segmentation or co-segmentation results can be quickly reported by patch clustering in the high-level feature space. The experimental results on the Princeton Segmentation Benchmark and the Shape COSEG Dataset exhibit superior segmentation performance of the proposed method over the previous state-of-the-art approaches.

1. Introduction

Automatic segmentation of 3D shapes is a fundamental operation in geometric modeling and shape processing (Wu et al., 2013). It helps shape understanding and is also central to many computer graphics problems, including mesh parameterization, skeleton extraction, resolution modeling, shape retrieval and so on. Most of existing segmentation algorithms partition a single 3D shape depending on a specific kind of signature that is often invariant to a certain transformation group (Veltkamp and Hagedoorn, 2001; Hong and Soatto, 2015).

Known signatures include scale-invariant heat kernel signatures (SIHKS) (Bronstein and Kokkinos, 2010), shape diameter function (SDF) (Shapira et al., 2008), Gaussian curvature (GC) (Gal and Cohen-Or, 2006) and so on. To our knowledge, they can be used for shape segmentation purpose. However, shape understanding is a complicated task and thus we can't rely on a single signature to settle the segmentation problem once and for all. This is due to the fact that a signature, represented as a statistics or deterministic function, can only characterize the geometric shapes from a special perspective.

Recently, researchers find that simultaneously segmenting a set of 3D shapes within the same class into consistent decompositions, i.e., *co-segmentation*, is possible to achieve a better segmentation result than the traditional segmentation that is targeted at a single object. Some of these algorithms (Kalogerakis et al., 2010; van Kaick et al., 2011) require labeled train-

* Corresponding author at: School of Information Science and Engineering, Ningbo Institute of Technology, Zhejiang University, Ningbo, 315100, PR China.
E-mail addresses: shuzhenyu@nit.zju.edu.cn (Z. Shu), xinshiqing@nbu.edu.cn (S. Xin).

http://dx.doi.org/10.1016/j.cagd.2016.02.015
0167-8396/© 2016 Elsevier B.V. All rights reserved.

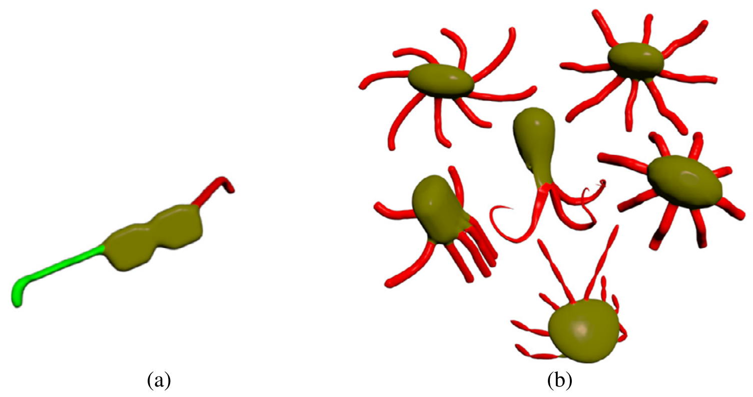


Fig. 1. The experimental results of our approach. (a) Segmentation of a single shape. (b) Co-segmentation of a family of similar shapes.

ing data to learn common segmentation rules. Generally speaking, the supervised algorithms are able to produce a desirable segmentation result if the labeled data set is sufficiently large. However, labeling 3D shapes is pretty time-consuming and tedious. By contrast, the unsupervised methods (Sidi et al., 2011; Huang et al., 2011; Hu et al., 2012; Meng et al., 2013; Wu et al., 2013) can segment 3D shapes automatically, without extra labeling. Generally, it has to be at the cost of segmentation performance.

In this paper, we propose an effective unsupervised method for shape segmentation/co-segmentation based on deep learning. In the very beginning, we decompose each 3D shape into primitive patches to generate over-segmentation and compute various shape signatures for the input models. The signatures used in this paper include SIHKS (Bronstein and Kokkinos, 2010), SDF (Shapira et al., 2008) and GC (Gal and Cohen-Or, 2006). However, each separate signature can only characterize part of geometric features. Therefore, in the next stage, we build a deep neural network for unsupervised learning such that high-level features can be extracted from the geometric signatures computed in the first stage. It's noted that this stage doesn't need a labeling operation. With the support of unsupervised deep learning, we can finally segment models guided by the high-level features. Fig. 1 shows an example of segmentation and co-segmentation computed by our approach.

We evaluate our approach on two open datasets, including the Princeton Segmentation Benchmark (Chen et al., 2009) and the Shape COSEG Dataset (Wang et al., 2012). Extensive experimental results exhibit the superior segmentation/co-segmentation performance of the proposed method over the previous state-of-the-art approaches.

Contributions. Our contributions are twofold.

- We introduce deep learning into the problem of shape segmentation and co-segmentation such that various shape signatures can be integrated into a high-level feature space. This algorithmic framework is extensible – it supports a variety of shape signatures and hopefully achieves better segmentation results if some new signatures are considered in this framework.
- Our method is data-driven but does not need a tedious labeling process. The new approach, in its nature, is also adaptive to different databases.

The remaining of the paper is organized as follows. Section 2 reviews the related work on model segmentation and shape descriptors. Section 3 presents the overall segmentation framework followed by detailed construction process of high-level features. The unsupervised deep learning technique is detailed in Section 4. After that, we give the segmentation and co-segmentation algorithm in Section 5. In Section 6, we show extensive experimental results, as well as comparisons with the state of the art. Finally, we give limitations and future work in Section 7 and conclude this paper in Section 8.

2. Related work

2.1. 3D shapes segmentation and co-segmentation

Shape segmentation (Attene et al., 2006; Shamir, 2008) aims at segmenting a 3D shape into meaningful parts and plays an important role in shape analysis and shape understanding. So far, a lot of methods have been proposed for solving this problem. A common practice is to build a shape signature by extracting a kind of geometric properties and then apply it in shape segmentation using some decomposition techniques, such as approximate convexity analysis (Kaick et al., 2014), concavity-aware fields (Au et al., 2012), extreme learning machine (Xie et al., 2014), spectral clustering (Rong and Hao, 2004), K-Means (Shlafman et al., 2002), core extraction (Katz et al., 2005), graph cuts (Golovinskiy and Funkhouser, 2008; Katz and Tal, 2003), random walks (Lai et al., 2008), randomized cuts (Golovinskiy and Funkhouser, 2008), normalized cuts

(Golovinskiy and Funkhouser, 2008), and so on. However, shape segmentation depends on the way how people understand a shape and thus is a very challenging task. An individual signature cannot provide sufficient geometric cues to differentiate meaningful parts (Chen et al., 2009).

Recently, researchers find that if the input is a family of 3D models that are deemed to be in the same class, segmenting the models guided by the underlying relevance is possible to get better results than segmenting them individually. This is the so-called *co-segmentation* problem. For example, Golovinskiy and Funkhouser (2009) transformed co-segmentation into a graph clustering problem. Their assumption is that a global rigid alignment exists between the input shapes, which facilitates iteratively establishing correspondence between respective parts. Later, Xu et al. (2010) used anisotropic part scales to part correspondence and their algorithm performs well on a diverse set of shapes. Zheng et al. (2014) suggested an indirect top-down approach to deal with large shape variations.

Because 3D shape segmentation can actually be regarded as a face clustering process in the feature space, a data-driven method may be used to obtain better performance. For example, Kalogerakis et al. (2010) proposed a supervised approach to do labeling and segmentation simultaneously. They showed that the segmentation performance can be dramatically improved by learning techniques. van Kaick et al. (2011) introduced an approach to part correspondence which incorporates prior knowledge imparted by a training set of pre-segmented, labeled models and combines the knowledge with content-driven analysis based on geometric similarity between the matched shapes. However, these supervised methods often require a huge set of manually labeled segmentation results, which greatly limits its use.

To overcome this problem, Sidi et al. (2011) presented an unsupervised method that takes co-segmentation as a clustering problem in a descriptor space. Huang et al. (2011) proposed to formulate the joint segmentation problem as an integer quadratic programming problem. Experimental results show that co-segmentation significantly outperforms traditional segmentation that targets at a single shape. Hu et al. (2012) presented an unsupervised approach for co-segmentation by over-segmenting the input models into primitive patches and then grouping similar patches via a subspace clustering scheme. Meng et al. (2013) suggested improving the co-segmentation results by a multi-label optimization process. Wang et al. (2012) presented a semi-supervised learning method where the user actively assists in the co-analysis by iteratively providing inputs. Their method requires only a sparse set of constraints to quickly converge toward a consistent and error-free semantic labeling of the set. Recently, Wu et al. (2013) suggested a spectral clustering method that generates consistent segmentation by performing spectral clustering in a fused space of shape descriptors.

To our best knowledge, we are the first to introduce deep learning to 3D shape segmentation. It is worth pointing out that the distinguished feature of our method is that it directly learns from unlabeled input 3D shapes and does not require manual labeling. Our approach is different than previous learning-based methods (Kalogerakis et al., 2010), where users have to label a lot of models for training. It is also different than optimization-based methods that are devised to select a desirable combination of shape features (Hu et al., 2012; Kalogerakis et al., 2010).

2.2. Shape descriptors

Shape descriptors are central to 3D shape segmentation. In recent years, numerous feature descriptors have been proposed, such as GC, SDF, average geodesic distance (AGD) (Shapira et al., 2010) and shape distribution (D2) (Osada et al., 2002). Roughly speaking, existing shape descriptors fall into two categories. One kind is global feature descriptors that describe the geometric properties of the overall shape. For example, Gatzke et al. (2005) built a curvature map signature for model matching based on geodesic distances. Vranic (2003) designed a rotation invariant feature vector based on functions on concentric spheres. Loffler (2000) proposed to convert a 3D shape to a series of 2D images. The other kind is local feature descriptors. For example, Bronstein and Kokkinos (2010) developed a scale-invariant heat kernel descriptor. The construction is based on a logarithmically sampled scale-space in which shape scaling corresponds to a translation. Knopp et al. (2010) presented a local 3D shape descriptor by using Hough-voting. Smeets et al. (2013) proposed a four-step algorithm to generate a local shape descriptor for face recognition under expression variations and partial data.

By contrast, the proposed shape segmentation and co-segmentation method in this paper uses high-level features learned from multiple shape signatures by a deep learning framework, rather than directly employs the input 3D feature descriptors. We use a collection of geometry-based shape descriptors as the input of multiple-level neural networks, which is different than the case in the computer vision field where images are represented as a matrix structure and high-level features are usually learned from pixels or pixel blocks that are the input of neural networks. Generally, the first few layers produce low-level features, while the last few layers produce high-level ones. However, the polygonal mesh based representation is not so regular as images and an input model may have a complicated shape and topology. It does not make sense to directly take triangles as the input of neural networks.

3. Overview

Our algorithm works on a 3D model database and it consists of four stages, as illustrated in Fig. 2. First, we compute the primitive patches for each shape independently. Then in the next stage, we calculate feature vectors for each patch. After that, we take the extracted feature vectors as the input of a deep neural network and generate a high-level feature space. Finally, we conduct segmentation and co-segmentation on the dataset by performing a clustering operation in the high-level feature space.

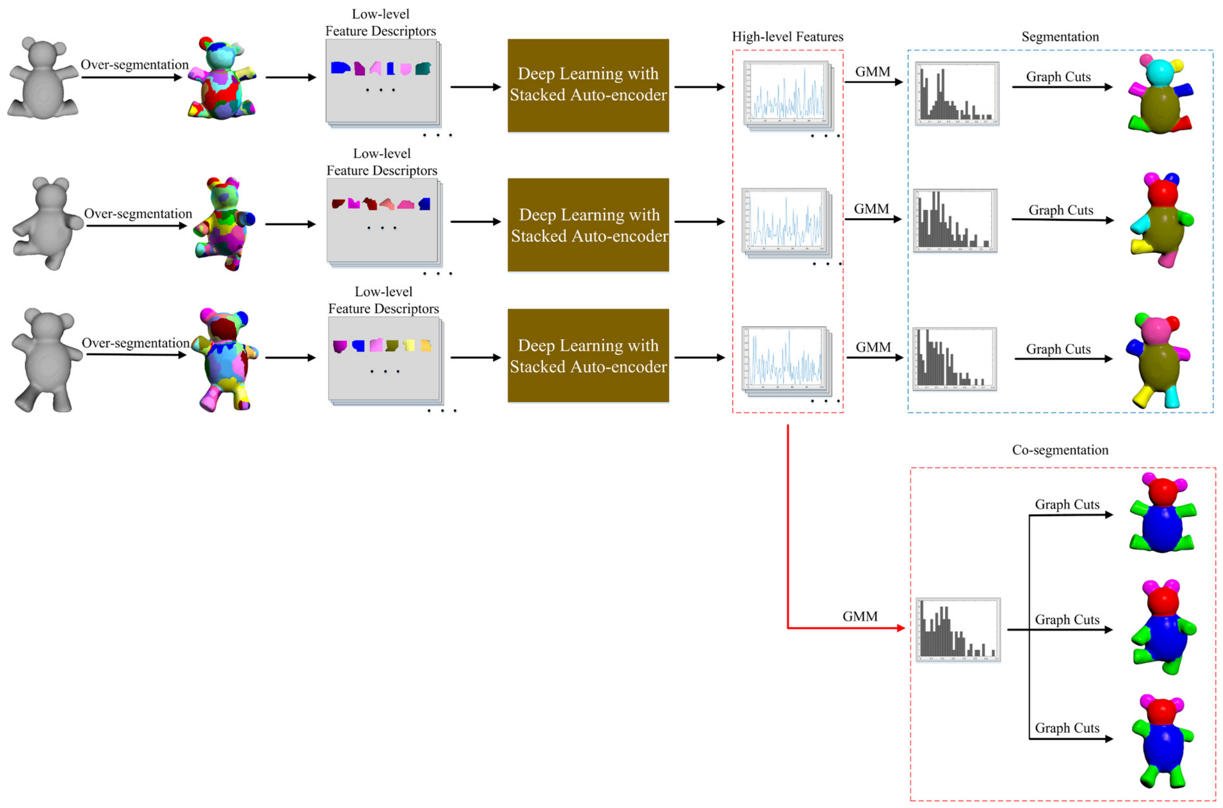


Fig. 2. The overall workflow of the proposed approach for 3D shape segmentation and co-segmentation.

3.1. *Over-segmentation*

Similar to the super-pixel based image segmentation (Ren and Malik, 2003; Shi and Malik, 2000), we divide each shape into primitive patches in the first stage. In implementation, we convert the input mesh into its dual graph and then associate two weights to each graph arc, i.e., a traversal cost, and a cut cost, which are defined based on dihedral angles. Since low-cost cuts in the graph correspond to favorable segmentation boundaries in the mesh, and low-cost traversal paths in the graph occur between points on the mesh likely to be in the same functional parts, we finally decompose the mesh into a number of patches, like that achieved in Golovinskiy and Funkhouser (2008). In our experiments, the number of patches is set to be $L = 50$; see Section 6 for details.

3.2. *Learning high-level features*

There is a common view in the computer vision field – high-level features are often learned from low-level ones. Patel et al. (2015) developed a novel probabilistic framework that accounts for why deep learning is able to work well in practice. In order to inherit the spirit of deep learning, we use low-level feature descriptors as the input and finally generate a high-level feature space. In the experimental setting, we select three widely known feature descriptors, including SIHKS, SDF and GC. These feature descriptors are deemed to have a capability of characterizing the geometric properties well from different perspectives.

For SDF and GC that are a scalar field on each patch, it is very easy to capture the feature distribution using histograms. For SIHKS that is a vector field on each patch, we extract the 1D feature distribution by the famous bag-of-feature (BoF) technique. It's noted that the number of bins in the histograms and that of the bags for the bag-of-feature representation are both set to be $B = 100$. In this way, any feature descriptor can be adapted into our algorithm framework. After that, we need to concatenate the low-level feature vectors and take them as the input of a deep neural network. Based on the deep learning technique, we can finally get a new feature space that characterizes the high-level features.

3.3. *Segmentation and co-segmentation*

For both segmentation and co-segmentation, we need to perform a clustering operation in the high-level feature space – first use the Gaussian mixture model (GMM) to define the probability for representing the presence of each patch within a cluster and then use the graph cuts algorithm (Boykov et al., 2001) to get the final results. The difference between

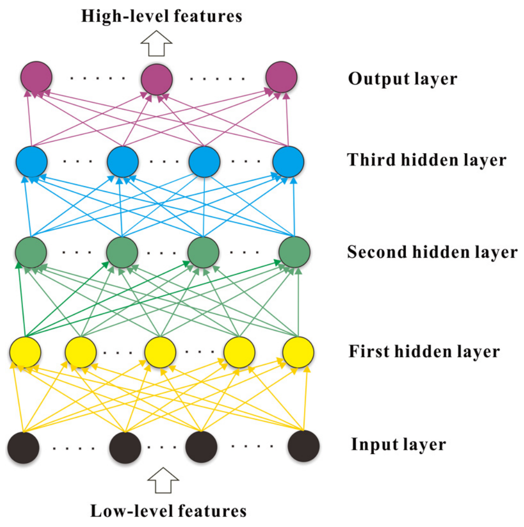


Fig. 3. The architecture of our deep neural network.

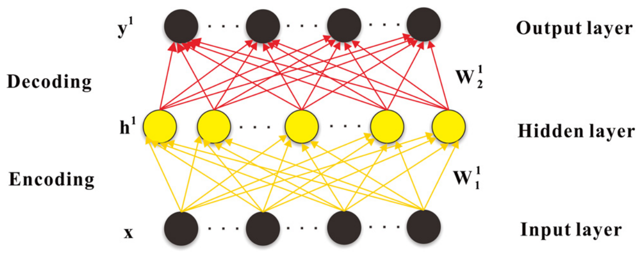


Fig. 4. The illustration of an auto-encoder (the bias parameters b_1 and b_2 are omitted for clarity).

segmentation and co-segmentation, in our approach, is that co-segmentation needs to define a common GMM to guide the consistent segmentation of a family of models.

4. Deep learning

In this section, we detail the architecture of the deep neural network used in this paper. For purpose of unsupervised learning, we use the stacked auto-encoders (SAEs) (Bengio et al., 2007) to construct high-level features since it is a commonly used technique in the research community.

4.1. The architecture of network

The architecture of our deep neural network is summarized in Fig. 3. The deep neural network has five layers in total: the input layer, three hidden layers and the output layer. The first layer takes a collection of 300-dimensional input vectors. The numbers of vector dimensions in the three hidden layers are respectively 350, 200, 150. We get a collection of 100-dimensional high-level feature vectors in the output layer.

4.2. Auto-encoders

SAEs is a neural network composed of multiple layers of sparse auto-encoders. It is often used for training in a fully unsupervised way. As shown in Fig. 4, the auto-encoder consists of an input layer, a hidden layer and an output layer. The auto-encoder is able to learn latent features of the input by minimizing the discrepancy between the input features and the reconstruction one from the latent features.

In Fig. 4, the bottom mapping represents the stage of the encoder, while the top mapping represents that of the decoder. Let N_I and N_H be the numbers of units in the input layer and the hidden layer respectively. Given a feature vector $x \in R^{N_I \times 1}$, the auto-encoder transforms x to a latent representation h^1 by a compound mapping of a linear transformation and a non-linear activation function s as follows:

$$h^1 = s(w_1^1 x + b_1^1), \tag{1}$$

where $h^1 \in R^{N_H}$ is the latent data, $w_1^1 \in R^{N_H \times N_I}$ is the encoding weight matrix, $b_1^1 \in R^{N_H}$ is the bias vector, and $s(\cdot)$ is the sigmoid function:

$$s(a) = \frac{1}{1+\exp(-a)}. \tag{2}$$

Then the latent representation h^1 is mapped to a feature vector $y^1 \in R^{N_I}$, which approximately reconstructs the input feature vector x by employing a compound mapping of a linear transformation and a non-linear activation function as follows:

$$y^1 = s(w_2^1 h^1 + b_2^1), \tag{3}$$

where h^1 is the latent data, $w_2^1 \in R^{N_I \times N_H}$ is the decoding weight matrix, and $b_2^1 \in R^{N_I}$ is the bias vector. Then we can learn the underlying features by minimizing the reconstruction error of the cost function:

$$C(x, y) = \frac{1}{2}\sum_{i=1}^{m} \|y_i - x_i\|^2, \tag{4}$$

where x is the input feature vector, m is the number of the input samples, and y is the reconstructed feature vector. Under certain circumstances, some weights may result in over-fitting (Dietterich, 1995). Hence, a regularization term D, also called the weight decay, is introduced for avoiding over-fitting. And the Equation (4) can be redefined as follows:

$$C(x, y, \theta) = \frac{1}{2}\sum_{i=1}^{m} \|y_i - x_i\|^2 + \lambda D(\theta), \tag{5}$$

where λ is the weight decay parameter, $\theta = \{w, b\}$, w and b represent the weights and the biases of the auto-encoder respectively, and $D(\theta) = \sum_{i=1}^{|\theta|} \theta_i^2$.

If the number of units in the hidden layer is larger (or equal) than that in the input layer, the auto-encoder may learn some useless knowledge from the input features. In order to avoid this situation, a sparsity constraint can be enforced on the hidden layer. We use a sparsity constraint based on the Kullback–Leibler (KL) divergence (Perez-Cruz, 2008) in this paper. To this end, the optimization problem can be formulated as follows:

$$\arg\min_{\theta} C(x, y, \theta) + \tau \sum_{i=1}^{N_H} KL\left(\rho \| \hat{\rho}_i\right), \tag{6}$$

where,

$$KL\left(\rho \| \hat{\rho}_i\right) = \rho \log \frac{\rho}{\hat{\rho}_i} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_i},$$

τ is the weight of sparsity penalty term, N_H is the number of the hidden units, ρ is the sparsity parameter, and $\hat{\rho}_i = \frac{1}{N_I}\sum_{j=1}^{N_I} (h_i)_j$ is the average activation of the hidden unit h_i. From problem (6), we can see that the sparsity penalty term will vanish if $\hat{\rho}_i = \rho$. So the closer $\hat{\rho}_i$ and ρ are, the sparser the hidden layer will be.

With this formulation, we use the back propagation algorithm (Rummelhart, 1986) and the gradient descent approach to train the auto-encoder by optimizing the cost function with respect to θ.

4.3. Stacked auto-encoders

SAEs is composed of multiple layers of sparse auto-encoders, where the latent features learned in previous auto-encoder are used as the input of the next auto-encoder. The whole training process of a SAEs is carried out in a greedy layer-wise way (Hinton and Salakhutdinov, 2006). Hinton et al. (2006) find that this approach can generate better parameters for deep neural networks and produce desirable results.

The details of the training process can be illustrated as Fig. 5. As shown in Fig. 5, after training each auto-encoder using the method described in Section 4.2, the outputs w_2^i and y^i of the auto-encoder are discarded and the latent features h^i of the auto-encoder are used to feed the next auto-encoder. When all auto-encoders are trained, we further employ the forward propagation method, based on the parameters w_1^i, to get the high-level feature vectors.

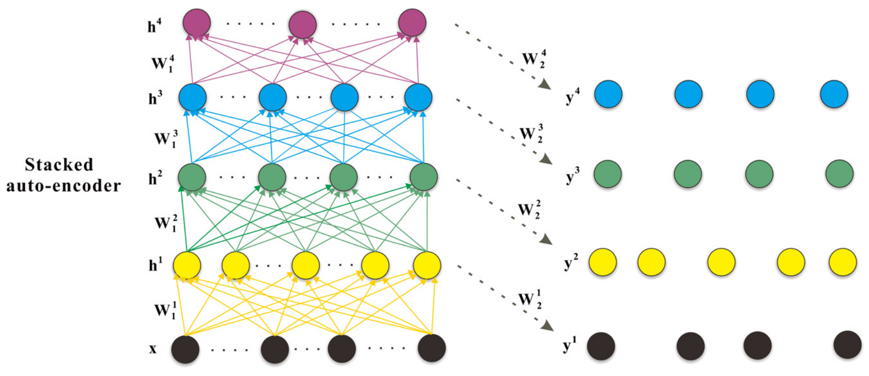


Fig. 5. Pre-training of SAEs.

Table 1
The number of sample shapes in each category of the experimental shape datasets.

Object categories	Amount	Object categories	Amount
Human	20	Bird	20
Cup	20	Armadillo	20
Glasses	20	Bust	20
Airplane	20	Mech	20
Ant	20	Bearing	20
Chair	20	Vase	20
Octopus	20	Fourleg	20
Table	20	Candelabra	28
Teddy	20	Goblet	12
Hand	20	Guitar	44
Plier	20	Lamp	20
Fish	20		

5. Segmentation and co-segmentation using high-level features

For segmenting a single 3D shape, we cluster the patches of the shape by considering their corresponding high-level feature vectors. GMM is employed for clustering in our method, resulting in a probability matrix depicting the probabilities for a patch belonging to a cluster. GMM enables us to utilize the graph-cut algorithm in the final the segmentation step. It's noted that the energy term is computed like that achieved in Meng et al. (2013).

For co-segmenting a set of 3D shapes from the same class, we cluster all the shape patches on a common basis, assuming that these models have the same number of clusters and a group of corresponding clusters are similar in shape. Again, we use GMM to get the probability matrix of a patch within a cluster, and the graph-cut algorithm to get the final co-segmentation results. To further boost the segmentation performance, we employ fuzzy cuts (Katz and Tal, 2003) to refine the jaggy boundaries between adjacent parts. In our implementation, we refine the boundaries in a fuzzy region that is 10 faces wide.

To demonstrate the effectiveness of high-level features and SAEs, we visualize the segmentation results in Fig. 6, where the features for segmentation are extracted from the corresponding layer. It can be seen that (1) the results are better if we use high-level features to guide segmentation, (2) the feature learning process by using SAEs is very useful for improving the segmentation performance, and (3) even if the Buddha model has a complicated shape and topology, our algorithm can still give a desirable segmentation result. A quantitative comparison for using different feature descriptors is also provided in Table 2 (see the last four columns).

6. Evaluation

In this section, we use extensive experimental statistics and results to validate our algorithm.

Experimental dataset. We test the segmentation algorithm on the Princeton Segmentation Benchmark (PSB) dataset with 19 different object categories, which is an open dataset for 3D shape segmentation and shape retrieval. In order to test co-segmentation, we use a composed database suggested in Hu et al. (2012) that has 20 different object categories, 16 categories from PSB and 4 from COSEG (Wang et al., 2012). We leave out 3 categories (Bust, Mech and Bearing) from 19 categories in PSB since the shapes in the three categories do not have meaningful correspondence. The detailed statistics of the datasets used in our experiments are shown in Table 1.

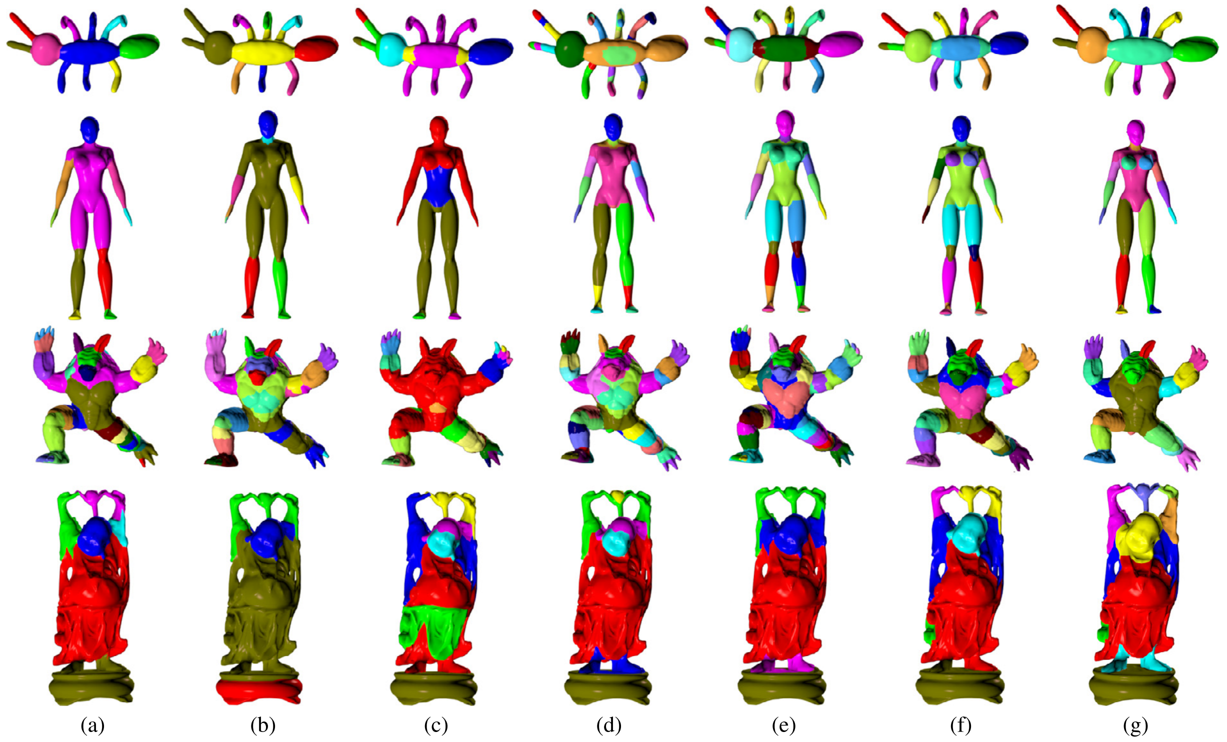


Fig. 6. Visualization of the segmentation results using different features. (a) SDF; (b) GC; (c) SIHKS; (d) Concatenating SDF, GC and SIHKS together; (e) The features extracted from the 2nd layer of our SAEs; (f) The features extracted from the 4th layer of our SAEs; (g) The features extracted from the final layer of our SAEs, i.e., the high-level features used in our method.

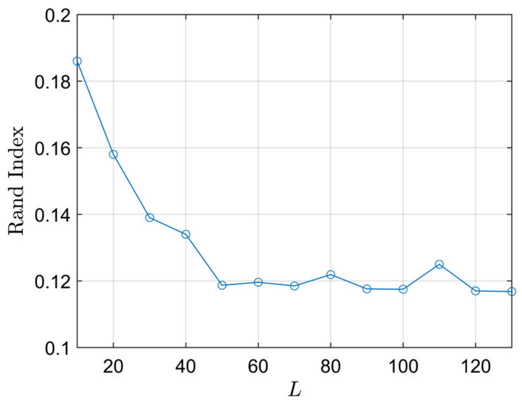
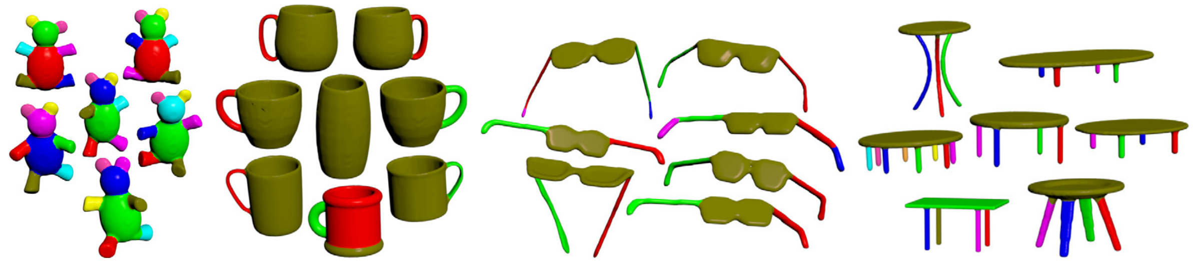


Fig. 7. The plot of the average Rand Index scores with regard to L on the PSB dataset, where lower values indicate better results.

For the 4 categories of 3D shapes selected from COSEG, we use the ground-truth provided by the authors (Wang et al., 2012) for evaluation. For the other 16 categories of 3D shapes from PSB, we use the human-generated labeling provided by Chen et al. (2009) as the ground truth of segmentation and the manually labeled training data provided by Kalogerakis et al. (2010) as the ground truth of co-segmentation respectively. Note that, there are 15 human-generated segmentations for each 3D shape in Chen et al. (2009), while only one for each shape in Kalogerakis et al. (2010).

Evaluation metrics. To evaluate our segmentation method, we adopt four metrics that are defined by Chen et al. (2009), including Rand Index, Cut Discrepancy, Hamming Distance and Consistency Error. Rand Index, named after William M. Rand, measures the similarity between two segmentations of the same shape. Cut Discrepancy is a boundary-based method evaluating the distance between different cuts. Hamming Distance, named after Richard Hamming, is a region-based method and measures the number of substitutions required to change one region into the other. Consistency Errors, whether the global version (GCE) or local version (LCE), are used to compute the hierarchical differences and similarities between segmentations.



(a) Teddy (b) Cup (c) Glass (d) Table

Fig. 8. Representative segmentation results produced by our approach on the Princeton segmentation benchmark dataset.

Table 2
The Rand Index scores of segmentation for each category with different methods and different feature descriptors, where "TFC" means the concatenation of GC, SDF and SIHKS.

Object categories	Ours	WcSeg	RandCuts	NormCuts	RandWalks	Kmeans	SDF	GC	SIHKS	TFC
Human	**0.116**	0.128	0.136	0.150	0.176	0.160	0.159	0.158	0.265	0.154
Cup	**0.096**	0.171	0.266	0.411	0.416	0.520	0.602	0.610	0.470	0.600
Glasses	0.173	0.173	**0.119**	0.151	0.244	0.201	0.223	0.237	0.276	0.227
Airplane	0.150	**0.089**	0.126	0.184	0.235	0.209	0.189	0.217	0.311	0.210
Ant	**0.001**	0.021	0.050	0.090	0.084	0.115	0.087	0.119	0.166	0.092
Chair	**0.040**	0.103	0.197	0.112	0.160	0.210	0.192	0.202	0.246	0.182
Octopus	0.036	**0.029**	0.096	0.104	0.103	0.141	0.133	0.152	0.221	0.127
Table	**0.040**	0.091	0.401	0.266	0.273	0.441	0.398	0.383	0.330	0.432
Teddy	**0.024**	0.056	0.064	0.133	0.125	0.186	0.173	0.194	0.267	0.174
Hand	0.135	0.116	**0.113**	0.166	0.191	0.185	0.167	0.206	0.364	0.201
Plier	0.151	**0.087**	0.133	0.191	0.220	0.196	0.204	0.211	0.271	0.206
Fish	0.288	**0.203**	0.300	0.367	0.394	0.413	0.399	0.402	0.355	0.363
Bird	0.171	**0.101**	0.116	0.191	0.227	0.189	0.212	0.228	0.340	0.238
Armadillo	**0.073**	0.081	0.112	0.142	0.133	0.129	0.112	0.118	0.252	0.118
Bust	0.275	0.266	**0.247**	0.330	0.329	0.353	0.373	0.374	0.360	0.359
Mech	**0.073**	0.182	0.342	0.349	0.367	0.469	0.442	0.209	0.382	0.419
Bearing	**0.056**	0.122	0.160	0.235	0.288	0.289	0.212	0.205	0.446	0.257
Vase	0.212	0.161	**0.157**	0.300	0.311	0.397	0.379	0.398	0.360	0.384
Fourleg	**0.140**	0.152	0.179	0.209	0.234	0.197	0.178	0.193	0.355	0.189
Average	**0.118**	0.123	0.174	0.216	0.237	0.263	0.254	0.254	0.318	0.259

Parameter settings. In this paper, the coefficient λ of the weight decay, the weight of sparsity penalty term τ and the sparsity parameter ρ in the optimization problem (6) is set to be 0.0001, 0.01 and 0.05 respectively. In our experiments, we tried various choices of L. Fig. 7 shows the average Rand Index scores with regard to different values of L on PSB. We can see that the Rand Index score gets worse if L is far less than 50, since in this case our algorithm cannot capture small parts very well. On the contrary, it is problematic if L is far larger than 50. First, it will make the constructed feature vectors have a huge size if there are only a few triangles in each patch. Second, the algorithm will be very inefficient if there are too many patches. Based on the above observation, the number of patches L is set to 50 for each 3D shape in our experiments.

6.1. Results

Fig. 8 shows the segmentation results for some representative categories of 3D shapes in PSB. Although there are large shape variations, the absolute majority of the segmentation results are desirable and consistent with our perception. The Rand Index score statistics of our segmentation on the PSB dataset, as well as those of other methods, are detailed in Table 2, from which we can see that our algorithm obtains an average Rand Index of 0.118 that outperforms the related algorithms.

Fig. 9 shows the co-segmentation results of some 3D shapes in PSB and COSEG datasets. We can see that our method produces consistent results even if there are large topological differences. For example, the five Chair models are very different from each other, but our algorithm still gives desirable co-segmentation. The co-segmentation of the Candelabra models also exhibits the powerfulness of our algorithm. For quantitatively evaluating our co-segmentation algorithm, we use the Rand Index metric to carry out the comparison between our co-segmentation method and other co-segmentation methods. The detailed statistics of our co-segmentation results on the PSB and COSEG datasets are shown in Table 3. The overall average Rand Index score is 0.089, which is obtained by first computing the average Rand Index score for each category respectively, and then averaging on all categories in the corresponding dataset.

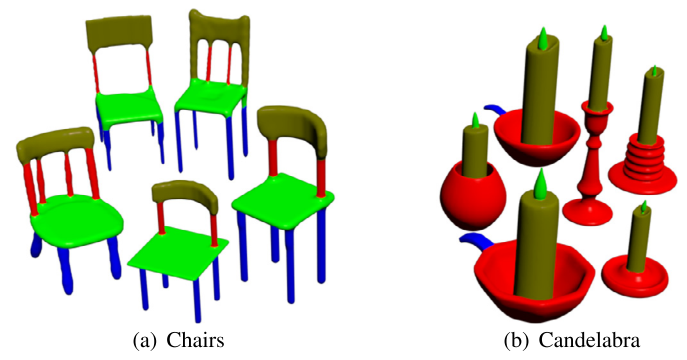


Fig. 9. Representative co-segmentation results produced by our approach on the Princeton segmentation benchmark and COSEG datasets.

Table 3
Average Rand Index scores of our co-segmentation results.

Object categories	Rand index	Object categories	Rand index
Human	0.148	Plier	0.147
Cup	0.038	Fish	0.159
Glasses	0.055	Bird	0.168
Airplane	0.147	Armadillo	0.092
Ant	0.012	Vase	0.198
Chair	0.076	Fourleg	0.128
Octopus	0.028	Candelabra	0.063
Table	0.019	Goblet	0.021
Teddy	0.026	Guitar	0.041
Hand	0.138	Lamp	0.069
		Average	**0.089**

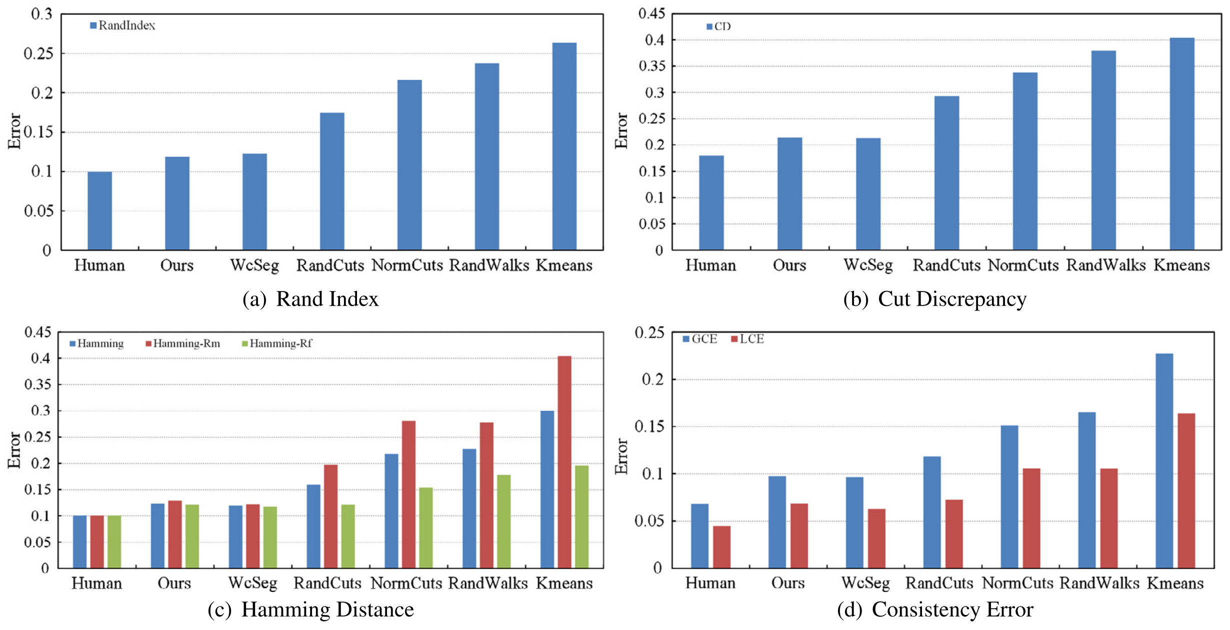


Fig. 10. Performance plots of different segmentation algorithms with respect to four evaluation metrics. Lower values indicate closer similarity to the human-generated ground truth.

6.2. Comparison

Comparisons with previous segmentation algorithms. We compare our method with the other five segmentation algorithms including Randomized cuts (Golovinskiy and Funkhouser, 2008), Normalized cuts (Golovinskiy and Funkhouser, 2008), Random walks (Lai et al., 2008), K-Means (Shlafman et al., 2002) and approximate convexity analysis (WcSeg) (Kaick et al., 2014) on

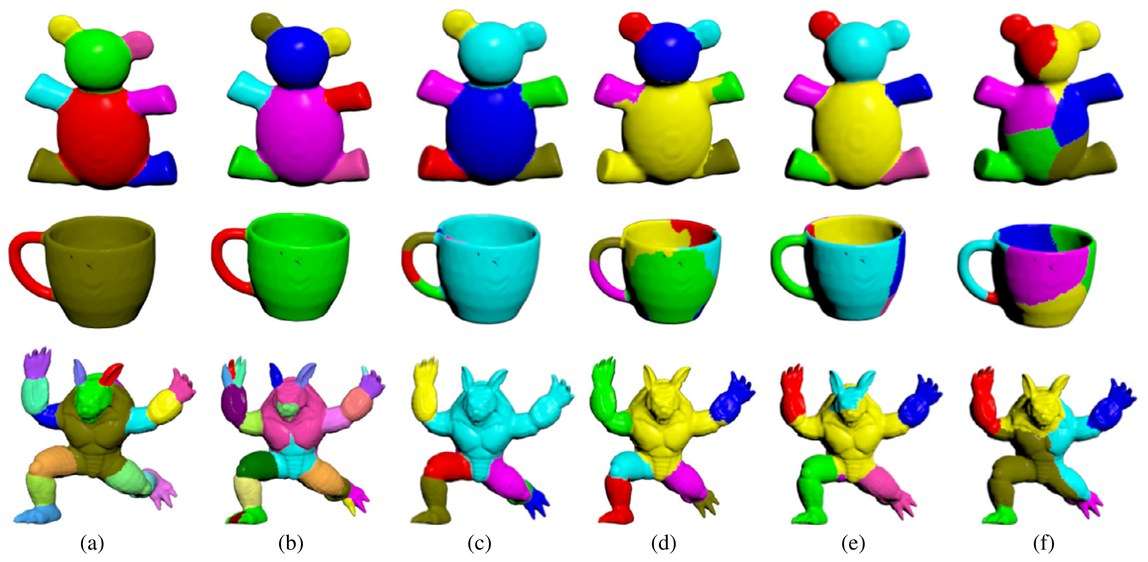


Fig. 11. Comparison with the other five segmentation algorithms. The approaches used here include (a) Ours; (b) WcSeg; (c) Randomized cuts; (d) Normalized cuts; (e) Random walks; (f) K-Means.

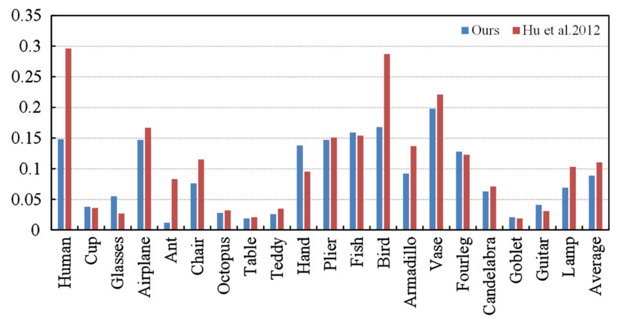


Fig. 12. Comparison between our approach with the state-of-the-art co-segmentation technique (Hu et al., 2012) on some categories using Rand Index scores. Lower values indicate closer similarity to the human-generated ground truth.

the PSB database. All the related segmentation algorithms are compared against human-generated segmentations (available in the PSB database). We use four metrics for evaluation and the detailed statistics plots are shown in Fig. 10. It can be seen that our segmentation method outperforms the other segmentation approaches no matter what kind of evaluation metrics is used, except the WcSeg method. But basically, as the Rand Index score indicates, our algorithm is better than the WcSeg algorithm and as far as the other three metrics are concerned, it is comparable to the WcSeg method. In Fig. 10, the leftmost bars of each subfigure with a “Human” label illustrate the performance of the human-generated segmentations. No matter what kind of metrics is used, lower values mean closer similarity to human-generated ground truth. Fig. 11 provides a qualitative comparison on the Teddy model, the Cup model and the Armadillo model.

Comparisons with previous co-segmentation algorithms. We make comparisons with two state-of-the-art algorithms: the unsupervised subspace clustering method (Hu et al., 2012) and the unsupervised affinity aggregation spectral clustering approach (Wu et al., 2013). The test datasets include PSB and COSEG. Fig. 12 and Fig. 13 show the comparisons with the methods proposed in Hu et al. (2012) and (Wu et al., 2013) respectively.

In Fig. 12, we can see that the average Rand Index score of the method proposed in Hu et al. (2012) is 0.11, worse than our average Rand Index score 0.089. Fig. 14 shows some segmentation results to clearly visualize the difference between the method proposed in Hu et al. (2012) and our algorithm. It can be seen that our method is able to precisely identify the desired boundaries of these Chair models, while their algorithm cannot.

Compared with the unsupervised affinity aggregation spectral clustering approach (Wu et al., 2013), our algorithm also exhibits an advantage. Our average Rand Index score is 0.08 while their average score is 0.093, as shown in Fig. 13. In Fig. 15, we can see that their algorithm cannot separate the top and middle parts of the bottom Cup model (see Fig. 15(b)) while our method can (see Fig. 15(a)).

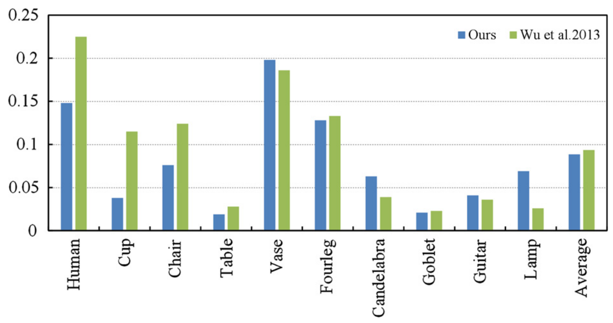


Fig. 13. Comparison between our approach with the state-of-the-art co-segmentation technique (Wu et al., 2013) on some categories using Rand Index scores. Lower values indicate closer similarity to the human-generated ground truth.

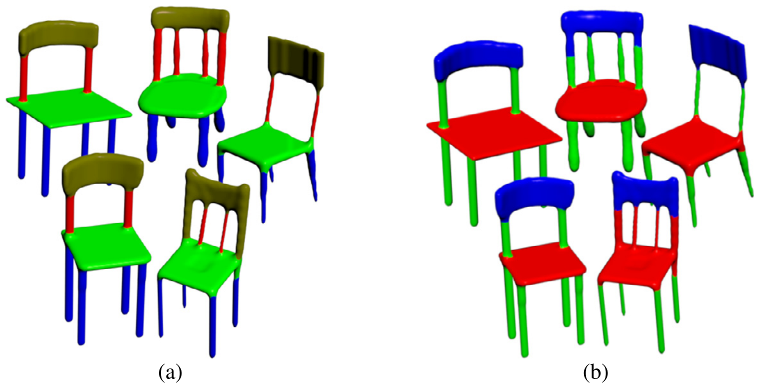


Fig. 14. Co-segmentation results of the Chair sets by applying the method of ours (a) and the method proposed in Hu et al. (2012) (b).

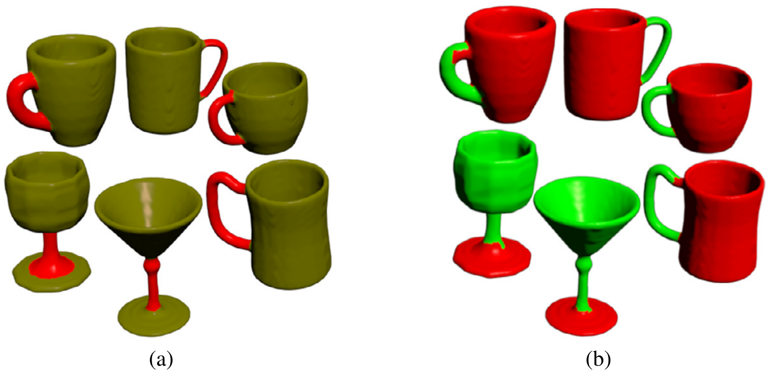


Fig. 15. Co-segmentation results of the Cup sets by applying the method of ours (a) and the method proposed in Wu et al. (2013) (b).

Table 4
The average computation time of different algorithms for segmentation and co-segmentation ('+': measured on a PC with 2.66 GHz CPU, '#': measured on a PC with 2 GHz CPU, '△': measured on a PC with 2.83 GHz CPU).

Algorithms	Segmentation						Co-segmentation		
	Ours	WcSeg	RandCuts	NormCuts	RandWalks	Kmeans	Ours	Hu et al. (2012)	Wu et al. (2013)
Time(s)	216.3^{+}	93.7^{+}	83.8$^{\#}$	49.4$^{\#}$	1.4$^{\#}$	2.5$^{\#}$	195.9^{+}	20.6^{+}	24.2$^{\triangle}$

6.3. Performance

We implemented the proposed algorithm in Matlab and C++. The computation time costs of the segmentation and co-segmentation algorithms are shown in Table 4. In average, our algorithm takes more than 3 minutes to process a shape, where the time cost for computing low-level features is about 11 seconds and that for computing high-level features and SAEs is about 3 minutes and 10 seconds. However, our current implementation, un-optimized yet, can be further speeded up if the parallel implementation technique is considered into the overall framework.

7. Limitation and future work

First, we can't automatically distinguish wanted features and unwanted features in the learning stage. Therefore, even if there are sufficiently many low-level geometric features, we can't guarantee that the segmentation results are definitely better. We need to propose a better strategy in the future.

Second, designing a desirable structure of the SAEs is non-trivial. We will conduct more experiments to obtain a general configuration such that the algorithm framework can truly work well across various model libraries and various basic shape signature combinations.

Finally, the required computation time, especially that spent training the deep neural network and generating the high-level feature space, is still too long. In the future, we will seek a parallelized or distributed implementation, which will greatly reduce the time costs and facilitate its use in practice.

8. Conclusion

In this paper, we propose a novel unsupervised method for segmenting and co-segmenting 3D shapes. After over-segmenting the shapes into primitive patches, we generate the high-level features from the low-level features of each patch by using deep learning. We finally use high-level features for segmenting a single shape or co-segmenting a group of shapes from the same family. We validate our method on two open datasets (PSB and COSEG), and make extensive comparisons with the state-of-the-art approaches on this problem. The experimental results demonstrate that our method can achieve desirable results.

Acknowledgements

Many thanks to the anonymous reviewers for their valuable comments and suggestions. We would also like to thank Oliver van Kaick, Zizhao Wu, Yunhai Wang and Ruizhen Hu for their kind help. This work is supported by National Natural Science Foundation of China (11226328, 61222206, 11526212, 61300168, 61273332), Natural Science Foundation of Zhejiang Province (LY13F020018), Opening Foundation of Zhejiang Provincial Top Key Discipline (XKXL1406), Natural Science Foundation of Ningbo City Grant (2015A610123), One Hundred Talent Project of the Chinese Academy of Sciences (Ligang-Liu), and Ningbo Sc. & Tech. (Innovation Team) Plan Project (2014B82015).

References

Attene, M., Katz, S., Mortara, M., Patane, G., Spagnuolo, M., Tal, A., 2006. Mesh segmentation – a comparative study. In: IEEE International Conference on Shape Modeling and Applications. 2006, Matsushima, p. 7.
Au, O.-C., Zheng, Y., Chen, M., Xu, P., Tai, C.-L., 2012. Mesh segmentation with concavity-aware fields. IEEE Trans. Vis. Comput. Graph. 18 (7), 1125–1134.
Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., 2007. Greedy layer-wise training of deep networks. Adv. Neural Inf. Process. Syst. 19, 153–160.
Boykov, Y., Veksler, O., Zabih, R., 2001. Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. 23 (11), 1222–1239.
Bronstein, M.M., Kokkinos, I., 2010. Scale-invariant heat kernel signatures for non-rigid shape recognition. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1704–1711.
Chen, X., Golovinskiy, A., Funkhouser, T., 2009. A benchmark for 3D mesh segmentation. ACM Trans. Graph. 28 (3), 1–12.
Dietterich, T., 1995. Overfitting and undercomputing in machine learning. ACM Comput. Surv. 27 (3), 326–327.
Gal, R., Cohen-Or, D., 2006. Salient geometric features for partial shape matching and similarity. ACM Trans. Graph. 25 (1), 130–150.
Gatzke, T., Grimm, C., Garland, M., Zelinka, S., 2005. Curvature maps for local shape comparison. In: International Conference on Shape Modeling and Applications, pp. 244–253.
Golovinskiy, A., Funkhouser, T., 2008. Randomized cuts for 3D mesh analysis. ACM Trans. Graph. 27 (5), 1–12.
Golovinskiy, A., Funkhouser, T., 2009. Consistent segmentation of 3D models. Comput. Graph. 33 (3), 262–269.
Hinton, G., Osindero, S., Teh, Y.-W., 2006. A fast learning algorithm for deep belief nets. Neural Comput. 18 (7), 1527–1554.
Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.
Hong, B.-W., Soatto, S., 2015. Shape matching using multiscale integral invariants. IEEE Trans. Pattern Anal. Mach. Intell. 37 (1), 151–160.
Hu, R., Fan, L., Liu, L., 2012. Co-segmentation of 3D shapes via subspace clustering. Comput. Graph. Forum 31 (5), 1703–1713.
Huang, Q., Koltun, V., Guibas, L., 2011. Joint shape segmentation with linear programming. ACM Trans. Graph. 30 (6), 1–12.
Kaick, O.V., Fish, N., Kleiman, Y., Asafi, S., Cohen-Or, D., 2014. Shape segmentation by approximate convexity analysis. ACM Trans. Graph. 34 (1), 1–11.
Kalogerakis, E., Hertzmann, A., Singh, K., 2010. Learning 3D mesh segmentation and labeling. ACM Trans. Graph. 29 (4), 1–12.
Katz, S., Leifman, G., Tal, A., 2005. Mesh segmentation using feature point and core extraction. Vis. Comput. 21 (8), 649–658.
Katz, S., Tal, A., 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans. Graph. 22 (3), 954–961.
Knopp, J., Prasad, M., Van Gool, L., 2010. Orientation invariant 3D object classification using hough transform based methods. In: Proceedings of the ACM Workshop on 3D Object Retrieval. ACM, pp. 15–20.
Lai, Y.K., Hu, S.M., Martin, R.R., Rosin, P.L., 2008. Fast mesh segmentation using random walks. In: Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling, pp. 183–191.
Loffler, J., 2000. Content-based retrieval of 3D models in distributed web databases by visual shape information. In: Proceedings of the IEEE International Conference on Information Visualization, pp. 82–87.
Meng, M., Xia, J., Luo, J., He, Y., 2013. Unsupervised co-segmentation for 3D shapes using iterative multi-label optimization. Comput. Aided Des. 45 (2), 312–320.
Osada, R., Funkhouser, T., Chazelle, B., Dobkin, D., 2002. Shape distributions. ACM Trans. Graph. 21 (4), 807–832.
Patel, A.B., Nguyen, T., Baraniuk, R.G., 2015. A probabilistic theory of deep learning. arXiv:1504.00641.
Perez-Cruz, F., 2008. Kullback–Leibler divergence estimation of continuous distributions. In: IEEE International Symposium on Information Theory, pp. 1666–1670.

Ren, X., Malik, J., 2003. Learning a classification model for segmentation. In: Proceedings of the 9th IEEE International Conference on Computer Vision, pp. 10–17.

Rong, L., Hao, Z., 2004. Segmentation of 3D meshes through spectral clustering. In: Proceedings of the 12th Pacific Conference on Computer Graphics and Applications, pp. 298–305.

Rummelhart, D.E., 1986. Learning representations by back-propagation errors. Nature 323, 533–536.

Shamir, A., 2008. A survey on mesh segmentation techniques. Comput. Graph. Forum 27 (6), 1539–1556.

Shapira, L., Shalom, S., Shamir, A., Cohen-Or, D., Zhang, H., 2010. Contextual part analogies in 3D objects. Int. J. Comput. Vis. 89 (2), 309–326.

Shapira, L., Shamir, A., Cohen-Or, D., 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis. Comput. 24 (4), 249–259.

Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 22 (8), 888–905.

Shlafman, S., Tal, A., Katz, S., 2002. Metamorphosis of polyhedral surfaces using decomposition. Comput. Graph. Forum 21 (3), 219–228.

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

Smeets, D., Keustermans, J., Vandermeulen, D., Suetens, P., 2013. meshsift: local surface features for 3D face recognition under expression variations and partial data. Comput. Vis. Image Underst. 117 (2), 158–169.

van Kaick, O., Tagliasacchi, A., Sidi, O., Zhang, H., Cohen-Or, D., Wolf, L., Hamarneh, G., 2011. Prior knowledge for part correspondence. Comput. Graph. Forum 30 (2), 553–562.

Veltkamp, R.C., Hagedoorn, M., 2001. In: Lew, M.S. (Ed.), Principles of Visual Information Retrieval. Springer-Verlag, London, UK, pp. 87–119. Ch. State of the art in shape matching.

Vranic, D.V., 2003. An improvement of rotation invariant 3D-shape based on functions on concentric spheres. In: IEEE International Conference on Image Processing, vol. 3, pp. 757–760.

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

Wu, Z., Wang, Y., Shou, R., Chen, B., Liu, X., 2013. Unsupervised co-segmentation of 3D shapes via affinity aggregation spectral clustering. Comput. Graph. 37 (6), 628–637.

Xie, Z., Xu, K., Liu, L., Xiong, Y., 2014. 3D shape segmentation and labeling via extreme learning machine. Comput. Graph. Forum 33 (5), 85–95.

Xu, K., Li, H., Zhang, H., Cohen-Or, D., Xiong, Y., Cheng, Z.-Q., 2010. Style-content separation by anisotropic part scales. ACM Trans. Graph. 29 (6), 1–10.

Zheng, Y., Cohen-Or, D., Averkiou, M., Mitra, N.J., 2014. Recurring part arrangements in shape collections. Comput. Graph. Forum 33 (2), 115–124.