Theme Article

Active 3-D Shape Cosegmentation With Graph Convolutional Networks

Zizhao Wu
Hangzhou Dianzi University

Ming Zeng
Xiamen University

Feiwei Qin
Hangzhou Dianzi University

Yigang Wang
Hangzhou Dianzi University

Jiří Kosinka
University of Groningen

***Abstract*—We present a novel active learning approach for shape cosegmentation based on graph convolutional networks (GCNs). The premise of our approach is to represent the collections of three-dimensional shapes as graph-structured data, where each node in the graph corresponds to a primitive patch of an oversegmented shape, and is associated with a representation initialized by extracting features. Then, the GCN operates directly on the graph to update the representation of each node based on a layer-wise propagation rule, which aggregates information from its neighbors, and predicts the labels for unlabeled nodes. Additionally, we further suggest an active learning strategy that queries the most informative samples to extend the initial training samples of GCN to generate more accurate predictions of our method. Our experimental results on the Shape COSEG dataset demonstrate the effectiveness of our approach.**

Segmentation of three-dimensional (3-D) shapes is a fundamental operation in geometric modeling and shape analysis. Recently, researchers have observed that by segmenting a set of 3-D shapes as a whole into consistent parts one can infer more knowledge than from an individual shape. This is the problem of cosegmentation. The results of shape cosegmentation can be applied to various applications in computer graphics and vision, such as modeling, shape retrieval, texturing, etc.

Digital Object Identifier 10.1109/MCG.2019.2891634
Date of publication 14 January 2019; date of current version 22 March 2019.

Published by the IEEE Computer Society

Most of the existing cosegmentation approaches[1–5] cast the cosegmentation problem as the clustering problem, where one of the key ingredients is to devise robust descriptors that can characterize different parts of shapes appropriately. For example, parts belonging to the same category end up close to each other in descriptor space, while they are distant if they belong to different categories. However, since different features characterize different aspects of shapes, it is not a trivial task to directly infer the semantic information from their feature descriptors of their parts. Recently, deep learning methods, especially deep convolution neural networks (CNNs), have been widely used in various computer vision tasks with great success. Many researchers have tried to introduce deep learning techniques to facilitate various 3-D shape analysis tasks. However, it is nontrivial to adapt a CNN designed for images to 3-D shapes, particularly to 3-D shapes modeled by irregular triangle meshes or point clouds with no regular neighborhoods like those available in images. To address this problem, many researchers have devoted their efforts to various modifications of CNNs, such as voxel-based CNNs,[6] view-based CNNs,[7] and graph-based CNNs,[8] demonstrating the great potential of these methods.

Following this trend, we present a novel shape cosegmentation method based on the graph convolutional network (GCN) by casting the cosegmentation problem as a graph learning problem. Specifically, we first decompose the input shapes into primitive patches, which are associated with representations initialized by extracting features. We then build a graph whose nodes are these primitive patches with feature representations, and edges are constructed by searching the nearest neighbors in feature space. Finally, we employ the GCN to dynamically update the hidden representation of each node by using a layer-wise propagation rule, and infer the class labels for all the unlabeled nodes in the network.

It is difficult to achieve (near) error-free results in the one-shot process of the GCN, particularity in the case when the initial labeled samples are sparse. To this end, we suggest an active learning (AL) technique to leverage the abundant unlabeled samples with a few labeled samples to generate a strong hypothesis. Our AL technique is devised to query the most informative samples in the unlabeled samples to extend the initial training dataset. The query of the AL process is raised at the end of every epoch of the GCN training process. As the process progresses, the GCN will generate more accurate predictions as more informative labeled nodes are provided for training. By iteratively performing these improvement steps, our algorithm eventually achieves nearly error-free results.

Comparing with the state-of-the-art AL approaches[1,2] for shape cosegmentation, our approach has several advantages. First, the GCN in our approach jointly performs feature selection and classification, instead of performing these separately.[1] Second, the graph representation in our network is dynamically adjusted, rather than relying on the fixed graph structure of the label propagation (LP) approach,[2] making our algorithm more efficient. Our main contribution is the introduction of the graph neural network architecture for shape cosegmentation. Further, we exploit an AL strategy to alleviate the bottleneck of training samples. Our experimental results on a benchmark dataset demonstrate the effectiveness and utility of our approach.

RELATED WORK

In this section, we give a brief review of existing work on shape cosegmentation, CNNs for 3-D shape analysis, and AL.

Shape Cosegmentation

In recent years, many techniques[1,2] have been proposed for cosegmentation of collections of 3-D shapes. Their goal is to simultaneously segment a set of shapes into meaningful parts, and build correspondences among them. For example, Golovinskiy and Funkhouser[3] obtain a consistent segmentation of a set by aligning all input shapes and clustering their primitives. Sidi *et al.*[4] have introduced a descriptor-based method for shape cosegmentation, relying on multiple feature descriptors in order to characterize the differences among primitive patches. Wu *et al.*[5] presented a multiple feature fusion approach for shape cosegmentation based on the affinity aggregation spectral clustering. Shu *et al.*[9] suggested an

unsupervised approach to learn the high-level features defined on the patches based on deep learning. Generally, these approaches can achieve promising performance. However, due to their limited low-level feature space and variety, these methods often struggle with largely varying datasets.

To alleviate this problem, supervised approaches have been proposed, which treat segmentation as a labeling problem. For example, Kalogerakis *et al.*[10] employ traditional machine learning techniques and construct classifiers to predict class labels. Inspired by the great success of CNNs in computer vision tasks, Guo *et al.*[11] introduce a deep learning framework that uses convolutional neural networks to predict labels. Both of these methods are based on geometry features and carefully labeled training data. Subsequent techniques include PointNet,[12] which directly takes points as input and outputs class labels. Kalogerakis *et al.*[7] combine image-based fully convolutional networks and surface-based conditional random fields to yield coherent segmentations of 3-D shapes. We note that the performance of supervised techniques greatly depends on the availability of high-quality training data that covers all shape variations. However, currently, such datasets are assembled manually and are limited in scale. Additionally, the training cost of supervised approaches is expensive.

Semisupervised learning (SSL) methods can be seen as a hybrid between the above two schemes, where the classification and analysis are performed based on both the labeled data and the structure of the unlabeled data. Wang *et al.*[1] suggest an AL method with the aid of constrained clustering, where the user can actively assist in the cosegmentation process by assigning pairwise constraints. However, the pairwise constraints in their algorithm are not expressive for user interaction. Wu *et al.*[2] introduce an optimized approach to improve interactive efficiency through LP. However, their method is graph based and the graph representation is fixed during the learning process, which means that the approach heavily relies on the initial graph construction. Further, in cases with noise in the graph, extensive and tedious human labeling efforts are required. We overcome these problems by introducing a GCN-based approach. Our approach jointly performs feature selection and label prediction for cosegmentation, and the graph representation is dynamically optimized by the GCN during the feature learning procedure.

CNNs for 3-D Shape Analysis

Despite its success in the setting of 2-D image data, the application of CNNs to 3-D shape analysis has been hindered due to that fact that 3-D shapes do not possess the regular neighborhood structures present in 2-D images. As a consequence, many researchers have devoted their efforts to various adaptation studies of the CNNs. Existing CNN structures used for 3-D shape analysis usually fall into three types: voxel-based CNNs,[6] view-based CNNs,[7] and graph-based CNNs.[8]

The voxel-based CNN[6] applied 3-D convolutional neural networks on voxelized shapes for 3-D shape analysis, shape retrieval, and shape classification. However, the computation cost of the 3-D convolution is expensive. View-based CNNs[7] have tried to represent the 3-D shapes as rendered 2-D images, and then make predictions based on image CNNs. However, this method loses the geometric details of the input meshes. Graph-based CNNs were first introduced by Gori *et al.*[14] Kipf and Welling[8] further extended the graph-based CNN model to the SSL setting and achieved very promising results in their experiments. In this paper, we also draw inspiration from the semisupervised GCN model,[3] and adapt the GCN model to facilitate shape cosegmentation.

SSL and AL

In many learning-based applications, training effective classifiers requires adequate labeled examples. However, labeling samples is expensive as it requires massive human efforts and expertise. To reduce or even eliminate labeling efforts, two learning strategies have been widely adopted. The first is AL, which aims to minimize the number of queries for building a strong learner by selecting the most informative samples for labeling from a large pool of unlabeled samples. The second is SSL, which tries to exploit the availability of unlabeled data for model training and improvement.

Recent studies further investigate the combination of SSL and AL.[13,14] They have showed that their combination yields better results than either SSL or AL alone using the same number of

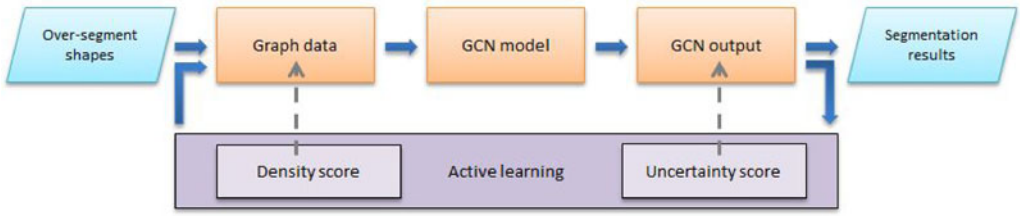

Figure 1. Pipeline of our method. First, each shape is oversegmented into primitive patches. Then, a graph is constructed where each node denotes a primitive patch. After that, the GCN model is employed to dynamically update the graph representation to make label predictions for the unlabeled nodes of the graph. In addition, we further suggest an AL strategy to query the most informative samples for labeling in order to expand the initial training set of the GCN. We iterate the GCN and AL until we achieve near error-free results.

labeled samples. Building on recent work on AL,[15,16] we propose an AL strategy to boost the performance of GCNs by querying the most informative samples among the unlabeled sample set.

OVERVIEW

The purpose of our algorithm is to segment and label a set of 3-D shapes into meaningful parts simultaneously. The input to our system is a set of shapes in the same category (e.g., "airplanes") and a label of interest (e.g., "wing"), and the output is a per-point label that indicates which part the point belongs to. In addition, our algorithm is devised to embrace supervision by the users to achieve accurate results.

The full algorithm consists of four stages (see Figure 1). First, each shape is oversegmented into primitive patches independently. Then, we extract feature vectors for each patch, followed with the *k*-nearest-neighbors (KNN) search in feature space to construct a graph where each node denotes a primitive patch and each edge reflects a relationship between two nodes. After that, the GCN model is used to dynamically update the graph representation while making a prediction for the unlabeled nodes of the graph. Since prediction accuracy relies on the labeled samples, our method employs an AL strategy to query the most informative samples for labeling. We iterate the last two steps till close to error-free results are achieved. Each of the stages is now described in turn.

Oversegmentation

Inspired by the super-pixel method in image analysis and by previous work on shape (co) segmentation,[1–5] we first decompose each shape into primitive patches to generate an oversegmentation. To this end, we employ normalized cuts to compute the primitive patches for each shape. In our experiments, the number of patches per shape is set to 30. Figure 2 gives an example of our oversegmentation results.

Graph Construction

Since our method is graph based, we first need to construct a graph based on the patches. Specifically, aiming for characterizing the primitive patches of shapes, we prefer to collect a set of shape descriptors by exploiting geometric

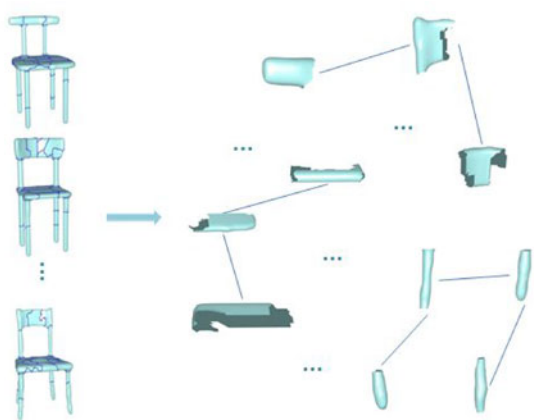

Figure 2. Example of shape oversegmentation and the graph construction. Given a category with 20 shapes, we oversegment each shape to 30 patches (left), resulting into 600 patches for the category. By extracting features defined on the patches, a graph is constructed where each node denotes a patch, and each edge represents a KNN relationship between nodes in feature space (right).

features. Following previous approaches,[1–5] we select five robust and informative shape descriptors: shape diameter function (SDF), conformal factor (CF), shape contexts (SC), average geodesic distance (AGD), and the Geodesic distance to the Base of the shape (GB). For descriptors with multiscale properties (such as SC), we simply sample and concatenate them. These features capture different properties of each triangle. Then, for each feature descriptor, we compute a histogram to capture its distribution for all triangles in each patch. We then accomplish our graph construction by using both the information from the feature vectors of the patches and the relationships between the patches.

Specifically, let p_i denote the *i*th patch and $h_{k,i}$ be the histogram for p_i of the *k*th descriptor, $k = 1, 2, \ldots, 5$. We concatenate the histograms of each patch to form the feature vector $x_i = [h_{1,i}, h_{2,i}, \ldots, h_{5,i}]$ for the *i*th patch. After that, we compute the dissimilarities between the pairs of patches based on the feature vectors $X = (x_1, x_2, \ldots, x_n)$ defined on the patches, where n is the total number of primitive patches. The dissimilarity between p_i and p_j is defined as $d(p_i, p_j) = \mathrm{EMD}(x_i, x_j)$, where $\mathrm{EMD}(x, y)$ is the earth-mover's distance between x and y. By applying the commonly used Gaussian kernel function on the distance, we obtain the similarity distance between p_i and p_j

$$a_{i,j} = \exp\left(-d\left(p_i, p_j\right)/2\sigma^2\right). \tag{1}$$

These constitute the elements of the similarity matrix A defined for all pairs of patches. We then optimize A through a KNN search based on the assumption that the local similarities are more reliable than the more distant ones. In our implementation, we set the number of bins to be 50 for each of the histograms and σ is set to be the mean of all distances.

Our graph is constructed based on the feature set X and the similarity matrix A. Without loss of generality, let our graph be represented as $G(V, E)$ with n nodes $v_i \in V$ and edges $(v_i, v_j) \in E$, where V is the set of nodes and E is the set of edges of G. We associate the feature set X to V, seen as node attributes, and the similarity matrix A is associated with the edge set E to complete the graph definition.

GCN for Shape Cosegmentation

Having defined the graph representation, we now introduce the GCN algorithm for semisupervised classification of the nodes in the graph. The GCN combines graph structures and vertex features, where the features of unlabeled nodes are mixed with those of nearby labeled nodes, and propagated over the graph through multiple layers with a layer-wise propagation rule. The prediction of labels for all unlabeled nodes is also achieved in the network, which depicts the cosegmentation results for the input shapes. This is explained in detail below.

AL on GCN

Since the training of the GCN still requires considerable amount of labeled data to make an accurate prediction, we propose an AL strategy to help to extend the training samples of the network. Our AL method selects the most informative samples for querying and iteratively appends them to the labeled set. Two popular AL query criteria are adopted here: uncertainty and representativeness. These are computed by calculating the information entropy score and the information density score, respectively. We then continue to train the GCN with the expanded label set, using the pretrained GCN as initialization.

As the process progresses, the GCN will generate more and more accurate predictions as more informative labeled nodes are provided for training. This is elaborated upon below.

GRAPH CONVOLUTIONAL NETWORK

Following the GCN definition,[8] a layer-wise propagation rule in the convolutional layers can be defined as

$$H^{(l+1)} = \emptyset\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2}$$

where $\tilde{A} = A + I$ is the adjacency matrix of the graph G with added self-connections via the identity matrix I, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and W^l is a layer-specific trainable weight matrix. $\emptyset(\cdot)$ denotes an activation function, which we define as the RELU function: $\mathrm{RELU}(\cdot) = \max(0, \cdot)$. $H^l \in R^{N \times D}$ is the matrix of activations in the *l*th layer with the initial status $H^0 = X$defined in the input layer.

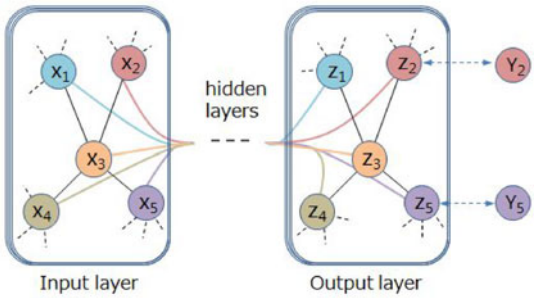

Figure 3. GCN for SSL. The input to the GCN is a graph with its feature representation. The GCN dynamically updates the hidden representation of each node by employing a layer-wise propagation rule, and finally infers the class labels for all unlabeled nodes in the output layer of the network.

Based on the convolutional operations, the GCN model is applied for semisupervised classification, as illustrated in Figure 3. Similar to the work proposed by Kipf and Welling,[8] we use a two-layer GCN, which applies a softmax classifier for semisupervised node classification on the graph

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\mathrm{RELU}\Big(\hat{A}XW^{(0)}\Big)W^{l}\Big) \tag{3}$$

where $W^{(0)}$ is the input-to-hidden weight matrix, W^{l} is the hidden-to-output matrix, $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, and $\mathrm{softmax}(x_i) = \exp(x_i)/z$ with $z = \sum_i \exp(x_i)$. The loss function is defined as the cross-entropy error over all labeled samples

$$L = -\sum_{l \in y_l}\sum_{f=l}^{F} Y_{lf} \ln \; Z_{lf} \tag{4}$$

where y_l is the set of indices of labeled nodes. As the entire network is designed to be differentiable end-to-end, the parameters $W^{(0)}$ and W^{l} can be trained with gradient-based optimization, and the class labels for unlabeled nodes can also be obtained based on the classifier of the GCN.

AL ON THE GCN

Similar to other kinds of convolutional neural networks, GCNs also need a set of labeled data for training and model selection. One way to boost the performance of a learning approach is via AL, which selects the most informative unlabeled samples for querying. Drawing on inspiration from recent work on AL,[16,17] we chose two widely adopted AL query strategies to actively select the samples for labeling: uncertainty sampling and density sampling (also called representativeness sampling).

Uncertainty sampling aims to choose the most uncertain instance to label. We employ information entropy as our uncertainty measure. The information entropy of a node n_i is defined as

$$\emptyset_u(n_i) = -\sum_{c=1}^{C} P(Y_i c = 1|G, L, X)\log P(Y_i c = 1|G, L, X), \tag{5}$$

where $P(Y_i c = 1|G, L, X)$ is the probability of node n_i belonging to class c computed by the GCN. A visualization of the uncertainty score is shown on an example in Figure 4(a).

Since uncertainty sampling may suffer from the problem of selecting outliers, density

Figure 4. Our AL criteria. (a) Uncertainty criterion. (b) Density criterion. The larger the embedding area of a sample, the higher its score. The squares denote the labeled samples. t-SNE[20] was applied to visualize the learned feature embedding. Our AL method selects the most informative unlabeled samples for querying by combining the two sampling criteria.

(a) Feature embedding with 8 samples (b) Feature embedding with 16 samples.

Figure 5. Visualization of the learning procedure of our GCN algorithm. Again, we adopted the t-SNE[20] approach to visualize the learned features of our graph. The squares denote the labeled samples. (a) Feature embedding with 8 samples (b) Feature embedding with 16 samples.

sampling is an effective solution to overcome this. The density of a sample is evaluated based on how many samples it is similar to: we compute the density score for each node, and the nodes with a high density score are selected.

To obtain the density score, we first employ the traditional cosine measure to estimate the similarity between two nodes

$$\cos\left(x_i, x_j\right) = x_i \cdot x_j / \left(\|x_i\| \cdot \left\|x_j\right\|\right) \tag{6}$$

where x_i and x_j are the feature vectors of nodes n_i and n_j, respectively. Practically, it would be inefficient to exhaustively calculate similarities between all pairs of samples as the unlabeled corpus is often very large. Similarly as used by Yu *et al.*,[16] we utilize the KNN-based density to measure the density value by estimating the average similarity between each node and its k most similar nodes. Mathematically, given a set of k most similar examples $S(n_i) = \{s_1, s_2, \ldots, s_k\}$ of an unlabeled sample n_i, the average similarity, or the density value, can be computed as

$$\emptyset_d(n_i) = \sum\nolimits_{s_i \in S(n_i)} \cos\left(n_i, s_i\right)/K. \tag{7}$$

A visualization of the density score is shown an an example in Figure 4(b).

We combine the uncertainty and density scores to form our AL criterion, which is defined to be their product

$$\emptyset(n_i) = \emptyset_u(n_i) * \emptyset_d(n_i). \tag{8}$$

The motivation behind this model is to use the density score to adjust the uncertainty score of an unlabeled sample n_i. This means that our AL model favors sample with high uncertainty and high density at each step of our learning cycle.

RESULTS AND COMPARISON

We now present the results obtained with our AL approach, compare our method to prior art, and discuss relevant implementation details.

Results and Accuracy

In this section, we conduct experiments on the COSEG dataset[11] and the PSB dataset.[19] Both of these are widely used datasets for evaluating mesh segmentation algorithms.

As used by Yi *et al.*,[13] we also employ the amount of supervision and accuracy to evaluate our method. The accuracy is computed as

$$\text{Accuracy} = \sum_{t \in T} a_t g_t(l_t) / \sum_{t \in T} a_t \tag{9}$$

where a_t is the area of the triangle $t \in T$, l_t is the predicted segmentation part, and $g_t(l_t)$ denotes the l_tth component of g_t, which equals to 1 if the prediction is correct.

We used our system to iteratively annotate the COSEG benchmark dataset. Our system selects the most informative sample for querying the annotation at each step of the iterative learning process, and the GCN model then propagates the label information to unlabeled samples while learning their high-level representations. This process proceeds until we obtain close to error-free results. Figure 5 shows an example in which the results improve as more training samples are provided.

Table 1. Summary of the annotation and verification scores for the varied COSEG dataset, where Num. denotes the number of shapes, Anno. denotes the number of annotations, and Acc., Pre., and Rec. stand for the accuracy, the precision, and the recall scores of the evaluation results, respectively.

Category	Num.	Anno.	Acc.	Pre.	Rec.
Candelabra	28	21	98.73	97.72	95.67
Chairs	20	16	98.21	97.78	96.95
Goblets	12	16	98.54	98.36	97.56
Guitars	44	26	95.54	96.10	90.24
Lamps	20	23	95.68	95.34	96.21
Vases	28	18	93.35	91.68	90.46
Irons	18	25	94.64	92.35	92.85
Large chairs	400	95	95.56	92.68	89.43
Large vases	300	102	94.66	92.35	92.31
Large tele-aliens	200	97	96.64	95.88	91.52

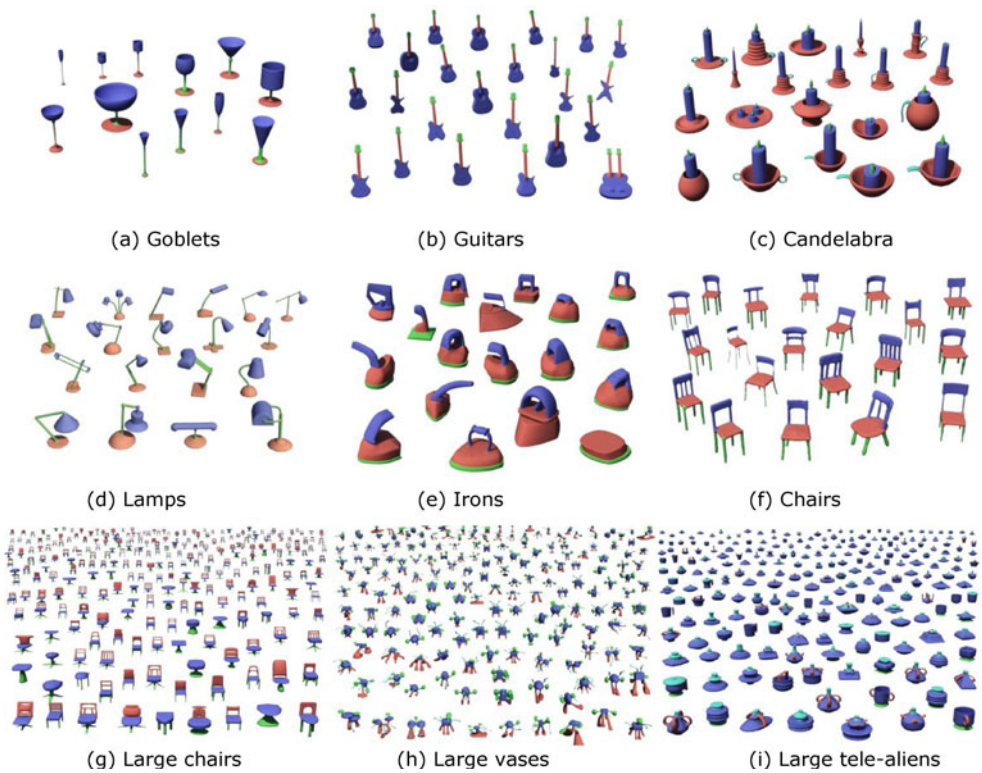

Figure 6. Our cosegmentation results on several representative categories from the Shape COSEG dataset. Corresponding segments are shown in the same color. (a) Goblets. (b) Guitars. (c) Candelabra. (d) Lamps. (e) Irons. (f) Chairs. (g) Large chairs. (h) Large vases. (i) Large telealiens.

Based on our extensive experiment, we report the human annotation and the verification (accuracy, precision, and recall) scores of our algorithm in Table 1. As can be seen in this table, our approach is able to obtain above 95% average accuracy among several categories of shapes. Our cosegmentation results are shown in Figure 6.

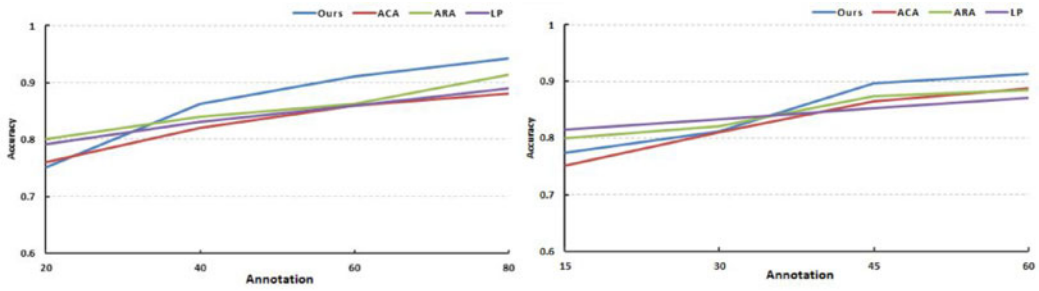

Figure 7. We compare our approach to the ACA method,[1] the LP method,[2] and the ARA method[13] and on the large chairs dataset and the large vases dataset. The *x*-axis is human annotation and the *y*-axis is the measured accuracy score. Note that our method generates more and more accurate results than the other methods as more training data are given. (a) Large chairs. (b) Large vases.

Comparison

We compare our approach to several state-of-the-art AL techniques. These include the active coanalysis (ACA) method proposed by Wang *et al.*,[1] the LP method,[2] and the active region annotation (ARA) method proposed by Yi *et al.*[13] We consider four different label rates, 5%, 10%, 15%, and 20%, per class in the training set in our evaluation. We note that ACA achieves the segmentation results based on a clustering strategy with pairwise constraints, which is designed to iteratively query users to provide "must-link" and "cannot-link" relationships between oversegment patches. In contrast, our approach requires users to label directly on patches, which is more time efficient. Our quantitative experimental results are shown in Figure 7.

Comparing to ARA, we note that ARA takes advantage of global shape-to-shape similarities, local feature similarities, and point-to-point correspondences in order to learn similarities between patches. Instead, due to the strong ability of representation learning of the GCN, our method requires fewer hand-crafted settings, which makes our approach more general and more widely applicable. Our evaluation results suggest that our approach has superior performance over ARA on several categories of shapes, as shown in Figure 7.

We also compare our method to the semisupervised labeling method of Wu *et al.*[2] We test their method on the COSEG dataset, and used the average accuracy score to make a comparison. We note that our approach achieves superior performance than theirs (see Figure 7). This was to be expected as our method iteratively selects the best informative samples for querying rather than labeling them blindly. Our approach thus greatly reduces redundancy among labeled samples.

Finally, we compare our approach to several supervised learning algorithms,[7,10,11] which range from a classical learning based approach[10] to deep learning based approaches.[7,11] For each category, N ($N = 6$, $N = 12$, $N = 19$) shapes are randomly selected for training and the remaining meshes are used for testing. Theoretically, either the supervised approach or the semisupervised approach can produce close to error-free results once sufficient training data are given. As a result, the larger the training data, the higher the accuracy score. Compared to supervised learning algorithms, our approach outperforms them on most of the categories while requiring less training data.

Taken the $N = 12$ case as an example, where the training data are approximately 60% of all data for a category, our method requires less than 5% (30 annotated patches over 600 patches of a category) training data while still achieving comparable results, as summarized in Table 2. Additionally, the supervised learning based approaches require several hours to train the classifier (8 h,[10] 4 h,[11] and 8 h[7]). In contrast, our method spends only several seconds on training a graph with 600 nodes (or 1200 nodes for 60 patches) for 200 epochs of a category. This is a strong advantage of our

Table 2. Segmentation accuracy on the PSB dataset, which contains 19 categories (20 models per category), summarizing a comparison of our approach to those given by Kalogerakis *et al.*[7,10] and by Guo *et al.*[11]. SB12 denotes 12 shapes chosen for training, and the remaining shapes were used for testing in each category. Note that our method achieves competitive results.

	SB12[10]	SB12[11]	SB12[7]	Ours ~5%		SB12[10]	SB12[11]	SB12[7]	Ours ~5%
Human	93.20%	91.22%	94.5%	93.32%	Plier	96.20%	96.22%	95.5%	95.42%
Cup	99.60%	99.73%	93.8%	92.75%	Fish	95.60%	95.64%	96.0%	94.85%
Glasses	97.20%	97.60%	96.6%	97.11%	Bird	87.90%	88.35%	88.5%	87.41%
Airplane	96.10%	96.67%	93.0%	94.84%	Armadillo	90.10%	92.27%	92.8%	92.32%
Ant	98.80%	98.80%	98.6%	96.43%	Bust	62.10%	69.84%	68.4%	**76.12%**
Chair	98.40%	98.67%	98.5%	98.21%	Mech	90.50%	95.60%	98.7%	92.57%
Octopus	98.40%	98.79%	98.3%	96.84%	Bearing	86.60%	92.46%	92.3%	**93.67%**
Table	99.30%	99.55%	99.5%	99.26%	Vase	85.80%	89.11%	86.8%	**90.10%**
Teddy	98.10%	98.24%	97.7%	97.78%	Fourleg	86.20%	87.02%	85.0%	**89.46%**
Hand	88.70%	88.71%	84.8%	**90.14%**					

approach when compared to supervised learning based algorithms.

Implementation Details

We use the same hyperparameters as used by Kipf and Welling[8] to initialize the GCN model, where the initial learning rate is set to 0.01, the maximum number of epochs is set to 200, and the dropout rate is set to 0.5. We follow the work[8] to set the number of convolutional layers to 2, and the hidden layer size is set to 32.

As for computation cost, we note that the computational complexity of the GCN is linear in the number of graph edges, as discussed by Kipf and Welling.[8] We evaluated the performance of our approach on a computer with Intel Core i7 processor, 8 GB of RAM, and Nvidia GeForce GTX1080. Given a graph with 600 nodes (20 shapes, 30 patches per-shape), 3600 edges, our method spends less than 0.1 s for each epoch. Note that the computational cost can be further sped up using FastGCN, so the scalability of our method is not an issue.

To evaluate the impact of the graph connectivity on our algorithm, we added noise to our graph by randomly changing the graph connectivity and modifying the number of edges. In our experiment, the convergence rates changed only slightly, but did not significantly affect the results. The connectivity affects the matrix A in (3), but the GCN learns W, which still ensures that the method converges. However, an unreasonable change in the graph connectivity will of course cause the GCN to fail to converge. Regarding the edge count in the graph, the GCN model converges to a better solution by taking into account longer-range information in the graph (achieved by increasing the KNN value), and the computational cost increases linearly.

It is important to emphasize that our pipeline is not very sensitive to the initial patch number of the graph construction, as demonstrated in Figure 8. In our experiment, we separately oversegmented the shapes into 30 patches and 60 patches, and also extracted feature descriptors. As can be seen in the figure, patches belonging to the same part typically have similar values, and the obtained histograms are very close to each other. In our experiment, the patch number only affected the result very slightly, showing that our approach is robust with respect to the parameter specifying the number of patches. Consequently, we recommend 30 as the default value. A smaller value (e.g., 10) will merge some parts that need to remain separate, and although a larger value is in principle better, it will unnecessarily increase computation time without producing significantly better results.

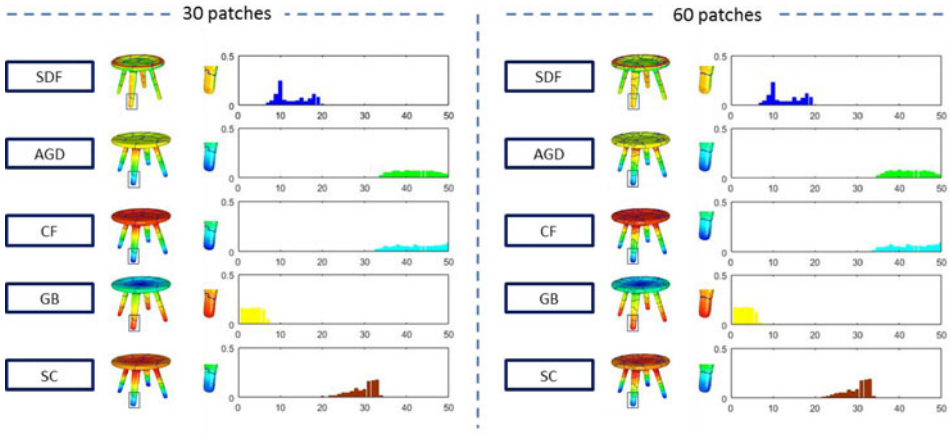

Figure 8. We show the impact of the number of patches (30 patches on the left versus 60 patched on the right) used in oversegmentation on the feature values. SDF stands for shape diameter function, AGD for average geodesic distance, CF for conformal factor, GB for geodesic distance to the base of the shape, and SC for shape contexts. Note that the histogram values based on all these metrics defined on the patches are robust to the patch number parameter.

CONCLUSION

We have presented an AL approach for shape cosegmentation. We regard the cosegmentation problem as a graph learning problem, where the GCN approach is leveraged to predict the class labels for the unlabeled samples through a layer-wise propagation rule defined in the network. Additionally, we have introduced an AL method that queries the most informative samples to extend the training samples of our method. We have validated the effectiveness and utility of our method on experimental results.

Our approach has several limitations which suggest avenues for future work. One such limitation is that our algorithm fails to deal with out-of-sample data, especially in the setting of dynamic graphs. And another limitation has to do with the fact that the similarity distances between the nodes in the graph are measured by means of concatenating multiple features and performing a KNN search. A more rigorous strategy to deal with multiple features could be expected to lead to even better results.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China (61602139, 61402387, 61502129), in part by the Open Project Program of State Key Lab of CAD&CG, Zhejiang University (A1803 and A1817), in part by the Guiding Project of Fujian Province (2018H0037), and in part by the Zhejiang Pro- vince Science and Technology Planning Project (2018C01030).

REFERENCES

1. Y. Wang, S. Asafi, O. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen, "Active co-analysis of a set of shapes," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 165–174, 2012.
2. Z. Wu, R. Shou, Y. Wang, and X. Liu, "Interactive shape co-segmentation via label propagation," *Comput. Graph.*, vol. 38, pp. 248–254, 2014.
3. A. Golovinskiy and T. A. Funkhouser, "Consistent segmentation of 3d models," *Comput. Graph.*, vol. 33, no. 3, pp. 262–269, 2009.
4. O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or, "Unsupervised co-segmentation of a set of shapes via descriptor space spectral clustering," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 126:1–126:9, 2011.
5. Z. Wu, Y. Wang, R. Shou, B. Chen, and X. Liu, "Unsupervised co-segmentation of 3D shapes via affinity aggregation spectral clustering," *Comput. Graph.*, vol. 37, no. 6, pp. 628–637, 2013.
6. P. Wang, Y. Liu, Y. Guo, C. Sun, and X. Tong, "O-CNN: Octree-based convolutional neural networks for 3d shape analysis," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 72:1–72:11, 2017.

7. E. Kalogerakis, M. Avekiou, S. Maji, and S. Chaudhuri, "3D shape segmentation with projective convolutional networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6630–6639.
8. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv:1609.02907.
9. Z. Shu *et al.*, "Unsupervised 3d shape segmentation and co-segmentation via deep learning," *Comput. Aided Geometric Des.*, vol. 43, pp. 39–52, 2016.
10. E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3D mesh segmentation and labeling," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 102:1–102:12, 2010.
11. K. Guo, D. Zou, and X. Chen, "3D mesh labeling via deep convolutional neural networks," *ACM Trans. Graph.*, vol. 35, no. 1, pp. 3:1–3:12, 2015.
12. C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.
13. L. Yi *et al.*, "A scalable active framework for region annotation in 3d shape collections," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 210:1–210:12, 2016.
14. M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. Int. Joint Conf. Neural Netw.*, 2005, vol. 2, pp. 729–734.
15. X. Wang, J. Wen, S. Alam, Z. Jiang, and Y. Wu, "Semi-supervised learning combining transductive support vector machine with active learning," *Neurocomputing*, vol. 173, pp. 1288–1298, 2016.
16. D. Yu, B. Varadarajan, L. Deng, and A. Acero, "Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion," *Comput. Speech Lang.*, vol. 24, no. 3, pp. 433–444, 2010.
17. J. Zhu, H. Wang, B. K. Tsou, and M. Y. Ma, "Active learning with sampling by uncertainty and density for data annotations," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 6, pp. 1323–1331, Aug. 2010.
18. H. Cai, V. W. Zheng, and K. C. Chang, "Active learning for graph embedding," arXiv:1705.05085.
19. X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3D mesh segmentation," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 73:1–73:12, 2009.
20. L. J. P. van der Maaten and G. E. Hinton, "Visualizing high-dimensional data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.

Zizhao Wu is an assistant professor with the School of Media and Design, Hangzhou Dianzi University, China. His research interests include 3-D shape analysis, machine learning, and human–computer interaction. Wu received the PhD degree in computer science from Zhejiang University. Contact him at wuzizhao@hdu.edu.cn.

Ming Zeng is currently an associate professor with the Software School of Xiamen University, China. His research interests mainly lie in computer graphics and computer vision, especially analysis and reconstruction of 3-D objects. He received the PhD degree in computer science from Zhejiang University in 2013. Contact him at zengming@xmu.edu.cn.

Feiwei Qin is an associate professor with the College of Computer Science and Technology, Hangzhou Dianzi University. His research interests include artificial intelligence, image processing, and CAD. He received the PhD degree in computer science from Zhejiang University. Contact him at qinfeiwei@hdu.edu.cn.

Yigang Wang was born in Songxian, Henan Province. He is a professor with the School of Media and Arts, Hangzhou Dianzi University. His research interests include image processing, computer graphics, and virtual reality. He received the PhD degree in applied mathematics from Zhejiang University, China. Contact him at yigang.wang@hdu.edu.cn.

Jiří Kosinka is an assistant professor with the Scientific Visualization and Computer Graphics Research Group, Bernoulli Institute, University of Groningen, The Netherlands. His research interests include computer graphics, geometric modeling and processing, and computer aided design. He received the PhD degree from Charles University, Prague, in 2006. Contact him at j.kosinka@rug.nl.