Eurographics Symposium on Geometry Processing 2014
Thomas Funkhouser and Shi-Min Hu
(Guest Editors)

Volume 33 (2014), Number 5

3D Shape Segmentation and Labeling via Extreme Learning Machine

Zhige Xie [1] Kai Xu [†1] Ligang Liu[2] Yueshan Xiong [1]

[1]HPCL, School of Computer Science, National University of Defense Technology, PR China
[2]School of Mathematical Sciences, University of Science and Technology of China, PR China

Abstract

We propose a fast method for 3D shape segmentation and labeling via Extreme Learning Machine (ELM). Given a set of example shapes with labeled segmentation, we train an ELM classifier and use it to produce initial segmentation for test shapes. Based on the initial segmentation, we compute the final smooth segmentation through a graph-cut optimization constrained by the super-face boundaries obtained by over-segmentation and the active contours computed from ELM segmentation. Experimental results show that our method achieves comparable results against the state-of-the-arts, but reduces the training time by approximately two orders of magnitude, both for face-level and super-face-level, making it scale well for large datasets. Based on such notable improvement, we demonstrate the application of our method for fast online sequential learning for 3D shape segmentation at face level, as well as realtime sequential learning at super-face level.

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Computer Graphics]: Segmentation—

1. Introduction

Labeled segmentation is one of the essential issues towards the high-level understanding of 3D shapes. Extensive research has been devoted to this important problem, with the help of recent advances on machine learning techniques. Existing approaches to this problem can be broadly categorized as supervised and unsupervised methods. In supervised methods, some learning techniques, such as JointBoost [KHS10] or AdaBoost [BLVD11], have been utilized to train a complex classifier. Based on the initial labeling, the actual segmentation is formulated as, e.g., Conditional Random Field (CRF) [KHS10], graph cuts [vKTS*11] or Snake [BLVD11], to obtain smooth segmentation results. For these methods, the state-of-the-art works can obtain segmentation accuracy of about 94% [KHS10]. However, the training process is extremely time-consuming, which can easily take several hours. On the unsupervised side, clustering methods has been utilized for segmentation and labeling [SvKK*11, HFL12]. To accelerate the computation, these methods often cluster super-faces [HKG11,WAvK*12] obtained via over-segmentation. Super-face-level clustering can greatly improve the running time, but the segmentation results are generally inferior to those by the supervised methods working at face level.

The goal of this work is to improve the training performance of supervised approach, making the highly accurate segmentation and labeling scale well for large datasets and furthermore, available for online learning scenarios. To this end, we employ the recently developed Extreme Learning Machine (ELM) to train a neural networks classifier. ELM works on generalized single-hidden layer feedforward networks (SLFNs) and provides good generalization at a much cheeper learning cost due to the avoidance of hidden layer tuning [HWL11]. Moreover, ELM can automatically set weights for each dimension of the feature vectors, acting similar to the boosting methods. We use the trained ELM classifier to predict labels for all mesh faces. Based on that, we perform graph-cut based segmentation optimization, constrained by the refined boundaries from both over-segmentation and ELM-based segmentation, leading to a smooth segmentation.

Our work makes two contributions. Firstly, we introduce ELM into the labeled segmentation of 3D shapes to reduce

[†] Corresponding author: kaixu@nudt.edu.cn

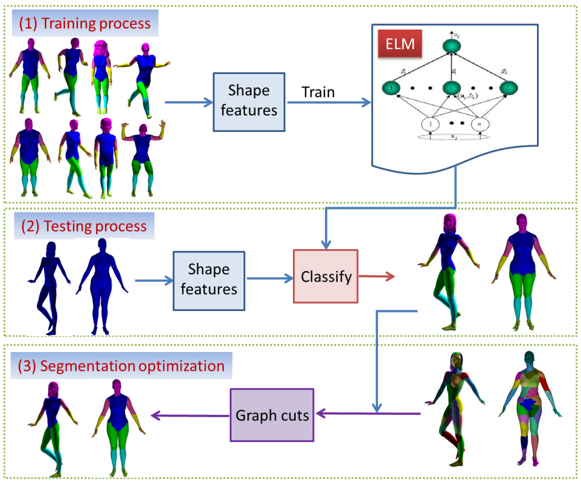

Figure 1: *The pipeline of our ELM-based segmentation algorithm. Stage 1: An ELM classifier is trained with an exemplar dataset. Stage 2: The classifier is used to predict the face labels on test models. Stage 3: The final smooth segmentation is obtained by graph-cut optimization constrained by the super-face boundaries obtained by over-segmentation and the active contours computed from ELM segmentation.*

the training cost of supervised method significantly. Secondly, we demonstrate how this efficient ELM-based approach can be extended for the task of online sequential learning for 3D shape segmentation. To the best of our knowledge, we are the first to provide a *realtime* online sequential learning application on 3D shape segmentation.

2. Related work

Supervised methods for segmentation and labeling. Supervised methods train a mesh face classifier from a collection of labeled 3D meshes and then use it to predict the face labels on test 3D meshes. Advanced machine learning techniques have been successfully utilized in training the classifier. The actual segmentation can be formulated as Conditional Random Field (CRF) or graph cuts to generate the final smooth segmentation.

Kalogerakis et al. [KHS10] proposed the first supervised labeled segmentation for 3D shapes, demonstrating significant improvement over traditional methods based on local analysis of shape concavity/convexity. van Kaick et al. [vKTS*11] incorporated prior knowledge imparted with a set of pre-segmented and labeled models with content-driven analysis to perform part correspondence. Benhabiles et al. [BLVD11] presented a fully automatic 3D mesh segmentation algorithm based on boundary edge learning, which could obtain better segmentation boundaries.

To date, the state-of-the-art supervised methods have obtained segmentation accuracy of 94%. However, the costly training process has been a major obstacle hindering its scalability for large datasets and availability for online scenarios.

Unsupervised segmentation and labeling. Much research has been dedicated to unsupervised approach for segmenting a set of 3D meshes simultaneously and consistently, which is also referred to as co-segmentation. Based on diffusion map in descriptor space, Sidi et al. [SvKK*11] proposed a method to co-segment a set of shapes with large variability. However, their method relies on initial segmentations for each input shape and may fail if the per-shape segmentation is poor. Huang et al. [HKG11] presented a novel linear programming approach to joint segmentation of a heterogeneous shape collection, producing comparable results to the supervised approaches on the Princeton Segmentation Benchmark (PSB) [CGF09]. Xu et al. [XLZ*10] clustered man-made objects based on their anisotropic part proportion styles in a correspondence-free manner, obtaining co-segmentation used for generating shape variations. Hu et al. [HFL12] employed subspace clustering to perform consistent shape segmentation. Meng et al. [MXLH12] built a statistical model to describe each cluster of parts obtained by an initial segmentation, and employed the multi-label optimization to iteratively improve the co-segmentation.

To accelerate unsupervised methods, researchers opt to compute features at patches, or super-faces [SvKK*11, HKG11]. Clustering at patch level is computationally much cheaper, but still takes several minutes. In general, unsupervised methods produces inferior segmentation and labeling results than supervised ones.

Extreme learning machines. Neural networks and support vector machines (SVMs) are two very important techniques for training classifiers in a supervised fashion. However, it has been shown that both the two techniques face some challenges, such as slow learning speed, intensive human intervention, etc. Extreme learning machine (ELM) [HZS06], as an emerging learning technique, aims to overcome these challenges and has received much attention lately. Among all the advantageous properties of ELM, the reduced training time is perhaps the most attractive to its various applications. ELM works with generalized single-hidden layer feedforward networks (SLFNs). The key feature of ELM is that the hidden layer of SLFNs need not be tuned [HZDZ12], making it achieve better scalability with the much faster learning speed.

3. Learning theory of extreme learning machines

In this section, we give a brief introduction to the fundamentals of ELM [HWL11]. For N arbitrary samples $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbf{R}^d \times \mathbf{R}^m$, a single-hidden layer feedforward network (SLFN) with L hidden nodes can be mathematically modeled as:

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i g_i(x_j) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}_j) = \boldsymbol{o}_j. \quad j = 1, \ldots, N \qquad (1)$$

The SLFN can approximate these N samples with zero error means, i.e., $\sum_{i=1}^{L} \|\mathbf{o}_j - \mathbf{t}_j\| = 0$, meaning that there exist $(\boldsymbol{a}_i, b_i)$ and $\boldsymbol{\beta}_i$ such that

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}_j) = \boldsymbol{t}_j. \quad j = 1, \ldots, N \tag{2}$$

The above N equations can be written in the matrix form:

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}, \tag{3}$$

where

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}(x)_1 \\ \vdots \\ \boldsymbol{h}(x)_N \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_1) \\ \vdots & \cdots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_N) \end{bmatrix}_{N \times L},$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\beta}_L^{\mathrm{T}} \end{bmatrix}_{L \times m} \quad \text{and} \quad \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{t}_N^{\mathrm{T}} \end{bmatrix}_{N \times m}.$$

$\boldsymbol{H}$ is called the hidden layer output matrix of the SLFN; $G(\boldsymbol{a}_i, b_i, \boldsymbol{x}_i)$ $(i = 1, \ldots, n)$ are the hidden layer feature mapping. After randomly generating the hidden node parameters $(\boldsymbol{a}_i, b_i)$, training an SLFN is equivalent to finding a least-squares solution $\hat{\beta}$ of Equation (3) [HZS06]:

$$\hat{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{T} \tag{4}$$

where $\boldsymbol{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\boldsymbol{H}$.

For the case where the number of training examples is huge, Huang et al. [HZDZ12] proposed a faster implementation of ELM. In their training process, every input feature has a weight term. In this paper, we discard this term in order to fully exploit the feature weight choosing mechanism of ELM. For online sequential training, we modify the online sequential ELM (OS-ELM) [LHSS06] by putting an ELM testing step in every training update step, to be able to terminate and output testing results at any time.

4. 3D shape segmentation and labeling via extreme learning machine

In this section, we provide the details of our ELM-based mesh segmentation method, including the shape descriptors involved and the three-stage training-segmentation process.

4.1. Algorithm overview

Our ELM-based segmentation algorithm proceeds in three stages (Figure 1). In the first stage, we calculate feature descriptors for all faces of the input meshes and feed these feature vectors to ELM for classifier training. In the second stage, the trained ELM classifier is used to segment and label the test meshes. Finally, we over-segment the test meshes into super-faces on which we perform graph-cut optimization to obtain smooth segmentation results.

4.2. Shape descriptors

We compute a set of shape descriptors for each face of an input mesh. These descriptors capture different aspects of the face properties, such as local shape geometry around the face as well as contextual information. The main idea of using a collection of descriptors is that their union is expected to be informative enough for face discrimination [vKTS*11]. Specifically, we extract the unary features used in the supervised segmentation of Kalogerakis et al. [KHS10], including those based on principal component and curvature analysis over the neighborhood of a face, shape diameter, average of geodesic distances, and geodesic shape contexts.

Besides unary ones, Kalogerakis et al. also use pairwise features including five different categories of descriptors for each pair of adjacent faces. The extraction of these pairwise features is much more expensive than that of unary ones. In their method, the JointBoost needs to be trained for both unary and pairwise features. The two trained JointBoost classifiers are then integrated into a CRF model. On the contrary, our ELM classifier is trained using only unary features, making the training process extremely fast. In our method, the pairwise segmentation optimization is incorporated into graph cuts (Section 4.4.2).

4.3. ELM classifier training and testing

In the training stage, we first normalize the features computed for all input meshes. Then the parameters, such as the number of hidden neurons, the number of training meshes and the activation function, are chosen. The details of parameter settings are described in Section 5.1.

The key change to the original ELM made by our method is that we discard the feature weight term in the training process, which means that we weight all features uniformly. In fact, ELM can automatically choose weights for every dimension of the features, similar to the feature selection of boosting methods. The output of the ELM training is a classifier and the associated accurate classification rate for the training data. In the testing stage, every face of the test meshes is classified into a specific class by the ELM classifier. The output will be used as the initial labeling for the graph-cut based segmentation optimization, which is described in the following.

4.4. Segmentation optimization

The labeling results obtained by ELM are usually noisy and the segmentation boundaries are jaggy. Therefore, we perform graph cuts to optimize the initial labeling into a smooth segmentation. To achieve smooth boundaries, the graph-cut

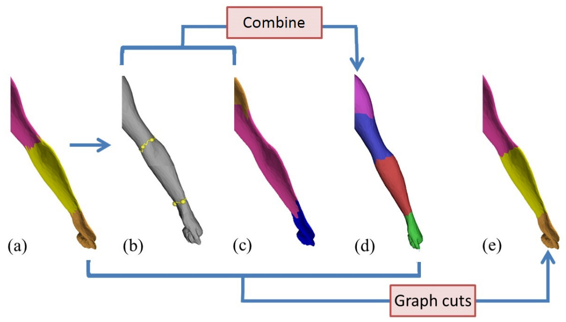

Figure 2: *An overview of segmentation optimization. (a): Initial segmentation result by ELM. (b): Active contours of the ELM segmentation boundaries. (c): Over-segmentation computed by normalized cuts. (d): Segmentation result from combining the boundaries from both (b) and (c). (e): Final segmentation obtained by graph-cut optimization.*

optimization is constrained by the combination of the superface boundaries obtained by over-segmentation, and the active contours computed based on ELM segmentation.

4.4.1. Boundary extraction

Boundaries of over-segmentation. We first over-segment each mesh into primitive patches [HKG11]. Given a mesh, we first employ the normalized cuts [GF08] to decompose it into primitive patches, and then optimize the patch boundaries to align them against shape features using fuzzy cuts [KT03]. Let us denote the boundaries of over-segmentation by B_{overseg} which is a set of boundary edges.

Active contours of ELM segmentation. Another type of boundaries we consider is the segmentation boundaries of ELM. However, ELM segmentation often produces jaggy and even incomplete boundaries. To extract complete and smooth boundaries from the ELM segmentation results, we perform contour completion and active contour movement, similar to the work of Benhabiles et al. [BLVD11]. Figure 2(b) shows the extracted contours from ELM segmentation of the right arm of a human model. Let us denote the refined boundaries of ELM segmentation by B_{ELM}.

Boundary combination. We combine all the boundaries obtained above by taking the union of the two sets of edges, denoted by $B_{\text{comb}} = B_{\text{overseg}} \cup B_{\text{ELM}}$, leading to a new, finer segmentation. In the new segmentation, we remove very small segments whose area is smaller than 1% of the total area of the mesh and merge it into its neighbouring segment with the maximum area. The process of boundary combination is demonstrated in Figure 2(b-d). B_{comb} is then used to guide a smooth segmentation in graph cuts.

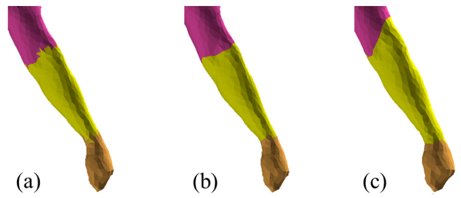

Figure 3: *The effect of the weighting factor ω_{uv} over segmentation boundary. (a): Initial labeling result by ELM. (b): Segmentation optimized by graph cuts with the weighting factor. (c): Segmentation by graph cuts without the weighting factor. Clearly, (b) preserves the boundaries in (a) better.*

4.4.2. Graph-cut optimization

With the labeling predicted by the ELM classifier, as well as the boundary constraints obtained above, we perform graph cuts to obtain the final smooth segmentation. Given a mesh, we represent it with a graph $G = \{V, E\}$, where the nodes V correspond to triangles of the mesh and an arc $(u, v) \in E$ exists if the faces u and v are adjacent. Let n be the number of all possible labels for the current shape category. For each face u, ELM produces n neuron outputs with corresponding probabilities. The ELM label of u, l_u, takes the one with the highest probability. The segmentation optimization is then reduced to finding the labeling l that minimizes the energy:

$$E(l) = \sum_{u \in V} E_D(u, l_u) + \lambda \sum_{(u,v) \in E} E_S(u, v, l_u, l_v), \qquad (5)$$

where l_u and l_v are the labels assigned to u and v, respectively. Moreover, E_D and E_S are the data and smoothness energy term, respectively. λ is used to tune the importance of the smoothness term; we use $\lambda = 50$ in all our experiments.

The data term $E_D(u, l_u)$ is used to ensure that a triangle $u \in V$ can be assigned with the label corresponding to the maximum probability, and is given by:

$$E_D(u, l_u) = -log(p(l_u|u)) \qquad (6)$$

where $p(l_u|u)$ is the probability that node u is assigned to label l_u by ELM classifier.

Similarly to [SSS*10], the smoothness term is defined as:

$$E_S(u, v, l_u, l_v) = \begin{cases} 0, & \text{if } l_u = l_v \\ \omega_{uv} \ell_{uv} (1 - log(\theta_{uv}/\pi)), & \text{otherwise} \end{cases} \qquad (7)$$

where ℓ_{uv} is the length of the incident edge of face u and v, and θ_{uv} is the dihedral angle between the two faces. ω_{uv} is a weighting factor incorporating the boundary constraints based on the boundaries of both over-segmentation and ELM segmentation.

We impose the boundary constraints through favoring the

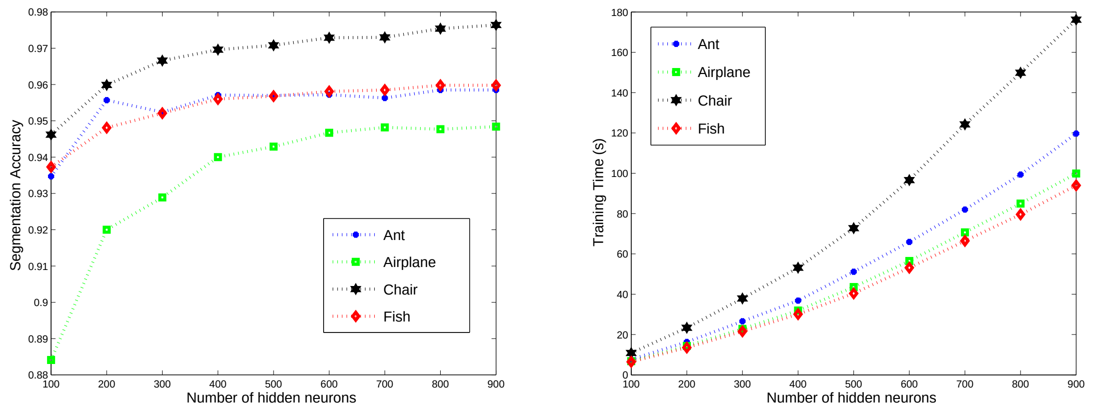

Figure 4: *The plots show the influence of different choices of $N_{neurons}$ over segmentation accuracy (left) and training time (right) for four datasets from PSB [CGF09]. Both segmentation accuracy and training time take the average for each dataset. It can be seen that when $N_{neurons}$ is set to* 500*, we obtain a good trade-off between training time and segmentation accuracy.*

cuts across the boundaries in B_{comb}:

$$\omega_{uv} = \begin{cases} 10, & \text{if } e_{uv} \in B_{\text{comb}} \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

where e_{uv} is the incident edge of face u and v. The effect of imposing boundary constraints with ω_{uv} is shown in Figure 3. We use the multi-label α-expansion algorithm [BVZ01] to minimize the energy $E(l)$.

5. Implementation and results

To evaluate the segmentation and labeling performance of our method, we test it over datasets across twenty-three shape categories, among which nineteen are from the Princeton Segmentation Benchmark (PSB) [CGF09] and the rest four from COSEG [WAvK*12]. For COSEG datasets, we evaluate our method against the ground-truth segmentations provided therein. For PSB datasets, we choose the ground-truth as in [KHS10].

5.1. Segmentation results at face level

There are two important parameters in our method, i.e. the number of hidden neurons used in ELM, $N_{neurons}$, and the segment count of over-segmentation, $N_{overseg}$. The choice of $N_{neurons}$ has significant effect over the speed of ELM training and the accuracy of ELM testing. In Figure 4, we show how different choices of $N_{neurons}$ would affect the training time and the final segmentation results. All the face-level results presented in this paper were produced with the fixed parameter setting: $N_{neurons} = 500$ and $N_{overseg} = 50$. In ELM classifier training, we choose the mostly used sigmoid function as the activation function for each input feature. Other choices, such as Radial Basis Function (RBF), are also possible. Figure 5 shows a gallery of segmentation results for twenty categories of datasets from both PSB and COSEG.

Table 1 (left) reports detailed results for the corresponding datasets in Figure 5, using four different configurations of training data size: 1) All shapes from the dataset except one (ELM-LeaveOne), 2) 75% shapes out of the dataset (ELM-75%), 3) Half of the dataset (ELM-50%) and, 4) 25% of the dataset (ELM-25%). The results shown in the corresponding columns are those by ELM labeling. We also demonstrate the results of segmentation optimization based on the labeling results by ELM-LeaveOne, denoted by ELM-Opt. The accuracy of labeled segmentation is measured by the area-weighted recognition rate proposed in [KHS10]. The Rand Index is computed using the method in [CGF09]. Both of them are measured against the ground-truth labeling.

5.2. Segmentation results at super-face level

Based on ELM labeling, our method can also perform mesh segmentation and labeling at super-face level. To generate training data with super-face-level labels, we first over-segment each input mesh into 100 super-faces [HKG11]. Then the ground-truth labels for the super-faces are generated based on the ground-truth labeling at face level. Specifically, given a super-face, we take the label which covers the maximal area within the super-face as its ground-truth label.

To train classifier at super-face level, we need to define feature for super-faces. Similar to Hu et al. [HFL12], we define the feature vector of a super-face as the histogram of the feature measure over all faces on that super-face. The number of bins for each histograms is set to 30 in all our experiments. The other parameters used by ELM are the same as those in Section 5.1.

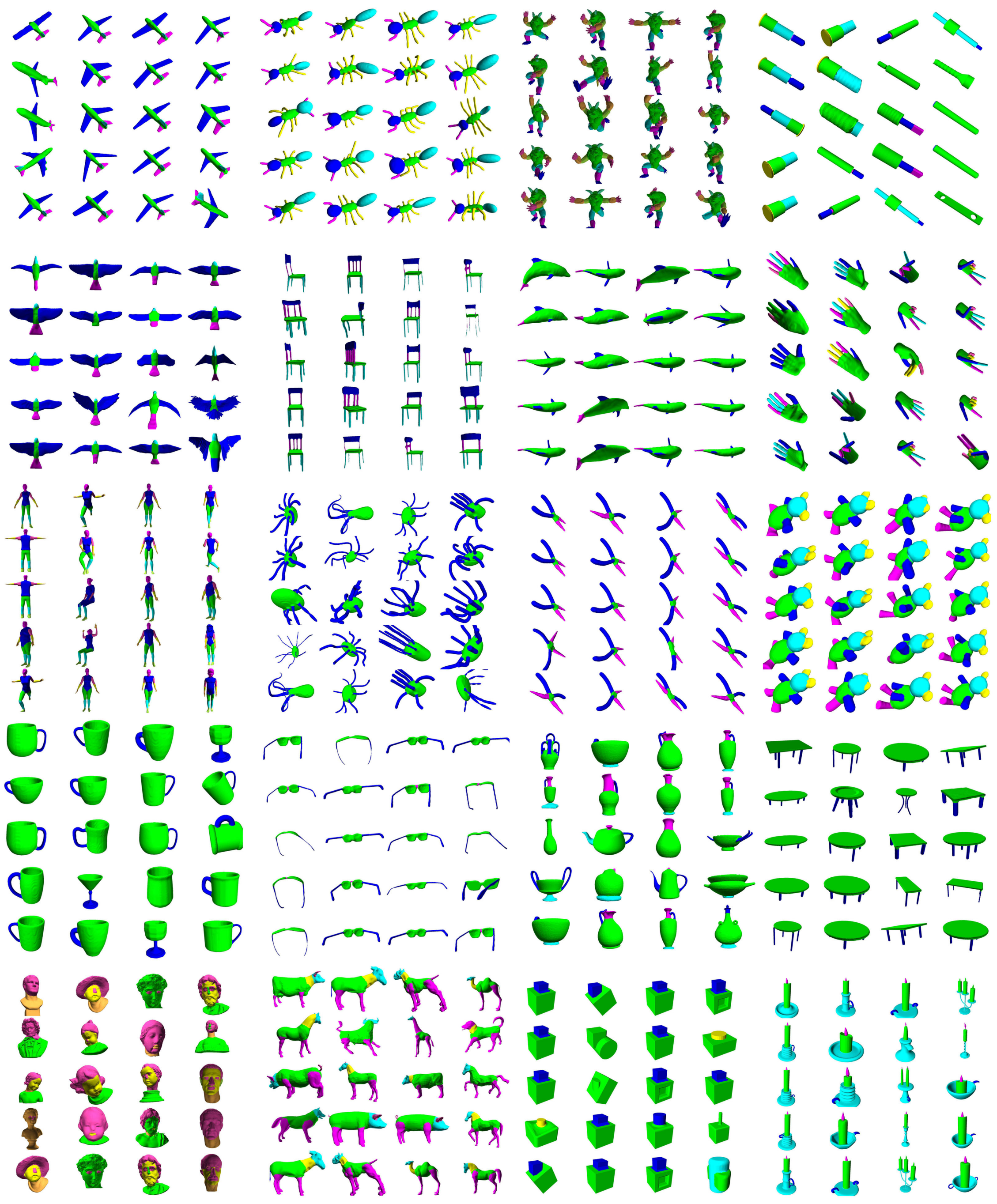

Figure 5: *A gallery of segmentation results by our ELM-Opt algorithm for twenty datasets: Airplane, Ant, Armadillo, Bearing Bird, Chair, Fish, Hand, Human, Octopus, Plier, Teddy, Cup, Glasses, Vase, Table, Bust, FourLeg, Mech, and Candelabra. The first nineteen are from PSB [CGF09] and the last from COSEG [WAvK* 12].*

	ELM Segmentation: Recognition Rate					Segmentation: RandIndex		
	25%	50%	75%	LeaveOne	Opt	Training	SB19	ELM-Opt
Ant	95.09	95.25	95.72	95.79	95.98	1.7	1.9	4.2
Airplane	90.51	92.38	93.20	94.29	94.49	7.4	7.9	8.9
Armadillo	83.57	84.02	84.79	84.21	87.86	6.3	8.0	9.4
Bird	93.56	95.53	95.57	95.58	95.68	4.4	10.4	16.6
Bearing	48.17	67.39	79.63	83.12	90.84	6.8	9.7	15.4
Chair	95.52	96.92	97.08	97.28	97.41	5.2	5.4	7.1
Fish	81.52	85.81	87.68	87.72	95.68	11.8	12.9	13.2
Human	80.87	84.43	86.27	87.47	88.61	11.2	11.9	9.8
Hand	69.50	73.10	75.33	82.60	83.47	9.1	10.4	11.4
Octopus	97.46	97.60	97.70	98.37	98.73	1.8	1.8	1.8
Plier	95.23	95.78	95.80	95.84	95.96	5.1	5.4	5.4
Table	96.22	97.33	98.31	98.37	99.31	5.9	6.2	5.9
Teddy	97.60	97.72	97.73	97.74	97.74	3.1	3.1	3.6
Glasses	95.95	96.68	96.70	96.72	96.79	8.4	13.6	8.8
Cup	95.87	96.43	97.27	98.23	98.32	9.8	9.9	10.3
Vase	71.19	79.00	80.63	88.81	89.52	10.5	16.0	15.6
Bust	48.41	52.23	58.62	60.12	61.36	18.8	21.4	22.1
FourLeg	77.40	80.78	80.89	84.23	91.44	15.9	13.2	9.1
Mech	80.81	81.89	82.50	84.95	87.49	8.5	10.0	15.9
Vase (COSEG)	92.12	93.34	93.66	93.67	93.87	0	-	6.8
Candelabra (COSEG)	83.39	89.78	96.49	96.60	96.79	0	-	3.5
Chair (COSEG)	96.66	96.95	97.35	97.40	97.70	0	-	4.2
Lamp (COSEG)	95.72	96.48	98.00	98.09	98.22	0	-	2.1

Table 1: *The left part of the table reports the segmentation accuracy (recognition rate) of our ELM method on various datasets, using four configurations of training data size: ELM-25%, ELM50%, ELM-75%, ELM-LeaveOne. The results by ELM-Opt (optimized from ELM-LeaveOne) is also given. The right part shows the Rand Index scores [CGF09] of the training segmentations, and the resulting segmentations obtained by SB19 [KHS10] and ELM-Opt. The Rand Index in the first nineteen datasets are measured against all human segmentations in the Princeton benchmark [CGF09], while those in the last four datasets against the ground-truth provided in COSEG. Therefore, the Rand Index of training segmentations in the last four sets are zero. The Rand Index scores of SB19 for the last four sets are missing since the method was not tested over those datasets.*

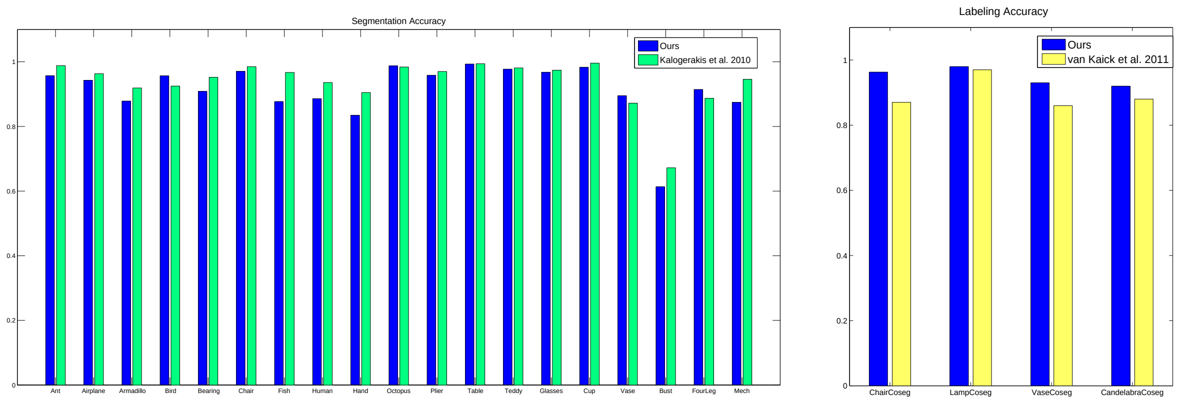

Figure 6: *Comparisons of segmentation accuracy between our ELM segmentation and two start-of-the-art methods. Left: comparison between ELM-Opt and SB19 [KHS10], where both methods use "leave-one-out" training data configuration for each dataset. Right: comparison between ELM-75% (boundary optimized by graph cuts) and the method by van Kaick et al. [vKTS*11], both of which use 75% of the dataset as training data.*

	ELM-LeaveOne		ELM-75%	
	Train (sec)	Acc. (%)	Train (sec)	Acc. (%)
Fish	1.89	93.05	1.71	92.92
Airplane	1.62	90.35	1.52	88.30
Teddy	1.91	96.80	1.73	96.64
Ant	1.84	96.10	1.64	95.34
Octopus	1.71	97.53	1.56	96.98

Table 2: *Training time (in second) and segmentation accuracy (recognition rate) of our super-face-level segmentation algorithm over two configurations of training data size. The five datasets are from PSB.*

Table 2 reports the training time and segmentation accuracy of super-face-level segmentation by our method, for two configurations of training sets, respectively. As expected, when working with super-faces, the total size of features input to ELM training is greatly reduced (less than 1% of that at face level). Consequently, the training time is further improved, while the segmentation accuracy is not sacrificed much. Note that the accuracy of labeled segmentation is measured against the original, face-level ground-truth.

5.3. Comparison with state-of-the-arts

In this section, we first compare our method with two supervised approaches. For PSB datasets, Kalogerakis et al. [KHS10] demonstrated the segmentation results of their method, with the training size configuration of "leave-one-out", similar to the ELM-LeaveOne for our method. Such configuration yields the most accurate results for both methods. In the comparison, we show results by ELM-Opt, which are optimized from those by ELM-LeaveOne. van Kaick et al. [vKTS*11] used the COSEG datasets and their method was evaluated with the training data size of 75% of the entire dataset. Hence, when comparing with [vKTS*11], we show the graph-cut optimization results based on ELM-75%.

Table 1 (right) compares the Rand Index scores obtained by ELM-Opt and the method of Kalogerakis et al. (SB19) [KHS10], over various datasets. While our method obtains comparable segmentation accuracy with theirs, our training time is typically less than one minute (Table 3), in contrast to several hours in their method. For some input shapes, our method produces visually more meaningful boundaries as compared to other methods; see Figure 7. Figure 6 compares the segmentation accuracy between ELM-Opt and [KHS10] (left side), as well as between ELM-75% (boundary optimized) and [vKTS*11] (right side). From the figure, our method is comparable to [KHS10] while consistently outperforming [vKTS*11].

For face-level segmentation and labeling, our method improves the training time over the above two works significantly. Since we could not obtain the source code of the two works, we provide an approximate timing comparison, based

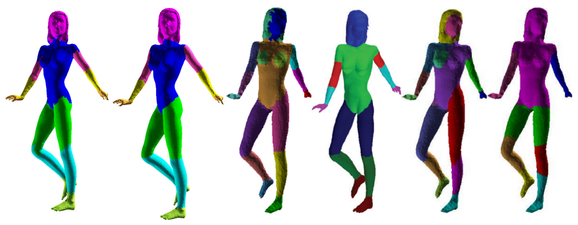
Figure 7: *From left to right, we show segmentation results on a human model, obtained by ELM-LeaveOne , ELM-Opt, as well as the methods from [BLVD11], [KHS10], [GF09] and [SSCO08]. Our method obtains visually more meaningful segmentation at the boundary between head and neck.*

on the timing report in the original papers, only to show a general idea of the extend of the improvement. For example, in [KHS10], the training process took about 8 hours for a dataset with 6 meshes with about 25~30K faces. Our method takes only about 15 seconds on the same dataset. Taking the difference of CPU speed into accout, our method improves the training time by about two orders of magnitude. Note that for almost all methods, testing time is usually negligible as compared to training time. In our case, the ELM labeling and the graph-cut optimization together take less than 3 seconds on average; see Table 3 for full timing report.

5.4. Results on larger datasets

Nowadays, the scale of shape collections keeps growing, leading to more and more large datasets on the Internet. So far, the existing methods for mesh segmentation and labeling can not scale well for the large datasets, due to the high training cost. Wang et al. [WAvK*12] introduced active learning for interactive labeling large model sets. Due to the fast training of ELM-based segmentation method, it is natural to use it to segment and label large datasets. We test our method on the three large scale datasets provided in COSEG [WAvK*12]. The three datasets, Tele-aliens, Vases and Chairs, contain 200, 400 and 300 models, respectively. All the three sets come with ground-truth labeling which can be used for training and evaluation. We apply our method (ELM-75%) on these datasets, with the default parameter setting. The segmentation accuracy, measured by recognition rate, for the three datasets are 83.23%, 87.09% and 85.86%, respectively. The training time for the three sets are 260.6s, 241.7s and 105.9s, respectively.

5.5. Performance

All the experiments were performed on a machine with Intel(R) Quad-Core 3.2 GHz CPU and 16GB RAM. The ELM training and testing processes, as well as graph-cut optimization, are implemented in Matlab. The total running time of

	LeaveOne	75%	Testing	Graph-cut
Airplane	43.56	34.62	1.41	0.42
Bird	36.57	29.55	1.15	0.43
Fish	40.40	31.99	1.28	0.52
Human	63.25	50.61	2.04	1.11
Octopus	59.20	45.98	1.75	1.02
Glasses	30.34	23.59	0.92	0.31

Table 3: *Time consumption (in second) of the various stages, i.e., training, testing and graph-cut optimization, on six datasets from PSB. The training time is given for two configurations of training data size.*

our face-level algorithm on a set of 20 shapes is typically less than one minute. The proportion of time consumption is around 96% for ELM training, 3% for ELM testing, and 1% for graph-cut optimization (Table 3). The average running time of our super-face-level algorithm is around 2 seconds; see Table 2.

6. Application

In many applications, training data often come one by one or chunk by chunk. In these cases, on-line learning is more preferable than batch learning since the former does not need retraining when a new datum is received [HWL11]. Online Sequential Extreme Learning Machine (OS-ELM) [LHSS06] is a simple and efficient online learning algorithm developed based on ELM. OS-ELM can absorb training data not only in a one-by-one manner, but also in a chunk-by-chunk fashion, with fixed or varying length, and update the classifier with only the newly incoming training data.

Online sequential learning at face level. The process of OS-ELM is described as follows. First, an initial ELM classifier is trained using an initial set of training data. When new training data come in, the classifier is updated using the new input data. The process is repeated until all input data have been received. The key feature of OS-ELM is that the classifier updating can be done incrementally, based only on the new training data.

Based on OS-ELM, we present an online sequential learning framework for 3D shape segmentation. The main change we make to the original OS-ELM is that we put an ELM testing procedure after every training update step, so that the entire algorithm can terminate and output segmentation results at any time. This modification is especially useful for our application scenario, where the user may provide the system with a few new labeled shapes each time and would like to see the newly updated labeling results immediately.

The system of online sequential learning for 3D shape segmentation can be realized with a client-server model (Figure 8). In the server machine, a database of 3D shapes are stored, with which an initial ELM classifier is trained. At

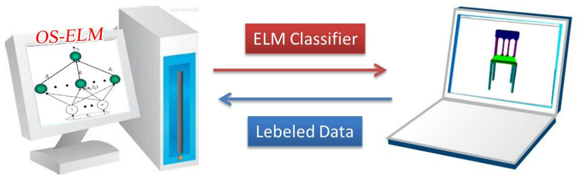

Figure 8: *A client-server model implementation of online sequential learning for 3D mesh segmentation.*

the client end, the user may introduce one or a bunch of new shapes, providing rough labels for them and uploading the labeled shapes to the server. When receiving the new data, the server will wake up and update the ELM classifier with the new data using OS-ELM algorithm. The updated ELM classifier is then sent to clients. With this updated ELM classifier, the user can obtain a more accurate labeling for the new shapes she has just introduced, since the classifier training benefits from both the labeled database on the server and the user's labeling.

In our experiments, we use the same datasets and parameter setting as in Section 5.1. Since the OS-ELM update process is fast, taking usually less than 20 seconds for the datasets we have tested, the response time is acceptable. Figure 9 plots the change of segmentation accuracy over sequentially provided training data. The ChairLargeScale dataset is one of the large sets in COSEG containing 100 chairs. For all the datasets, the accuracy becomes stable after 25% of the training data are provided.

Realtime online learning at super-face level. When working with super-faces, the OS-ELM can achieve *real-*

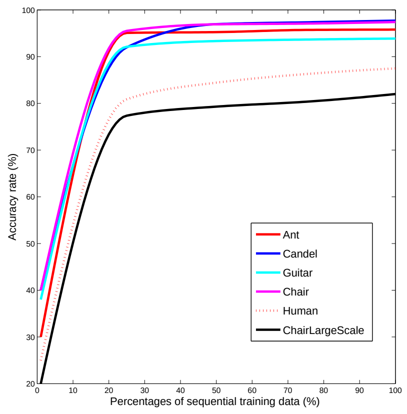

Figure 9: *The change of segmentation accuracy over sequentially provided training data with online sequential learning at face level. The curves are smoothed with cubic interpolation for readability purpose.*

	Initial accuracy (%)	Update accuracy (%)	Final accuracy (%)	Update time (sec)
Fish	61.50	1.66	93.05	0.14
Airplane	48.33	2.21	90.35	0.13
Teddy	54.17	2.24	96.80	0.11
Ant	67.75	1.49	96.10	0.15
Octopus	49.75	2.51	97.53	0.17

Table 4: *Training time and segmentation accuracy of online sequential learning at super-face level on five datasets from PSB. Initial accuracy is the testing accuracy of the initial OS-ELM classifier trained with two shapes for each dataset. Final accuracy is the testing accuracy of the final OS-ELM classifier trained with all shapes for each dataset. Update accuracy is the incremental improvement of accuracy in each update. Update time is the averaged time consumption per update.*

time online learning. The method of feature extraction for super-faces is the same as in Section 5.2. To achieve an even faster classifier updating, we implement the Least-Square Incremental Extreme Learning Machine (LS-IELM) algorithm [GHL14]. All other experimental configuration and parameter setting are the same as face-level. The running time and segmentation accuracy is reported in Table 4. With such a fast updating (about 0.14s per update on average), our method achieves realtime online sequential learning for 3D shape segmentation.

7. Conclusion, limitations and future works

We have described a new method of labeled segmentation of 3D shapes based on Extreme Learning Machine. As previous works, we employ a set of feature descriptors to train a classifier, with which we predict the face labels used for mesh segmentation. However, instead of relying on the time-consuming boost methods, we train a neuron network classifier using ELM, achieve highly efficient training process. To obtain smooth segmentation boundaries, we employ graph cuts to optimize the segmentation boundaries under the constraints of the super-face boundaries of over-segmentation and the active contours of ELM segmentation. Experimental results show that our method achieves comparable results against the state-of-the-art methods while improving the training time significantly, making our method scale well for large datasets, and moreover, available for online sequential learning for mesh segmentation.

Our current method has two main limitations. Firstly, the quality of our final segmentation results highly depends on that of over-segmentation. Secondly, our ELM training takes only the labeling information in the training data into account, leading to a face classifier, but does not exploit the boundary information imparted in the data, aiming at a boundary edge classifier as in [BLVD11].

In the future, we plan to fully utilize the boundary information from the training data for boundary classifier training, to further improve the segmentation boundaries. We would also like to extend our ELM training and testing procedures to other 3D shape analysis applications. Our work could represent a first step towards the realtime 3D shape segmentation. Moreover, we believe that the general idea of realtime online sequential learning enabled by the ELM learning framework would find more applications towards high-level understanding of 3D shapes.

8. Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments. We would like to thank Kuan Li for fruitful discussion. We are also grateful to Oliver van Kaick for providing his code of feature extraction and Guang-Bin Huang for the code of ELM. This work was partially supported by the Research Fund for the Doctoral Program of Higher Education of China (No. 20104307110003), the National Natural Science Foundation of China (No. 61379103, 61202333, 61222206, and 61303189), China Postdoctoral Science Foundation (No. 2012M520392) and One Hundred Talent Project of the CAS, China.

References

[BLVD11] Benhabiles H., Lavoué G., Vandeborre J.-P., Daoudi M.: Learning boundary edges for 3d-mesh segmentation. *Computer Graphics Forum 30*, 8 (2011), 2170–2182. 1, 2, 4, 8, 10

[BVZ01] Boykov Y., Veksler O., Zabih R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pat. Ana. & Mach. Int. 23*, 11 (2001), 1222–1239. 5

[CGF09] Chen X., Golovinskiy A., Funkhouser T.: A benchmark for 3d mesh segmentation. *ACM Trans. on Graph (SIGGRAPH) 28*, 3 (2009), 73:1âĂŞ–73:12. 2, 5, 6, 7

[GF08] Golovinskiy A., Funkhouser T.: Randomized cuts for 3d mesh analysis. *ACM Trans. on Graph (SIGGRAPH Asia) 27*, 5 (2008), 145:1–145:12. 4

[GF09] Golovinskiy A., Funkhouser T.: Consistent segmentation of 3d models. *Computers & Graphics 33*, 3 (2009), 262–269. 8

[GHL14] Guo L., Hao J.-h., Liu M.: An incremental extreme learning machine for online sequential learning problems. *Neurocomputing 128* (2014), 50–58. 10

[HFL12] Hu R., Fan L., Liu L.: Co-segmentation of 3d shapes via subspace clustering. *Computer Graphics Forum 31*, 5 (2012), 1703–1713. 1, 2, 5

[HKG11] Huang Q., Koltun V., Guibas L.: Joint shape segmentation with linear programming. 125:1–125:11. 1, 2, 4, 5

[HWL11] Huang G.-B., Wang D. H., Lan Y.: Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics 2*, 2 (2011), 107–122. 1, 2, 9

[HZDZ12] Huang G.-B., Zhou H., Ding X., Zhang R.: Extreme learning machine for regression and multiclass classification. *IEEE Trans. on Syst., Man, and Cybern. 42*, 2 (2012), 513–529. 2, 3

[HZS06] Huang G.-B., Zhu Q.-Y., Siew C.-K.: Extreme learning machine: theory and applications. *Neurocomputing 70*, 1 (2006), 489–501. 2, 3

[KHS10] Kalogerakis E., Hertzmann A., Singh K.: Learning 3d mesh segmentation and labeling. *ACM Trans. on Graph (SIGGRAPH) 29*, 4 (2010), 102:1–102:12. 1, 2, 3, 5, 7, 8

[KT03] Katz S., Tal A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. on Graph (SIGGRAPH) 22*, 3 (2003), 954–961. 4

[LHSS06] Liang N.-Y., Huang G.-B., Saratchandran P., Sundararajan N.: A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Trans. on Neural Networks 17*, 6 (2006), 1411–1423. 3, 9

[MXLH12] Meng M., Xia J., Luo J., He Y.: Unsupervised co-segmentation for 3d shapes using iterative multi-label optimization. *Computer-Aided Design 45*, 2 (2012), 312–320. 2

[SSCO08] Shapira L., Shamir A., Cohen-Or D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer 24*, 4 (2008), 249–259. 8

[SSS*10] Shapira L., Shalom S., Shamir A., Cohen-Or D., Zhang H.: Contextual part analogies in 3d objects. *Int. J. Comp. Vis. 89*, 2-3 (2010), 309–326. 4

[SvKK*11] Sidi O., van Kaick O., Kleiman Y., Zhang H., Cohen-Or D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. on Graph (SIGGRAPH Asia) 30*, 6 (2011), 126:1–126:10. 1, 2

[vKTS*11] van Kaick O., Tagliasacchi A., Sidi O., Zhang H., Cohen-Or D., Wolf L., Hamarneh G.: Prior knowledge for part correspondence. *Computer Graphics Forum 30*, 2 (2011), 553–562. 1, 2, 3, 7, 8

[WAvK*12] Wang Y., Asafi S., van Kaick O., Zhang H., Cohen-Or D., Chen B.: Active co-analysis of a set of shapes. *ACM Trans. on Graph (SIGGRAPH Asia) 31*, 6 (2012), 165:1–165:10. 1, 5, 6, 8

[XLZ*10] Xu K., Li H., Zhang H., Cohen-Or D., Xiong Y., Cheng Z.-Q.: Style-content separation by anisotropic part scales. *ACM Trans. on Graph (SIGGRAPH Asia) 29*, 6 (2010), 184:1–184:11. 2