Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Co-skeletons: Consistent Curve Skeletons for Shape Families

Zizhao Wu[a,b,*], Xingyu Chen[b], Lingyun Yu[c], Alexandru Telea[d], Jiří Kosinka[b]

[a]School of Media and Design, Hangzhou Dianzi University, China
[b]Bernoulli Institute, University of Groningen, the Netherlands
[c]Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China
[d]Department of Information and Computing Sciences, Utrecht University, the Netherlands

ARTICLE INFO

Article history:
Received May 1, 2020

Co-skeleton, curve skeleton, mesh processing, shape segmentation

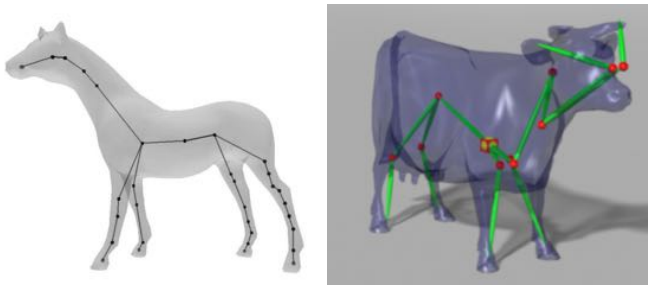
ABSTRACT

We present co-skeletons, a new method that computes consistent curve skeletons for 3D shapes from a given family. We compute co-skeletons in terms of sampling density and semantic relevance, while preserving the desired characteristics of traditional, per-shape curve skeletonization approaches. We take the curve skeletons extracted by traditional approaches for all shapes from a family as input, and compute semantic correlation information of individual skeleton branches to guide an edge-pruning process via skeleton-based descriptors, clustering, and a voting algorithm. Our approach achieves more concise and family-consistent skeletons when compared to traditional per-shape methods. We show the utility of our method by using co-skeletons for shape segmentation and shape blending on real-world data.

1. Introduction

Skeletons are thin and locally centered structures which describe the geometry, topology, and symmetry properties of shapes compactly and intuitively. This makes skeletons powerful tools for applications such as shape segmentation [1], manipulation [2, 3], and blending [4]. Existing skeletonization methods can be classified in methods that compute *surface* skeletons [5, 6] and methods that compute *curve* skeletons [7, 8]. Surface skeletons capture shape geometry better, but curve skeletons are much simpler (and faster) to compute, represent, and analyze, and are the dominant skeleton type currently used in applications [9].

Among the many existing curve skeletonization, important differences exist regarding the *quality* of the produced skeletons, which is measured by criteria including thinness, centeredness, compactness, robustness to noise, homotopy equivalence to the input shape, and computational complexity [12]. Quality issues create problems when using skeletons in certain applications such as mesh rigging. Figure 1 shows the curve skeletons (CSs) extracted by two such methods. The extracted skeletons exhibit problems such as shrinkage of end branches with respect to the corresponding shape parts or even exiting the shape at places. Other skeletonization methods exhibit different problems with respect to the mentioned quality criteria. Given such problems, most of the skeleton-based mesh rigging approaches require that



(a) Le *et al. 2003* [10] (b) Liu *et al. 2014* [2]

Fig. 1. Curve skeletons extracted by Le *et al.* [10] and Liu *et al.* [2] for mesh rigging. As visible, the produced skeletons are not always locally centered and, at places, even exit the shape.

*Corresponding author: wuzizhao@hdu.edu.cn

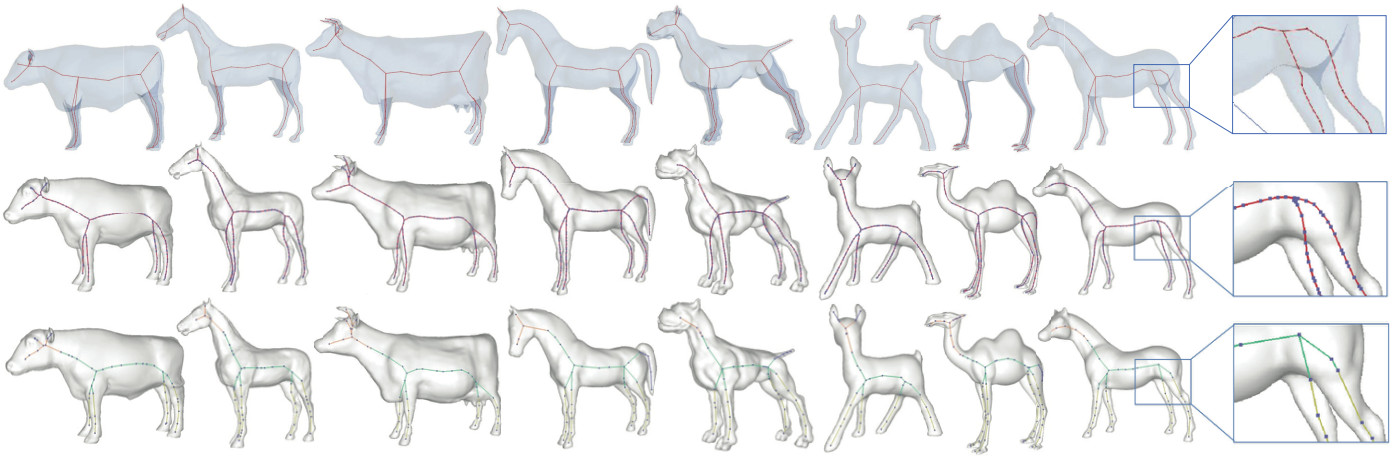


Fig. 2. Top and mid rows: Skeletons extracted from the *fourleg* subset of the Princeton Shape Benchmark (PSB) [11] using the Mean Curvature Flow [8] and Mesh Contraction [7] methods, respectively. These skeletons have a variable sampling density and preserve (or not) similar details across different shapes. Bottom row: Co-skeletons extracted by our approach are more concise and consistent in preserving similar details across different shapes.

skeletons are manually specified for the input shape, a process which is time-consuming and error-prone.

Obtaining high quality curve skeletons — important for applications like rigging [13] and shape segmentation [1] — has been done so far by proposing increasingly improved skeletonization methods. Yet, new methods may introduce new problems, more user parameters, or have a more complex implementation [9]; and must be thoroughly tested on large shape collections [14, 12]. Another problem of all skeletonization methods is that they cannot guarantee that they preserve the *same* level-of-detail on similar parts of *any* input shape. Given an actual shape and parameter settings, details may be kept in the skeleton or simplified away. This creates *inconsistent* skeletons from the viewpoint of applications that use them further to manipulate shapes.

We propose a different approach: Inspired by recent approaches on co-analysis of 3D shape collections, rather than aiming to compute a high-quality skeleton from a single *shape*, we use a *collection* of shapes (of the same kind) to compute their skeletons. The intuition behind this is that a given skeletonization method will be able to extract good-quality skeletons from *most* parts of *most* shapes, and will not fail consistently on the same parts of all shapes. By combining information from all extracted skeletons, we obtain *co-skeletons* which represent well all shapes in a given family with controlled and consistent sampling density and presence of significant details in all skeletons, even when large variations exist between individual shapes.

Our method works as follows: Given a 3D shape collection, we extract the curve skeletons from all shapes, using any existing good-quality CS method chosen by the user. Next, we use several descriptors to characterize these skeletons, and use them to cluster similar branches from different skeletons. Finally, we infer the semantic correlation among corresponding edges and use this information to *jointly* prune all skeletons to achieve *conciseness* (representing CSs with few sampling points and edges) and *consistency* (the same type of shape detail creates the same type of skeleton branch) over an entire shape family. We show our automatic co-skeleton extraction framework by applications of shape co-segmentation and shape blending.

Our main contributions are as follows:

1. We present an approach to induce semantic correlations for curve skeletons of 3D shapes;
2. We propose a semantic-based skeleton pruning approach;
3. We show the usability of our pruned skeletons by applications of shape co-segmentation and shape blending.

We organize this paper as follows. Section 2 reviews related work in curve skeletonization. Section 3 outlines our framework. Section 4 details our pruning algorithm for skeleton consistency. Section 5 presents results and applications. Section 6 discusses our proposal and outlines future work directions.

2. Related Work

2.1. Skeleton Extraction

Skeletons, or medial axes, were introduced by Blum [15] based on the medial axis transform (MAT) which computes the centers and radii of maximal balls lying within a shape. Skeletons have been used in many applications, including shape segmentation [1, 7], manipulation [2, 3], matching [16], and modeling [4]. For additional details, we refer to recent surveys of 3D skeletonization methods and their applications [9, 17].

3D shapes admit two main skeleton types: *Surface* skeletons, defined following Blum's MAT; and *curve* skeletons, defined more loosely as curvilinear (1D) structures locally centered in the shape [18, 12]. Curve skeletons are much more popular, since they are considerably simpler and faster to compute, represent, and manipulate than surface skeletons. Yet, they lack a universally accepted definition [9]. This has led to many curve skeletonization methods, each emphasizing different desirable properties up to different extents, *e.g.*, homotopy equivalence to the input shape, robustness to noise, smoothness, centeredness, and invariance to isometric shape transformations [14, 12]. For example, Biasotti *et al.* [19] proposed Reeb graphs that formalize a concise topological encoding of shapes and can be embedded

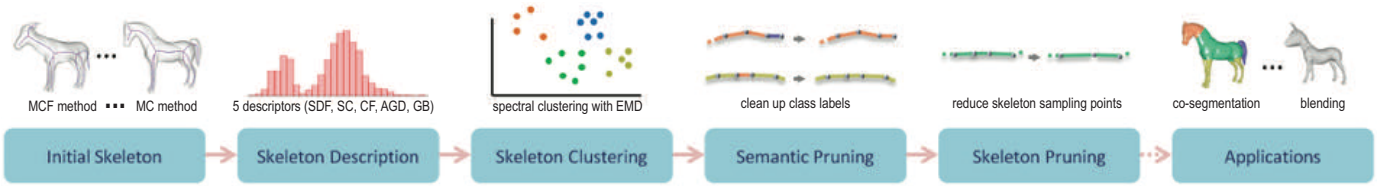


Fig. 3. Our method has six steps. Curve skeletons are extracted from 3D input shapes using existing skeletonization methods. These skeletons are next reduced to five per-face descriptors. The descriptors of all skeletons from a shape family are next clustered. Finally, we prune (simplify) the clustered data to remove semantic noise (semantic pruning) and obtain skeletons with a small sample count (skeleton pruning). We use co-skeletons for shape co-segmentation and blending applications. Section 3 details all the pipeline steps.

geometrically to define compact curve skeletons. Hassouna *et al.* [20] extract well-centered curve skeletons by tracing curves seeded at high-divergence points in a gradient vector field. Li *et al.* [21] pioneered mesh decimation methods to compute curve skeletons. Marini *et al.* [22] suggest shape-prototypes which summarize the most relevant features of a shape class to help with 3D shape retrieval.

Contraction methods are arguably the most widely used techniques for curve skeletonization [7, 23, 24, 25] as they are relatively simple to implement, fast, and can handle large and densely-sampled meshes [12]. Within this class, Au *et al.* [7] contract a mesh into a zero-volume skeletal shape by implicit Laplacian smoothing with global positional constraints. This mesh is converted into a 1D curve-skeleton through a connectivity-surgery process to remove collapsed faces while preserving its shape and the original topology. Tagliasacchi *et al.* [8] compute the Voronoi diagram of a 3D shape's vertices and its medial poles and next iteratively use an implicit constrained Laplacian solver to optimize triangulation by local remeshing until the shape's volume vanishes. Upon convergence, the method produces a medially-centered curve skeleton. In our work, we use both Au *et al.* [7] and Tagliasacchi *et al.* [8] to extract initial curve skeletons that we further refine into our co-skeletons. Figure 2 (top and mid rows) shows skeletons computed by both these methods for various shapes. As visible there, these two methods compute globally similar, but locally different, curve skeletons. However, we will show that this does not influence the results obtained by our co-skeletons.

Sensitivity of the computed skeletons to shape changes, due to sampling density or small-scale noise, is arguably the key challenge of all existing skeletonization methods [9]. Sensitivity, also called instability, is handled in two main ways: First, one can smooth the input shape prior to skeletonization to remove noise [26, 27]. Secondly, one can prune the extracted skeletons to remove so-called spurious branches, based on various importance metrics [28, 29, 30]. Rather than relying upon automatic importance metrics, Giachetti *et al.* [31] guide skeleton pruning manually. Jiang *et al.* [25] propose a curve skeleton extraction approach by coupled graph contraction and surface clustering. Baran and Popović [13] present a skeleton extraction method to automatically rig an unfamiliar character for skeletal animation. Skeletal importance is typically defined based on the area of the input shape that contracts to a given skeleton point [32, 30, 33, 34]. However, no such approach can *ensure* that the same pruning *level* occurs for the same details present in *different* input shapes having possibly different sampling densities. In contrast, we perform skeleton pruning based on high-level semantic information extracted from an entire shape collection. This ensures skeletons that have the same level of detail for all shapes in such a collection. Figure 2 (bottom row) is an example that illustrates this property of our results.

Collections of shapes have been studied in the context of skeletonization: Schaefer and Yuksel [35] introduced an example-based skeleton-extraction approach for mesh deformation. Zheng *et al.* [36] proposed a consensus skeleton pruning approach. Yet, such approaches only deal with mesh *sequences* (dynamic meshes with fixed connectivity), used *e.g.* to capture different poses of the *same* shape. In contrast, our method handles shape collections showing large variations, including topological changes and different shapes — see *e.g.* the different animals in the *fourleg* collection in Fig. 2. Also, note that, technically, the method in [36] is driven by *pairwise* skeleton correspondences. In our case, we do not use pairwise correspondences, but treat an entire *family* at a time.

Skeleton extraction from natural images is another related challenging topic [17], with no single method able to *consistently* create good skeletons in all cases. To address this, Jerripothula *et al.* [37] suggested a joint co-skeletonization and co-segmentation framework, exploiting inherent interdependencies of skeletons and segments to assist each other. We are inspired by this work, but our proposed solution uses a different technique.

2.2. Shape Co-analysis

There has been recent increasing interest in the co-analysis of shape collections. The premise is that more information can be extracted by analyzing a collection than when analyzing individual shapes. An example hereof is *co-segmentation* — the simultaneous segmentation of all shapes in a set in a consistent manner. This has been shown to be of great utility for modeling and texturing [38, 39, 40]. Golovinskiy and Funkhouser [41] pioneered consistent co-segmentation by aligning all shapes and then clustering their primitives. Following this work, many co-segmentation approaches have been proposed, using unsupervised learning [39, 42, 43] or semi-supervised learning [44, 45]. Deep learning has shown excellent performance in this direction, with several methods proposed for shape segmentation [46, 47]. Yet, such methods heavily rely on large training datasets.

Besides co-segmentation, other approaches exist for the co-analysis of a shape-set. Laga *et al.* [48] presented an effective algorithm to obtain semantic correspondences between 3D shapes

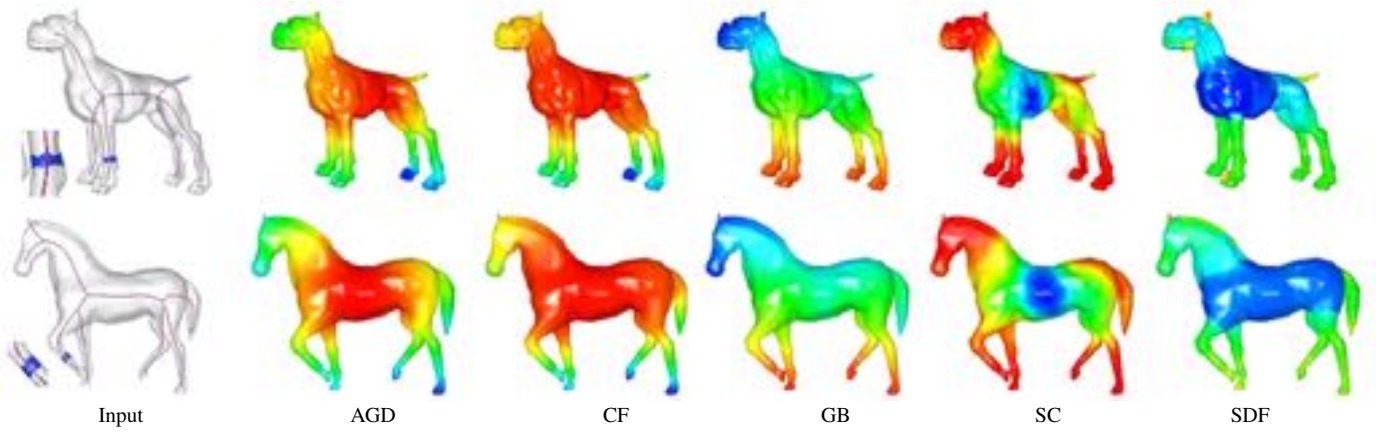


Fig. 4. Feature extraction in our pipeline. Left column: Two examples of input shapes with their initial curve skeletons (red) and the shape faces (blue) associated with a selected skeleton edge. The other columns show our five feature descriptors computed on the two models (feature names are detailed in Section 3). These descriptors are aggregated, via their histogram distributions, to form the skeleton-edge descriptors.

that finds part-wise correspondences. Kaick *et al.* [49] constructed a unified shape co-hierarchy from a shape set, providing a richer characterization of the shape-set beyond coarse template-based or part-level correspondence. Yumer and Kara [50] propose a co-abstraction method where shapes in a set are abstracted as much as possible while still preserving the unique geometric characteristics distinguishing them from each other. Xu *et al.* [51] synthesize new shapes by analyzing a given shape-set using genetic algorithms. Kim *et al.* [52] construct cuboid model templates of large shape sets. Fish *et al.* [53] learn the configurations of a shape-set as geometric distributions. Yumer and Kara [54] use co-constrained handles to deform shape-sets to find and respect the geometric and spatial constraints among different shape parts. Our work is inspired by these techniques: We compute family-consistent skeletons in terms of sampling density and semantic relevance. Hence, even if the underlying curve-skeletonization method that we use is imperfect, the problems that it creates on *individual* shapes are alleviated or removed by considering *all* shapes in the family.

3. Proposed Method

Figure 3 shows the pipeline of our method. Our input is a collection of N shapes $\mathcal{I} = \{I_1, \ldots, I_N\}$, $I_i \subset \mathbb{R}^3$, of one shape family. By a family, we mean a set of shapes that belong to the same semantic class, *e.g.*, four-legged animals or chairs. Shapes are represented as boundary meshes [9]. The goal of our method is to obtain family-consistent curve skeletons $\mathcal{S} = \{S_1, \ldots, S_N\}$, one for each shape in the input collection. For each shape I_i, we consider its curve skeleton S_i, modeled as a set of 3D sample points P_i connected by edges E_i, *i.e.*, $S_i = (P_i, E_i)$, $P_i \subset \mathbb{R}^3$, $E_i = \{e_{i,1}, e_{i,2}, \ldots, e_{i,n_i}\} \subset P_i \times P_i$. For a collection $\mathcal{I}$, we first extract the initial skeletons S_i for each shape I_i individually, using existing state-of-the-art methods [7, 8]. We then extract features for all skeleton edges $e_{i,j}$ and cluster edges in a joint descriptor space to infer their semantic correlation. To obtain concise and family-consistent skeletons, we propose and apply two pruning algorithms, which lead to co-skeletons S_i suitable for shape co-segmentation and shape blending. We next describe each step of our pipeline in turn.

Initial Skeleton Extraction: We extract shape skeletons S_i using the mean curvature flow (MCF) method [8] or, alternatively, the mesh contraction (MC) method [7]. Any other curve skeletonization method can be used directly, if desired. For selecting alternatives to MCF and MC, one can study the survey of Sobiecki *et al.* [14] to pick the method of choice based on various desirable properties, such as speed, ease of use, type of input (mesh or voxel volume), or robustness. For brevity, we next show results based on the MCF method [8], which we found slightly easier to use than MC and producing smoother curve skeletons (see also Fig. 2). However, using MC yields very similar co-skeletons to MCF, so the choice between the two is largely left to the user's preference. Regardless of the skeletonization method choice, we compute skeletons for each shape with approximately $|E_i| = 100$ skeleton edges each.

Skeleton Description: No matter which skeletonization technique one uses, every skeleton edge can be mapped to a set of shape faces [9]. This mapping [55], known as the feature transform $FT : S \rightarrow \mathcal{P}(I)$, with $\mathcal{P}$ denoting the power set, maps $FT(e \in S)$ to the set of faces in I that correspond to a skeleton edge e. The FT complements topological shape information, captured by the curve-skeleton's branching structure, with geometric information that encodes which skeleton fragments capture which shape-surface details. Both information types are essential for semantic or functional prediction.

We use five shape descriptors to characterize surface details, similar to previous co-analysis approaches [39, 42, 56]: Shape Diameter Function (SDF) [57], Conformal Factor (CF) [58], Shape Contexts (SC) [59], Average Geodesic Distance (AGD) [60], and Geodesic distance to the Base of the shape (GB) [39]. These descriptors are defined on mesh faces. Taking the computation of SC as an example, given a mesh face, we compute the distribution of all other faces (weighted by their area) in logarithmic geodesic-distance bins and uniform-angle

bins, where angles are measured relative to the normal of each face. Hence, each shape face is described by five scalar values that correspond to the five above mentioned descriptors.

Given a skeleton edge $e_{i,j} \in S_i$ with its associated faces $FT(e_{i,j}) \in I_i$ and their face descriptors, we use normalized histograms with a specified bin value to measure the feature distribution of $e_{i,j}$. Using histograms ensures that the number of faces $|FT(e_{i,j})|$ belonging to a skeleton edge $e_{i,j}$ is normalized over all skeleton edges. We next compute a so-called *descriptor space* over all shapes I_i in a family. Figure 4 shows two examples. The leftmost column shows the faces (blue) associated to a skeleton edge (red). The other columns show the five feature descriptors we compute, color-coded on a rainbow colormap. As explained, these descriptors are ultimately recorded on the skeleton edges via their feature histograms.

Skeleton Clustering: As already explained, there is no unanimously accepted formal definition of 3D curve skeletons, let alone of co-skeletons for a shape *family*. This implies that it is difficult to define consistency. Hence, we proceed by processing all skeleton edges in a shape family in a unified and global manner. As we do not have explicit semantic information, we resort to clustering, which is the approach of choice in many related co-analysis techniques. We therefore simplify (cluster) *all* edges of *all* skeletons of a shape family, together with their computed descriptor values, in the per-family descriptor space. To simplify notation, let e_a and e_b be two skeleton edges in the whole family. For each descriptor, let $p_a = FT(e_a)$ be the set of faces corresponding to an edge e_a. Let $h_{a,k}$ be the histogram over p_a of the k-th descriptor ($1 \le k \le 5$). We define the *dissimilarity* between two edges e_a and e_b with respect to the k-th descriptor as

$$d_k(e_a, e_b) = \mathrm{EMD}(h_{a,k}, h_{b,k}),$$

where $\mathrm{EMD}(h, \bar{h})$ is the Earth Mover's Distance [61] between histograms h and $\bar{h}$. EMD is a typical method for evaluating dissimilarity between two multidimensional distributions in a feature space. We next apply a Gaussian kernel to the distances d_k to build an affinity matrix $W_k = (w_{a,b,k})$ for each descriptor k, with entries

$$w_{a,b,k} = \exp\left(-\frac{d_k(e_a, e_b)}{2\sigma^2}\right), \tag{1}$$

where $w_{a,b,k}$ is the dissimilarity between e_a and e_b for the k-th descriptor. We set the number of bins to 50 for each histogram, and σ to the mean of all dissimilarities, respectively.

We now seek a way to combine the five affinity matrices W_k into a single matrix, to be next used to perform the co-skeleton computation. We note that our five descriptors characterize partially-related shape aspects. For instance, the AGD and CF descriptors take typically large values on a shape's center and low values on its extremities; see Fig. 4. Hence, simply merging the five affinity matrices W_k into a single matrix would result in redundant information. To avoid this, we use affinity-aggregation spectral clustering [42] to jointly perform feature selection and clustering — that is, reduce the amount of redundant information and also decompose the resulting information into self-similar subsets. For this, we proceed as follows: Let $\alpha = [\alpha_1, \ldots, \alpha_5]$ be weights associated to the affinity matrices $W_1, \ldots, W_5$ that indicate how much each matrix contributes to describing similarity over the shape family. We formulate the affinity-aggregation spectral clustering as

$$\min_{\alpha, F} \sum_{k=1}^{5} \sum_{a,b} \alpha_k w_{a,b,k} \|f_a - f_b\|^2, \tag{2}$$

where $F = [f_1, \cdots, f_m]$ is the indicator vector in joint feature space having a total of m samples.

The minimization in Eqn. 2 involves two unknown vectors, α and F. To solve for them, we use a two-step minimization approach that alternatively fixes one unknown vector and varies the other. During optimization, two additional constraints must be satisfied: The first one comes from normalized spectral clustering, *i.e.,* the final diagonal matrix D must satisfy

$$1 = F'DF = F'(\alpha_1 D_1 + \cdots + \alpha_5 D_5)F = \sum_{k=1}^{5} \alpha_k s_k, \tag{3}$$

where

$$s_k = F' D_k F$$

and D_k is a diagonal matrix whose i-th diagonal element is the sum of the elements in the i-th row of W_k. Using this constraint, spectral clustering typically converges to a result [62]. The second constraint comes from the Cauchy-Schwartz inequality, which leads to constraining the sum of the weighted matrices in a normalized condition, *i.e.*,

$$\sum_{k=1}^{5} \sqrt{\alpha_k} = 1. \tag{4}$$

By applying the Lagrange multiplier method to the constraints in Eqns. 3 and 4, the problem of finding α can be reduced to a one-dimensional line-search problem, which is easy to solve. For more details, we refer the interested reader to [63].

After obtaining F, we run k-means in feature space to cluster the data into C classes, where C is assigned according to the number of parts of each shape in the input family $\mathcal{I}$, under

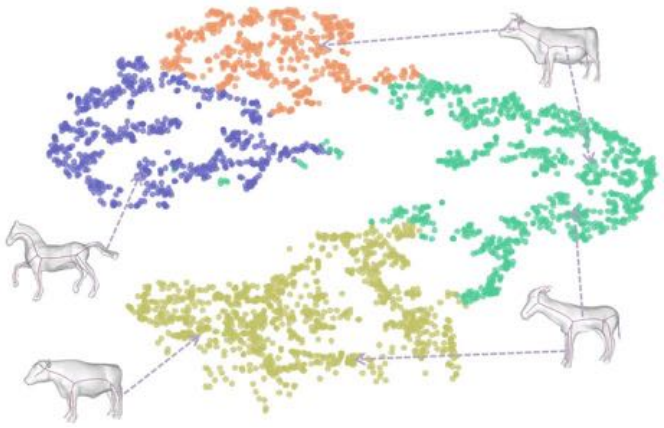


Fig. 5. Skeleton edge clustering in joint feature space. Family-consistent semantic correlation can be deduced from the clusters.

human supervision. That is, for a given shape family, the user has to decide what is a suitable number of parts that typical shapes in that family have — or, putting it differently, by how many parts the user wants to model shapes in that family. Figure 5 shows our clustering result on the *fourleg* dataset, visualized using t-SNE [64] with different clusters colored differently. Points that are close in the embedding (2D) space have thus similar feature vectors. Skeleton edges that belong to the same cluster are assumed to be semantically similar. Note that this assumption is reasonable as many co-analysis algorithms operate under it (see Sec. 2). As the figure shows, four large clusters appear, which correspond to four parts of shapes in the *fourleg* dataset.

Semantic Pruning: We next use semantic pruning to remove so-called *semantic noise*, which occurs in our clustering due to three reasons: (1) Shape meshes may contain noise, which propagates into the five descriptors. (2) The descriptors themselves contain a certain amount of fuzziness, as they only measure geometric properties rather than the 'true' semantic ones. (3) Our feature selection and clustering steps may introduce artifacts. To remedy these issues, we introduce a semantic pruning step. This step uses the connectivity of skeleton edges, based on the idea that two connected skeleton edges have a high probability of carrying the same semantic information. Importantly, semantic pruning only 'merges' the semantic information extracted from different skeletons in the same family; it does *not* actually remove skeleton nodes, a task which is done next by the skeleton pruning. We describe semantic pruning in detail in Sec. 4.1.

Skeleton Pruning: A final concern is to compute *compact* co-skeletons, *i.e.*, having a small number of points. This assists, speed-wise, all operations that next use co-skeletons. To obtain compact co-skeletons, we carry out skeleton pruning to reduce potential skeleton *over-sampling*. Since existing skeleton-pruning algorithms do not take into account semantic part information over a family of shapes [9], we propose a new pruning procedure that considers and respects such semantic properties. We describe skeleton pruning in detail in Sec. 4.2.

4. Skeleton Pruning Details

We next describe the semantic and skeleton pruning algorithms which aim to reduce semantic noise, respectively over-sampling, in our produced co-skeletons.

4.1. Semantic Pruning

As stated in Sec. 3, our semantic pruning exploits the observation that two connected skeleton edges usually hold the same semantic information. Hence, this connectivity information can add extra information to the initial clustering results.

Our semantic pruning algorithm exploits four properties of a skeleton edge to measure the confidence that two skeleton edges fall within the same semantic part. These are the length of the edge, the angles between the edge and the two edges connected to it, and the semantic information of the connected edges themselves.

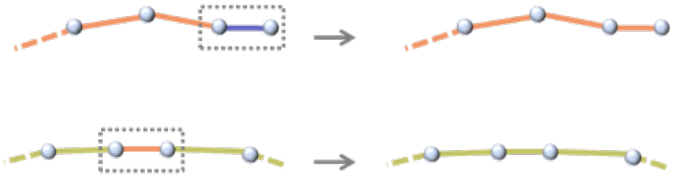


Fig. 6. Semantic pruning for leaf skeleton edge (top) and a joint skeleton edge (bottom). Colors show class. The role of semantic pruning is to clean up class labels, not to remove edges; skeleton pruning does the latter.

Edge length: Given a skeleton edge e_i, we compute its normalized length l_i (with respect to the longest edge in I, so $l_i \in [0, 1]$) and its angles $\beta_{i,j}$ to edges $N_i = \{e_j\}$ to which e_i is connected in the skeleton graph E. Let the cluster index of e_i, as computed by k-means (Sec. 3), be denoted by $c_i \in [1, \ldots, C]$. Given our clustering, c_i thus encodes semantic similarity of the edges. With the above, we define the *confidence score* K_i of e_i as

$$K_i = \lambda l_i + \sum_{e_j \in N_i} b_{i,j} l_j \gamma(i, j), \tag{5}$$

where $\gamma(i, j) = 1$ when $c_i = c_j$ and $\gamma(i, j) = -1$ otherwise. The hyperparameter λ (default: 1.5) defines the relative weight given to edge lengths *vs* edge angles in the confidence score.

Edge angles: The angles $\beta_{i,j}$ are first normalized into $[0, 1]$ by computing

$$\hat{b}_{i,j} = (1 + \cos\beta_{i,j})/2 \tag{6}$$

and then mapped by a Gaussian kernel to yield the weights

$$b_{i,j} = \exp\left(-\frac{\hat{b}_{i,j}^2}{2}\right).$$

This way, the smaller the angle $\beta_{i,j}$, the smaller the final weight $b_{i,j}$. When $\beta_{i,j} = \pi$, we obtain the highest value of $b_{i,j} = 1$.

Equation 5 assigns a low confidence to an edge e_i when its connected edges e_j have different semantic definition, *i.e.*, come from different clusters than e_i. Conversely, a high confidence score tells that e_j has the same semantic information as e_i.

We next sort all skeleton edges e_i of a shape collection ascendingly on their scores K_i and process them in this order as follows. For each e_i, we refine its confidence score K_i to equal the one of its highest-confidence edge-neighbor, *i.e.*, $K_i = \max_{e_j \in N_i} K_j$. After processing an edge, we refresh its confidence score and that of its neighbours. We repeat the process until we have processed approximately 10% of all edges (see Fig. 6). This value has been set empirically based on tests comprising many shape categories.

4.2. Skeleton Pruning

From the initial skeletons with their pruned semantic attributes, we now perform skeleton pruning to reduce over-sampling and to remove spurious branches. This way, we achieve *compact* co-skeletons that describe the respective shapes with a small number of sampling points and edges.

Fig. 7. Illustration of our skeleton pruning process. A skeleton node is removed depending on its associated angle and the lengths of its incident edges.

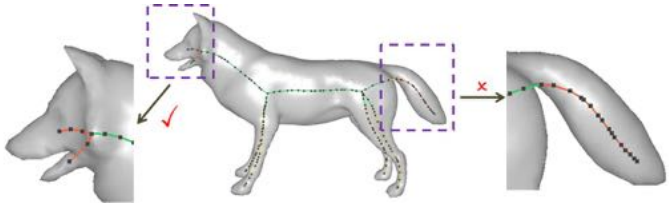


Fig. 8. A correct case (left) and a wrong case (right) of semantic pruning in one example. Our approach may fail to deal with the case of successive semantic noise in skeleton edges.

In contrast to semantic pruning, we now focus on the nodes (vertices) $\mathbf{x}_i \in P$ of the curve skeletons. We categorize all nodes into three groups: leaf, joint, and branch. A *leaf* node is incident with only one skeleton edge; a *joint* node with two edges; and a *branch* node with at least three edges. As most nodes are joint nodes in curve skeletons, we focus on pruning those only. Also, not pruning leaf or branch nodes ensures that the topology of the pruned skeletons stays identical to the initial ones. Node positions are not altered by pruning, so the pruned skeletons maintain their centeredness with respect to the original shapes.

We exclude from pruning joint nodes whose incident edges carry different semantic information (cluster labels c_i). Pruning such nodes would be difficult, as we would need to somehow merge different cluster labels into newly created edges. Let e_i and e_j be the two incident edges for a node candidate for pruning, and let $\beta_{i,j}$ be the angle spanned by these edges. We use $\beta_{i,j}$ and the normalized edge lengths l_i and l_j, defined as in Sec. 4, to compute a node confidence score as

$$V_{i,j} = (l_i + l_j)\hat{b}_{i,j} \tag{7}$$

with $\hat{b}_{i,j}$ defined by Eqn. 6. Following Eqn. 7, nodes with large angles and with short incident edges have low confidence values. Conversely, nodes with small angles and long incident edges get high confidence values. This models the fact that we want to prune densely-sampled and relatively-straight skeleton branches. We sort all skeleton nodes ascendingly on their confidence values $V_{i,j}$ and prune (remove) nodes in this order, one at a time. After each node removal, we recompute the confidence scores $V_{i,j}$ of the node's two neighbors in the skeleton graph. We prune until approximately 75% of the joint nodes are pruned. Different thresholds yield a more, respectively less, aggressive skeleton simplification, as desired by the application at hand (see Fig. 7).

5. Results and Applications

We tested our method on a PC with an Intel 4GHz i7 processor and 8GB RAM. Our time complexity mainly depends on the descriptor computation. Clustering takes under 1 minute with 20 iterations for 2K skeleton edges. Semantic pruning and skeleton pruning computations are linear in skeleton size, taking one second for 2K skeleton edges. Overall, our end-to-end pipeline takes about 8 minutes for small datasets (20 shapes with around 2K skeleton edges/shape), and scales linearly for larger data.

We demonstrate the utility of our co-skeletons by comparing our results with the raw curve skeletons (extracted as explained in Sec. 5.1). We also show co-skeletons in action in two applications: shape segmentation and shape blending (Sec. 5.2).

5.1. Co-skeleton results

We compare our co-skeletons with the initial MCF and MC curve skeletons [7, 8]. As test data, we used the Princeton Shape Benchmark (PSB) dataset [11], which contains sets of shapes of multiple types, *e.g.*, animals, chairs, human models, furniture, and vehicles. Figure 2 shows that our co-skeletons are significantly more concise (have fewer nodes) than the original curve skeletons, while preserving overall desirable characteristics such as centeredness and topology. Moreover, each edge of our co-skeletons is annotated with semantic information (color-coded in Fig. 2). For instance, all edges pertaining to the animals' heads, legs, or rump, have the same color. Such semantic information can next be used in a wide range of applications, such as shape matching, retrieval, or segmentation.

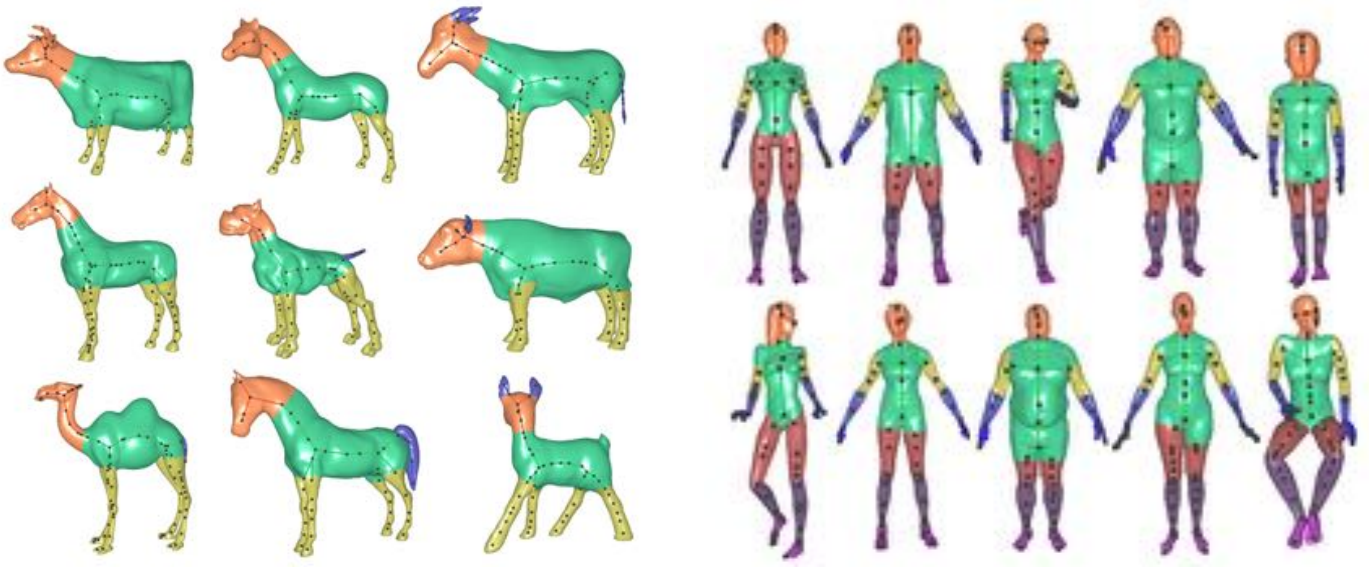
Figure 8 shows an additional result of our pruning: In most cases, even in the presence of semantic noise, our approach succeeds. Yet, in cases with significant semantic noise, our approach may fail. This is due to the fact that our method uses a voting mechanism that takes into account the semantic information of connected skeleton edges. One potential remedy is the use of a better, more robust, initial skeletonization method, than [8] or [7]. Any (existing or future) skeletonization method that accepts 3D meshes as input, and produces a polyline representation of the curve skeleton, together with the feature transform of its points, is directly applicable.

5.2. Co-skeleton Applications

We next aim to show the potential of co-skeletons by presenting two applications that may benefit from them: shape

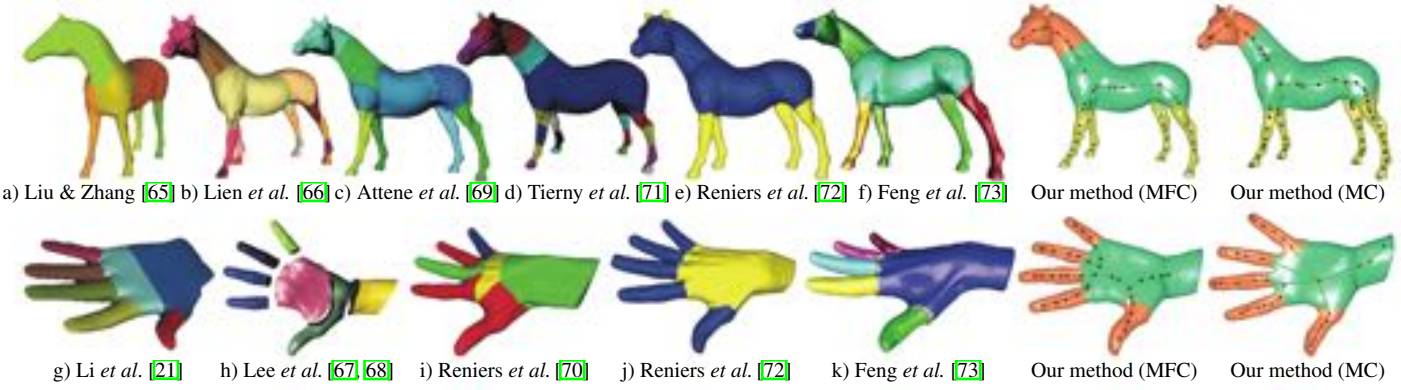
Table 1. Comparison of the average accuracy (Eqn. 8) of our co-segmentation results *vs* existing techniques [39, 43, 42, 46, 47]. We separate unsupervised techniques [39, 43, 42] and supervised ones [46, 47] for fair comparison. [46, 47] use only 6 training shapes in their experiments.

Shape category	Average accuracy per category					
	Ours	[39]	[43]	[42]	[46]	[47]
Human	**83.2**	-	70.4	78.0	-	-
Glasses	95.8	-	98.3	92.4	96.78	97.15
Airplane	80.2	-	83.3	79.6	95.56	93.90
Ant	88.1	-	92.9	90.1	-	-
Chair	93.8	85.0	89.6	87.6	97.93	97.05
Octopus	92.4	-	97.5	96.8	98.61	98.67
Table	98.6	-	99.0	98.4	99.11	99.25
Teddy	89.8	-	97.1	94.9	98.00	98.04
Hand	83.4	-	91.9	90.3	-	-
Plier	80.9	-	86.0	83.4	95.01	95.71
Fish	80.3	-	85.6	82.4	96.22	95.63
Bird	76.9	-	71.5	72.0	87.51	89.03
Armadillo	67.6	-	87.3	78.5	-	-
Fourleg	**92.1**	77.3	88.7	87.7	-	-
Candelabra	82.5	84.8	93.9	97.2	-	-
Lamp	91.1	94.1	90.7	98.4	-	-



(a) Fourleg dataset (b) Human dataset

Fig. 9. Segmentation results based on our co-skeletons. Different colors depict different semantic parts. The pruned (co-)skeletons inherently encode co-segmentation results. Further results can be found in Fig. 13 in the Appendix.



a) Liu & Zhang [65] b) Lien *et al.* [66] c) Attene *et al.* [69] d) Tierny *et al.* [71] e) Reniers *et al.* [72] f) Feng *et al.* [73] Our method (MFC) Our method (MC)

g) Li *et al.* [21] h) Lee *et al.* [67, 68] i) Reniers *et al.* [70] j) Reniers *et al.* [72] k) Feng *et al.* [73] Our method (MFC) Our method (MC)

Fig. 10. Comparison of our co-skeleton segmentation using MC and MCF skeletons with nine other segmentation methods.

co-segmentation and shape blending.

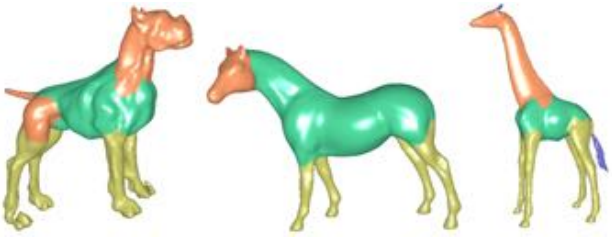
Co-segmentation: Since skeleton edges are inherently linked to collections of faces of the input shape(s), we can use the semantic information our algorithm produces to segment shapes. Like state-of-the-art co-segmentation approaches [39, 43, 42], we also use the graph cut algorithm [74] to optimize the boundaries of different segments. Figure 9 shows several segmentation results based on our co-skeletons for the *fourleg* (Fig. 9a) and *human* datasets (Fig. 9b). As visible, the produced segmentations are consistent, in the sense that different shapes (from the same family) get segmented at approximately the same level of detail — four limbs, rump, and head, for the *fourleg* shapes, and rump, head, legs (thigh, calf, foot), and hands (forearm, arm), respectively. However, details such as ears or horns for the *fourleg* shapes, are sometimes not separately segmented, for the shapes in which they are very small.

We next compare our co-segmentation results to five state-of-the-art methods [39, 43, 42, 46, 47]. Note that these are also co-segmentation methods which consider shape families rather than individual shapes. Similar to these methods, we measure the amount of area of a shape that is labeled correctly as

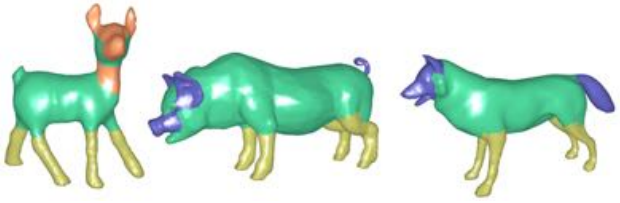
$$\mathrm{acc}(I) = \frac{\sum_i a_i \delta(c_i, t_i)}{\sum_i a_i}, \tag{8}$$

where a_i, c_i, and t_i are the area, label computed by our co-segmentation, and respectively ground-truth label of face i of a given shape I, and δ is Kronecker's delta.

Table 1 lists the accuracy values averaged per shape family for the PSB benchmark, with unsupervised and supervised methods reported separately. It shows that our co-segmentation achieves comparable results for this benchmark in the unsupervised group. However, we gain better performance for some shape families, *e.g.*, human and fourlegs, due to the semantic pruning step. Our results are driven by skeletons, so they contain both *skeleton* and *segmentation* information, in contrast to other pure segmentation approaches. We also see that supervised learning methods perform overall better than unsupervised ones. Yet, as said, supervised methods require significant training data, which we do not need. See also Fig. 11 for additional insights.



(a) Segmentation using the method of Sidi *et al.* [39]



(b) Segmentation using the method of Wu *et al.* [42]

Fig. 11. Segmentation results using existing methods can lead to wrong part prediction. Our method improves on these results (see Figure 9).

Finally, we compare our segmentation results with nine classical segmentation methods, which do not consider shape families (that is, which are not of the co-segmentation type). Figure 10 shows the results for the *hand* and *horse* shapes from the PSB benchmark. The rightmost two columns show our results obtained with co-skeletons constructed from Mesh Contraction (MC) [7], respectively Mean Curvature Flow (MCF) [8] base skeletons. We consider in the comparison both skeleton-based and surface-based segmentation methods, as follows. In the first class, Reniers *et al.* [70] detect curve skeleton junctions and use these to trace geodesic cuts to segment the parts of a shape. The method was further improved in [72] to reduce oversegmentation. Tierny *et al.* [71] segment shapes by analyzing their Reeb graphs, which are related to curve skeletons. Lien *et al.* [66] formulate (and solve) shape segmentation and curve-skeleton computation as a joint optimization problem. Feng *et al.* [73] extend the geodesic-cut-based segmentation in [70, 72] to surface skeletons, which encode both shape geometry and topology, thus provide more information for the segmentation. Finally, Li *et al.* [21] use mesh decimation methods for both shape segmentation but also their curve-skeleton computation.

In the second class, we have methods that segment shapes purely based on the information encoded by their surface. Lee *et al.* [67, 68] segment surface meshes using snake cuts which are optimized based on local mesh features such as curvature and excentricity. Attene *et al.* [69] segment shapes by fitting primitives from a given set (library). Liu and Zhang [65] encode the shape's faces into a similarity matrix which they then decompose by spectral clustering.

Overall, we see that our segmentation results (Fig. 10 rightmost two columns) compare very favorably with existing methods. In particular, our segment borders are smooth and wrap naturally around the shape, while this is not always the case for the other methods (see Fig. 10 c, h, i). Also, our co-skeletons ensure that there is no oversegmentation present, a phenomenon that can be observed for some of the other methods (Fig. 10 b, d, i). From these and other tested examples, we noticed that our segmentation method produces results most similar to the skeleton-cut method of Feng *et al.* (see Fig. 10 f, k). This can be explained by the fact that both methods optimize for smooth cuts, though with different mechanisms (Feng *et al.* use geodesic tracing; we use graph cuts). Also, both methods use skeletons to drive the segmentation. However, while we use *curve* skeletons which, as explained in Sec. 2 are simple and fast to compute, in particular by the MC and MCF methods that we use here, Feng *et al.* use *surface* skeletons, which are considerably slower and more complex to compute and analyze.

Figure 10 shows an additional insight: We see that our segmentations obtained by skeletons computed with two quite different methods (MC and MCF) are very similar. This is due to the fact that we do not use the *raw* skeletons for segmentation, but the *co-skeletons* which, as explained, stabilize skeletons over an entire family by removing outlier details. Further, this suggests that our segmentation approach based on co-skeletons does not strongly depend on the underlying skeletonization method. Hence, one can obtain similar segmentation results by substituting MC or MCF with other, better (*e.g.*, faster and/or easier to use) skeletonization methods.

Shape Blending: Besides shape segmentation, other applications also benefit from co-skeletons, including shape blending. Raw skeletons extracted using even state-of-the-art algorithms are typically inadequate for shape blending, due to the lack of semantic information on skeleton edges and/or over-sampling. To use skeletons, manual post-processing for cleaning and/or annotation is typically needed. In contrast, our co-skeletons can be directly used for shape blending.

We show this by using our co-skeletons to perform shape blending by the technique of Alhashim *et al.* [4]. We first use skeleton edges to reconstruct the spatio-structural graph, and augment this graph by constructing morphing paths between semantically-correlated parts/edges of different shapes of a family. This allows us to keep track of evolving states of the shapes and maintain the topological constraints needed for blending. Next, we select from the obtained results those which show plausible blends and combinations (this selection is done by the user based on what one actually deems to be plausible for a given application context). Finally, we reconstruct shapes based on the structure graphs through the feature transform (FT) mapping from skeleton edges to the shape faces. This yields new blended shapes within the input family. Figure 12 shows several examples of blended shapes, demonstrating the effectiveness of our co-skeletons for family-based shape blending.

6. Discussion and Conclusion

We have presented a novel approach to extracting co-skeletons of a given set of related shapes. In contrast to per-shape skeletons, our co-skeletons have similar quality, measured in terms of simplification level, centeredness, and preservation of details across all considered shapes in a family. Our method has two main use cases. First, we reduce the dependence on the availability of a high-quality skeletonization method, which may not

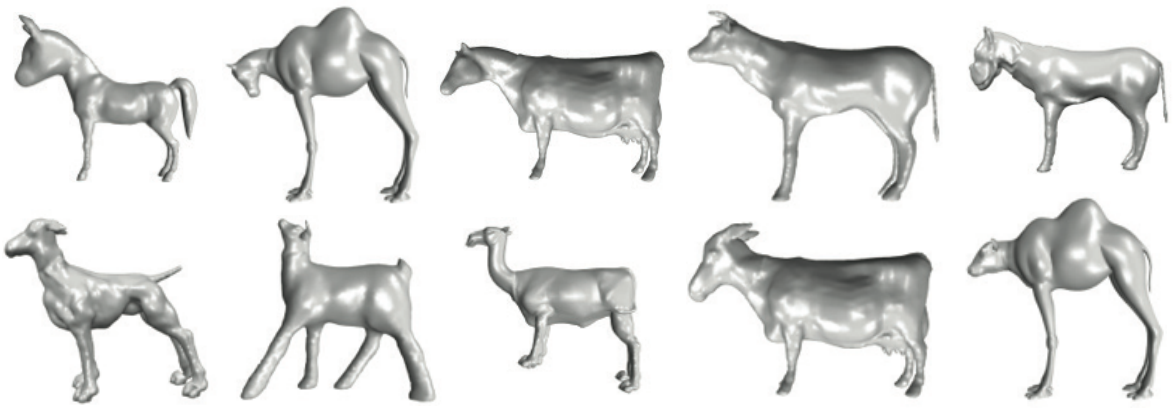


Fig. 12. Evolution results on the *fourleg* dataset. Using our co-skeletons, we can easily generate new shapes by evolving different combinations across each family. See Fig. 2 for the initial shapes in the family.

be easy to set up for any set of shapes. Secondly, we maximize the likelihood that the user obtains consistent skeletons over similar-type shapes with no additional parameter tweaking effort. We show the added value of co-skeletons on two applications: shape co-segmentation and shape blending.

While effective, easy to use, fast, and generic, our method has some limitations. First, we use multiple surface-based descriptors to infer the semantic relationships between skeleton edges by 'mapping' such edges to similar surface parts. Yet, the exact relation between specific types of skeleton fragments (*e.g.*, branch ends, junctions, or high-curvature zones) and specific surface details (*e.g.* edges, dents, tubular structures, or other detail types) is not yet fully clear. We aim to study this relationship in more detail, so we are next able to guarantee that the extracted co-skeletons account for specific shape-detail types that are of interest for users in specific applications.

Secondly, we used two curve-skeletonization methods [7, 8] to extract initial skeletons. It is important to study how our co-skeleton proposal behaves when using other curve-skeletonization methods (for a candidate set, see [9, 14]), so as to increase the confidence that our co-skeleton quality does not (strongly) depend on the choice of the underlying skeletonization method. In the long run, we aim to remove this dependency on a specific skeletonization method by extracting co-skeletons *directly* from a shape set. Thirdly, we used here a set of 5 shape descriptors (Sec. 3) which are well-known for related tasks in shape analysis literature. Whether other descriptors would perform better for our task, is an open question.

Finally, concerning the comparison with unsupervised shape segmentation methods, we should say that our co-segmentation benefits from additional information, present in the number of shape parts which is set by the user, which the aforementioned methods do not have. This shows, on the one hand, that adding such semantic information (which is available once we consider an entire shape *family*) benefits segmentation. On the other hand, this should not be seen as a limitation of unsupervised methods since these methods do not utilize such extra information.

Our co-skeleton validation is currently based on only two applications: shape co-segmentation and blending. While the initial results presented here are encouraging, it is of high added value to examine how co-skeletons work for other applications such as shape animation and morphing, and to evaluate our co-skeletons on other benchmark datasets, such as [75].

References

[1] Reniers, D, Telea, A. Skeleton-based hierarchical shape segmentation. In: Proc. SMI. 2007, p. 179–188.

[2] Le, BH, Deng, Z. Robust and accurate skeletal rigging from mesh sequences. ACM Trans Graph 2014;33(4):84:1–84:10.

[3] Yan, H, Hu, S, Martin, RR, Yang, Y. Shape deformation using a skeleton to drive simplex transformations. IEEE TVCG 2008;14(3):693–706.

[4] Alhashim, I, Li, H, Xu, K, Cao, J, Ma, R, Zhang, H. Topology-varying 3D shape creation via structural blending. ACM Trans Graph 2014;33(4):158:1–158:10.

[5] Manzanera, A, Bernard, TM, Prêteux, FJ, Longuet, B. Medial faces from a concise 3D thinning algorithm. In: Proc. ICCV. 1999, p. 337–343.

[6] Amenta, N, Choi, S, Kolluri, RK. The power crust. In: Proc. ACM SMA. 2001, p. 249–266.

[7] Au, OK, Tai, C, Chu, H, Cohen-Or, D, Lee, T. Skeleton extraction by mesh contraction. ACM Trans Graph 2008;27(3):44:1–44:10.

[8] Tagliasacchi, A, Alhashim, I, Olson, M, Zhang, H. Mean curvature skeletons. Comp Graph Forum 2012;31(5):1735–1744.

[9] Tagliasacchi, A, Delamé, T, Spagnuolo, M, Amenta, N, Telea, A. 3D skeletons: A state-of-the-art report. Comp Graph Forum 2016;35(2):573–597.

[10] Liu, P, Wu, F, Ma, W, Liang, R, Ouhyoung, M. Automatic animation skeleton construction using repulsive force field. In: Proc. IEEE Pacific Graphics. 2003, p. 409–413.

[11] Chen, X, Golovinskiy, A, Funkhouser, T. A benchmark for 3D mesh segmentation. ACM Trans Graph 2009;28(3):1–12.

[12] Sobiecki, A, Yasan, H, Jalba, A, Telea, A. Qualitative comparison of contraction-based curve skeletonization methods. In: Proc. ISMM. Springer; 2013,.

[13] Baran, I, Popovic, J. Automatic rigging and animation of 3D characters. ACM Trans Graph 2007;26(3):72.

[14] Sobiecki, A, Jalba, A, Telea, A. Comparison of curve and surface skeletonization methods for voxel shapes. Patt Rec Lett 2014;47:147–156.

[15] Blum, H. A transformation for extracting new descriptors of shape. In: Models for the perception of speech and visual form. 1967,.

[16] Au, OK, Tai, C, Cohen-Or, D, Zheng, Y, Fu, H. Electors voting for fast automatic shape correspondence. Comput Graph Forum 2010;29(2):645–654.

[17] Saha, P, Borgefors, G, di Baja, GS. A survey on skeletonization algorithms and their applications. Patt Recogn Lett 2016;(76):3–12.

[18] Cornea, ND, Silver, D, Min, P. Curve-skeleton properties, applications, and algorithms. IEEE TVCG 2007;13(3):530–548.

[19] Biasotti, S, Falcidieno, B, Spagnuolo, M. Extended Reeb graphs for surface understanding and description. In: Proc. DGCI. Springer; 2000, p. 185–197.

[20] Hassouna, MS, Farag, AA. Variational curve skeletons using gradient vector flow. IEEE Trans Patt Anal Mach Intell 2009;31(12):2257–2274.
[21] Li, X, Woon, TW, Tan, TS, Huang, Z. Decomposing polygon meshes for interactive applications. In: Proc. ACM SI3D. 2001, p. 35–42.
[22] Marini, S, Spagnuolo, M, Falcidieno, B. Structural shape prototypes for the automatic classification of 3d objects. IEEE Computer Graphics and Applications 2007;27(4):28–37.
[23] Cao, J, Tagliasacchi, A, Olson, M, Zhang, H, Su, Z. Point cloud skeletons via Laplacian based contraction. In: Proc. SMI. 2010, p. 187–197.
[24] Chuang, M, Kazhdan, MM. Fast mean-curvature flow via finite-elements tracking. Comput Graph Forum 2011;30(6):1750–1760.
[25] Jiang, W, Xu, K, Cheng, Z, Martin, RR, Dang, G. Curve skeleton extraction by coupled graph contraction and surface clustering. Graphical Models 2013;75(3):137–148.
[26] Shen, W, Bai, X, Hu, R, Wang, H, Latecki, LJ. Skeleton growing and pruning with bending potential ratio. Pattern Recog 2011;44(2):196–209.
[27] Ward, AD, Hamarneh, G. The groupwise medial axis transform for fuzzy skeletonization and pruning. IEEE TPAMI 2010;32(6):1084–1096.
[28] Bai, X, Latecki, LJ, Liu, W. Skeleton pruning by contour partitioning with discrete curve evolution. IEEE TPAMI 2007;29(3):449–462.
[29] Liu, H, Wu, Z, Zhang, X, Hsu, DF. A skeleton pruning algorithm based on information fusion. Pattern Recog Letters 2013;34(10):1138–1145.
[30] Reniers, D, van Wijk, JJ, Telea, A. Computing multiscale curve and surface skeletons of genus 0 shapes using a global importance measure. IEEE TVCG 2008;14(2):355–368.
[31] Barbieri, S, Meloni, P, Usai, F, Scateni, R. Skeleton lab: an interactive tool to create, edit, and repair curve-skeletons. In: Proc. Smart Tools and Apps for Graphics – Eurographics Italian Chapter. 2015, p. 121–128.
[32] Dey, T, Sun, J. Defining and computing curve-skeletons with medial geodesic function. In: Proc. ACM SGP. 2006, p. 143–152.
[33] Jalba, A, Kustra, J, Telea, A. Surface and curve skeletonization of large 3D models on the GPU. IEEE TPAMI 2013;35(6):1495–1508.
[34] Jalba, A, Sobiecki, A, Telea, A. An unified multiscale framework for planar, surface, and curve skeletonization. IEEE TPAMI 2015;38(1):38–45.
[35] Schaefer, S, Yuksel, C. Example-based skeleton extraction. In: Proc. SGP. Eurographics; 2007, p. 153–162.
[36] Zheng, Q, Sharf, A, Tagliasacchi, A, Chen, B, Zhang, H, Sheffer, A, et al. Consensus skeleton for non-rigid space-time registration. Comput Graph Forum 2010;29(2):635–644.
[37] Jerripothula, KR, Cai, J, Lu, J, Yuan, J. Object co-skeletonization with co-segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR. IEEE Computer Society; 2017, p. 3881–3889.
[38] Kalogerakis, E, Hertzmann, A, Singh, K. Learning 3D mesh segmentation and labeling. ACM Trans Graph 2010;29(4):1–11.
[39] 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 Graph 2011;30(6):126:1–126:9.
[40] Chaudhuri, S, Kalogerakis, E, Guibas, LJ, Koltun, V. Probabilistic reasoning for assembly-based 3D modeling. ACM Trans Graph 2011;30(4).
[41] Golovinskiy, A, Funkhouser, TA. Consistent segmentation of 3D models. Computers & Graphics 2009;33(3):262–269.
[42] Wu, Z, Wang, Y, Shou, R, Chen, B, Liu, X. Unsupervised co-segmentation of 3D shapes via affinity aggregation spectral clustering. Computers & Graphics 2013;37(6):628–637.
[43] Hu, R, Fan, L, Liu, L. Co-segmentation of 3D shapes via subspace clustering. Comput Graph Forum 2012;31(5):1703–1713.
[44] Wu, Z, Shou, R, Wang, Y, Liu, X. Interactive shape co-segmentation via label propagation. Computers & Graphics 2014;38:248–254.
[45] Wang, Y, Asafi, S, van Kaick, O, Zhang, H, Cohen-Or, D, Chen, B. Active co-analysis of a set of shapes. ACM ToG 2012;31(6):165:1–10.
[46] Guo, K, Zou, D, Chen, X. 3D mesh labeling via deep convolutional neural networks. ACM Trans Graph 2015;35(1):3:1–3:12.
[47] Wang, P, Gan, Y, Shui, P, Yu, F, Zhang, Y, Chen, S, et al. 3d shape segmentation via shape fully convolutional networks. Comput Graph 2018;70:128–139.
[48] Laga, H, Mortara, M, Spagnuolo, M. Geometry and context for semantic correspondences and functionality recognition in man-made 3d shapes. ACM Trans Graph 2013;32(5):150:1–150:16.
[49] van Kaick, O, Xu, K, Zhang, H, Wang, Y, Sun, S, Shamir, A, et al. Co-hierarchical analysis of shape structures. ACM Trans Graph 2013;32(4):69:1–69:10.
[50] Yümer, ME, Kara, LB. Co-abstraction of shape collections. ACM Trans Graph 2012;31(6):166:1–166:11.
[51] Xu, K, Zhang, H, Cohen-Or, D, Chen, B. Fit and diverse: set evolution for inspiring 3D shape galleries. ACM ToG 2012;31(4):57:1–57:10.
[52] Kim, VG, Li, W, Mitra, NJ, Chaudhuri, S, DiVerdi, S, Funkhouser, TA. Learning part-based templates from large collections of 3D shapes. ACM Trans Graph 2013;32(4):70:1–70:12.
[53] Fish, N, Averkiou, M, van Kaick, O, Sorkine-Hornung, O, Cohen-Or, D, Mitra, NJ. Meta-representation of shape families. ACM Trans Graph 2014;33(4):34:1–34:11.
[54] Yümer, ME, Kara, LB. Co-constrained handles for deformation in shape collections. ACM Trans Graph 2014;33(6):187:1–187:11.
[55] Hesselink, W, Roerdink, J. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. IEEE TPAMI 2008;30(12):2204–2217.
[56] Wu, Z, Zeng, M, Qin, F, Wang, Y, Kosinka, J. Active 3-d shape cosegmentation with graph convolutional networks. IEEE Computer Graphics and Applications 2019;39(2):77–88.
[57] Shapira, L, Shalom, S, Shamir, A, Cohen-Or, D, Zhang, H. Contextual part analogies in 3D objects. Intl J on Comp Vision 2010;89(1-2):309–326.
[58] Ben-Chen, M, Gotsman, C. Characterizing shape using conformal factors. In: Proc. 3DOR. Eurographics; 2008, p. 1–8.
[59] Belongie, S, Malik, J, Puzicha, J. Shape matching and object recognition using shape contexts. IEEE TPAMI 2002;24(4):509–522.
[60] Hilaga, M, Shinagawa, Y, Komura, T, Kunii, TL. Topology matching for fully automatic similarity estimation of 3D shapes. In: Proc. ACM SIGGRAPH. 2001, p. 203–212.
[61] Rubner, Y, Tomasi, C, Guibas, LJ. The earth mover's distance as a metric for image retrieval. Intl J Computer Vision 2000;40(2):99–121.
[62] von Luxburg, U. A tutorial on spectral clustering. Statistics and Computing 2007;17(4):395–416.
[63] Huang, HC, Chuang, YY, Chen, CS. Affinity aggregation for spectral clustering. In: Computer Vision and Pattern Recognition (CVPR). IEEE; 2012, p. 773–780.
[64] van der Maaten, L, Hinton, G. Visualizing data using t-sne. Journal of Machine Learning Research 2008;9:2579–2605.
[65] Liu, R, Zhang, H. Segmentation of 3D meshes through spectral clustering. In: Proc. IEEE Pacific Graphics. 2004, p. 298–305.
[66] Lien, J, Keyser, J, Amato, N. Simultaneous shape decomposition and skeletonization. In: Proc. ACM SPM. 2006, p. 219–228.
[67] Lee, Y, Lee, S, Shamir, A, Cohen-Or, D. Intelligent mesh scissoring using 3D snakes. In: Proc. IEEE Pacific Graphics. 2004, p. 279–287.
[68] Lee, Y, Lee, S, Shamir, A, Cohen-Or, D, Seidel, HP. Mesh scissoring with minima rule and part salience. CAGD 2005;22:444–465.
[69] Attene, M, Falcidieno, B, Spagnuolo, M. Hierarchical mesh segmentation based on fitting primitives. Visual Computer 2006;22:181–193.
[70] Reniers, D, Telea, A. Hierarchical part-type segmentation using voxel-based curve skeletons. Visual Computer 2008;24:383–395.
[71] Tierny, J, Vandeborre, J, Daoudi, M. Topology driven 3D mesh hierarchical segmentation. In: Proc. SMI. 2007, p. 215–220.
[72] Reniers, D, Telea, A. Part-type segmentation of articulated voxel shapes using the junction rule. CGF 2008;27(3):1845–1852.
[73] Feng, C, Jalba, A, Telea, A. Improved part-based segmentation of voxel shapes by skeleton cut spaces. Math Morphol Theory Appl 2015;1:1–20.
[74] Boykov, Y, Veksler, O, Zabih, R. Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Anal Mach Intell 2001;23(11):1222–1239.
[75] Lavoué, G, Vandeborre, JP, Benhabiles, H, Daoudi, M, Huebner, K, Mortara, M, et al. SHREC'12 track: 3D mesh segmentation. In: Proc. 3DOR. 2012,.

Appendix

Here we showcase our method on further results. Fig. 13 presents (co-)segmentation results based on our co-skeletons building on the per-shape input skeletons computed using Mean Curvature Flow [8] and Mesh Contraction [7].

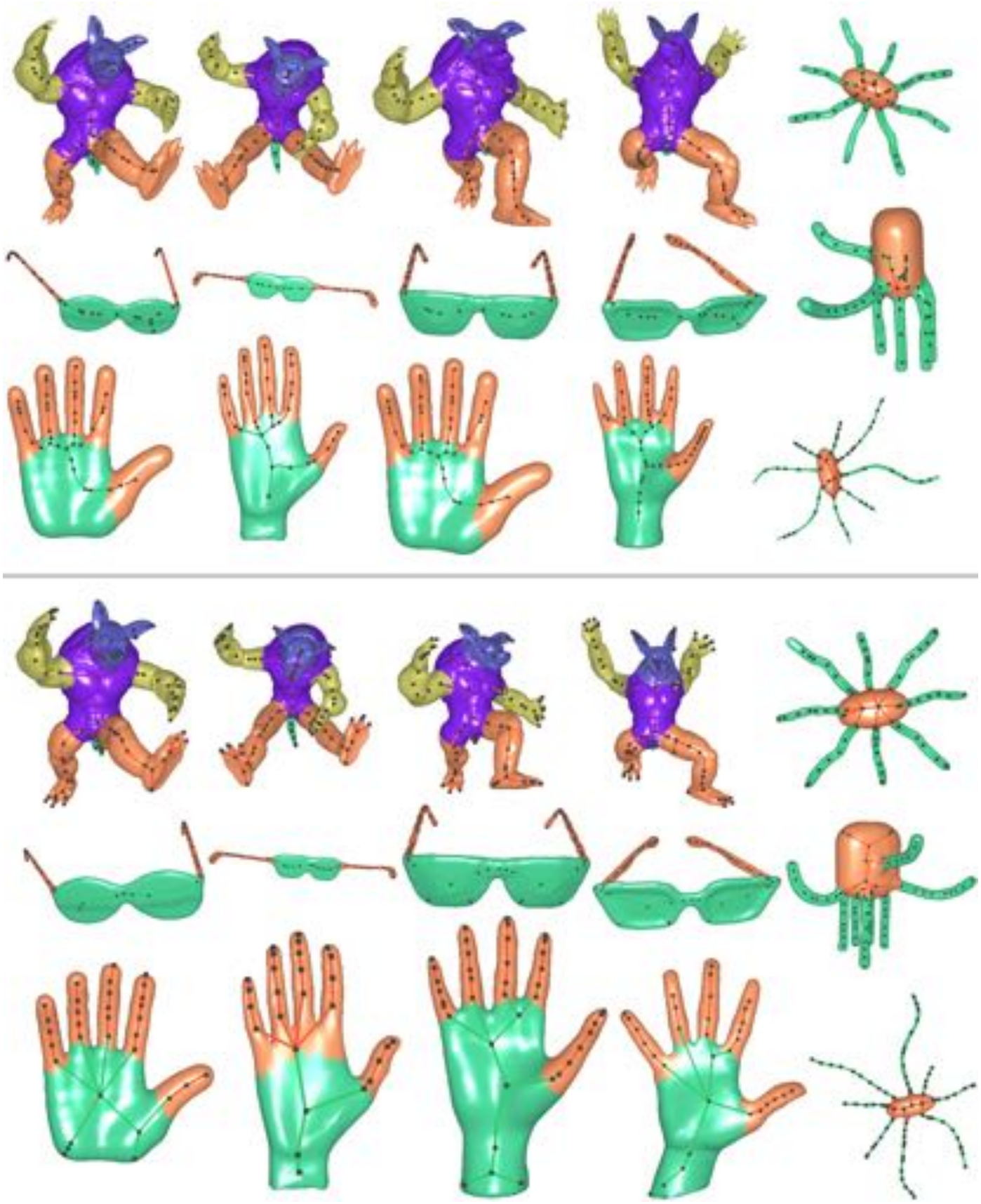


Fig. 13. Co-skeletonization and co-segmentation results of our method based on Mean Curvature Flow [8] (top half) and Mesh Contraction [7] (bottom half).