Graphical Models 75 (2013) 137–148

Contents lists available at SciVerse ScienceDirect

Graphical Models

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

Curve skeleton extraction by coupled graph contraction and surface clustering

Wei Jiang [a], Kai Xu [a,*], Zhi-Quan Cheng [a,*], Ralph R. Martin [b], Gang Dang [a,*]

[a] School of Computer, National University of Defense Technology, Changsha City, Hunan Province 410073, China
[b] School of Computer Science and Informatics, Cardiff University, Cardiff, Wales CF24 3AA, UK

ARTICLE INFO

Article history:
Received 30 August 2012
Received in revised form 10 October 2012
Accepted 26 October 2012
Available online 10 November 2012

Keywords:
Curve skeleton
Voronoi
Graph contraction
Surface clustering
Deformation

ABSTRACT

In this paper, we present a practical algorithm to extract a curve skeleton of a 3D shape. The core of our algorithm comprises coupled processes of graph contraction and surface clustering. Given a 3D shape represented by a triangular mesh, we first construct an initial *skeleton graph* by directly copying the connectivity and geometry information from the input mesh. Graph contraction and surface clustering are then performed iteratively. The former merges certain graph nodes based on computation of an approximate centroidal Voronoi diagram, seeded by subsampling the graph nodes from the previous iteration. Meanwhile, a coupled surface clustering process serves to regularize the graph contraction. Constraints are used to ensure that extremities of the graph are not shortened undesirably, to ensure that skeleton has the correct topological structure, and that surface clustering leads to an approximately-centered skeleton of the input shape. These properties lead to a stable and reliable skeleton graph construction algorithm.

Experiments demonstrate that our skeleton extraction algorithm satisfies various desirable criteria. Firstly, it produces a skeleton homotopic with the input (the genus of both shapes agree) which is both robust (results are stable with respect to noise and remeshing of the input shape) and reliable (every boundary point is visible from at least one curve-skeleton location). It can also handle point cloud data if we first build an initial skeleton graph based on k-nearest neighbors. In addition, a secondary output of our algorithm is a skeleton-to-surface mapping, which can e.g. be used directly for skinning animation.

Highlights: (1) An algorithm for curve skeleton extraction from 3D shapes based on coupled graph contraction and surface clustering. (2) The algorithm meets various desirable criteria and can be extended to work for incomplete point clouds.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

A skeleton is a compact and effective representation of a solid shape [1–3], efficiently encoding both the shape's geometry and topology. Thus, skeletal representations find a wide range of applications in areas such as surface manipulation [4–6], surface reconstruction [7,8], shape matching [9], and computer animation [10].

Skeleton extraction has been extensively studied. Many approaches have been proposed for computing medial structures, which in general can either be a 2D medial surface [11,12], or a 1D curve skeleton [1,13]. In this paper, we aim to extract curve skeleton, as it is a more concise representation, with generally wider application.

The curve skeleton is a one-dimensional structure that represents a simplified version of the geometry of a shape, and is not uniquely defined. However, a good skeleton should possess certain desirable properties, including that it should be: (i) homotopic to the input shape (i.e. have the same topology), (ii) well-centered, and (iii) reliable in that

* Corresponding authors.
E-mail addresses: kaixu@nudt.edu.cn (K. Xu), zqcheng@nudt.edu.cn (Z.-Q. Cheng), gangdang@nudt.edu.cn (G. Dang).

1524-0703/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.gmod.2012.10.005

every boundary point is visible from at least one curve-skeleton location. The method should also be robust to noise and variations in tessellation of the input shape.

An intuitive approach to computation of a curve skeleton is to iteratively shrink the shape to a curve. this is done, for example, by the collapsing algorithm of Li et al. [14], which constructs a line segment skeleton by collapsing mesh edges in the order of edge length. However, this method is sensitive to the mesh tessellation. More recently, Au et al. [15] proposed Laplacian surface contraction approach to achieve a high quality curve skeleton, but this method works only for closed manifold meshes.

In [16], we introduced a novel curve skeleton extraction algorithm by graph contraction. The algorithm produces reasonable output on various input closed meshes, overcoming the limitation of traditional Voronoi-based approaches [12,17] that they produce a medial surface rather than a curve skeleton. In this paper, we extend the constrains on graph contraction, discuss the properties of our algorithm more thoroughly, and show how to extend our method to extract curve skeletons from incomplete point clouds [8,18].

Our algorithm is based on coupled processes of graph contraction and surface clustering. Graph contraction is guided by computation of an *approximate centroidal Voronoi diagram* (ACVD) [17], which provides a fast and efficient simplification and clustering algorithm for 3D surfaces. Although in the 2D case, the Voronoi diagram is an effective approximation paradigm for producing a curve skeleton [19], Amenta et al. [20,12,3] pointed out that the direct extension of this approach to 3D shapes is not trivial. Like other Voronoi-based algorithms [12], ACVD produces a medial surface, rather than a medial curve, and further processing is needed to reach a curve skeleton.

Graph contraction iteratively contracts a *skeleton graph* while performing surface clustering on the input shape in a coupled manner. Given a 3D shape represented by a triangular mesh, we first initialize the skeleton graph by directly copying the input mesh. Graph contraction reduces the number of graph nodes, while surface clustering adjusts the nodes' positions for the next iteration. The skeleton graph thus gradually contracts to a curve skeleton for the input shape. Surface clustering also plays an important role during skeletonization in helping to ensure that the result is a skeleton curve rather than a medial surface. Surface clustering geometrically positions the skeleton graph nodes in accordance with cluster centers, while adjacency of clusters is used to determine edges and hence connectivity of the skeleton graph.

The principal output is an approximately-centered curve skeleton. A secondary output of our algorithm is a skeleton-to-surface mapping, which is directly useful in applications such as skinning mesh deformation. We also note that our method is extensible to point clouds if we first build an initial skeleton graph using the k-nearest neighbors (k-NN) method. Our tests on a range of 3D shapes show that our method possesses the various desirable properties mentioned above.

2. Related work

Curve skeleton extraction and its applications have been studied both theoretically and algorithmically. One way of classifying approaches to the problem is as *volumetric* or *geometric* [15], according to whether an interior representation or only a surface representation of the object is used. We focus on the most relevant *geometric* methods, and recent related advances; see [1,2,15] for comprehensive reviews.

Direct *volumetric* approaches are based on thinning [11,21,22]. They produce a curve skeleton by iteratively removing voxels from the boundary of an object until the required thinness is obtained. Most volumetric approaches are based on the distance-field [22–29]. The distance field (or distance transform) [24,25] is defined over the interior of a 3D shape as the smallest distance from a given point to the surface. Other types of volumetric fields [23,26] have also been used to extract curve skeletons.

Geometric methods can be applied to objects represented by polygonal meshes or point set surfaces, e.g. [14,15,19,30–33]. A popular approach is to use the Voronoi diagram [19,31], generated form the vertices of the 3D polygonal representation or directly from a set of unorganized points [12]. Our algorithm is related to these approaches but uses the *approximate centroidal Voronoi diagram* (ACVD) [17] to speed up the computation. A significant difference from these methods, however, is that we produce a curve skeleton rather than the straightforward medial surface as done in [12,17].

The Reeb graph captures the topology of a manifold by following the evolution of level sets of a real-valued function, and can also be used to build the curve-skeleton [34]. The real-value functions used can take various forms, such as the geodesic distance [35] or harmonic [36] functions.

The observation that Laplacian smoothing contracts a mesh was first employed by Au et al. [15] to find skeletons of mesh surfaces, and extended to handle point sets by Cao et al. [33]. Tagliasacchi et al. [44] improved this contraction-based method through incorporating Voronoi pole structure into mean curvature flow. Tagliasacchi et al. [8] proposed the notion of generalized rotational symmetry axis (ROSA) as a basis for extracting the skeleton for an oriented, incomplete point cloud. Our algorithm can also handle point clouds, even in the presence of incomplete data (see [8,18]).

Other geometric methods use different approaches to compute the curve skeleton, and several share some similarities with the ideas presented here. Li et al. [14] construct a line segment skeleton by edge collapse. Katz and Tal [30] first decompose a mesh surface into segments using minimal curvature cues and fuzzy clustering, and then use this segmentation to construct a skeleton. In the work of [37], skeleton extraction and mesh decomposition are performed simultaneously, using approximate convex decomposition. These approaches exploit the relationship between skeletonization and decomposition [38]. Although we also perform mesh decomposition, i.e., surface clustering based on ACVD, the fundamental difference between these algorithms and ours is that we utilize graph contraction to extract the skeleton, rather than generating a shape decomposition and then subsequently extract the skeleton.

Other recent work has considered use of skeleton *examples* [39–41], especially for articulated shapes. Example-based algorithms can take multiple examples of existing skeletons into account, thus improving the robustness of skeleton determination. Such methods also deliver pose information as well as the skeleton, which is of benefit in applications such as animation [5,41]. As already noted, a secondary output of our surface clustering-based method is a skeleton to surface mapping, which is also of use to downstream applications.

3. Curve skeleton extraction algorithm

Given a triangle mesh, we extract its curve skeleton using coupled processes of graph contraction and surface clustering. We first summarize our algorithm, then in the following sections, we further explain these components, including the graph contraction and surface clustering processes, as well as an optional skeleton refinement step.

3.1. Overview

Algorithm 1 gives pseudocode for our algorithm. Let us denote the input mesh by $\mathcal{M}$, and the skeleton graph by $\mathcal{G}$. The latter is initialized as a direct copy of $\mathcal{M}$. The surface clusters are denoted by Ω, where initially each vertex of $\mathcal{M}$ is put in an independent cluster. The skeleton graph and clustering are consistent in the sense that the clustering Ω determines the graph $\mathcal{G}$, and the mapping from the mesh $\mathcal{M}$ to the skeleton $\mathcal{G}$. During iterative skeletonization, we perform coupled graph contraction and surface clustering. Graph contraction operates on $\mathcal{G}$, and works in a similar way to ACVD [17], but incorporates constraints to ensure the graph is not over-contracted. This progressively coarsens and contracts the initial graph into a medial structure of the input mesh. This graph contraction is coupled with a surface clustering process, which operates on the surface clusters Ω on $\mathcal{M}$. Surface clustering serves to group mesh regions, but also controls graph node positions, and connectivity of the skeleton graph, for the graph contraction step on the next iteration. Iteration stops when no further contraction is permissible. If the user requires a more detailed skeleton and surface clusters, an optional refinement process may be used which places additional graph nodes between those already determined, and accordingly refines the clustering. As well as giving denser sampling, this can also help to provide better distributed skeleton points. Overall, the skeleton graph is contracted into a final curve skeleton. Fig. 1 shows several iterations of skeletonization (a–d) and skeleton refinement (e).

Algorithm 1. Curve skeleton extraction by coupled graph contraction and surface clustering

Require: A mesh $\mathcal{M}$
Ensure: Its skeleton $\mathcal{K}$ and surface clusters $\Omega = \{C_i | i = 1, \ldots, n\}$
1: Initialize skeleton graph $\mathcal{G}$ from $\mathcal{M}$;
2: Initialize surface clusters $\Omega = \{C_i | i = 1, \ldots, N_V\}$;
3: $N_{seeds} = 200$;
4: **repeat**
5: $N_{seeds} = r \times N_{seeds}$;
6: Graph contraction on $\mathcal{G}$ using N_{seeds};
7: Surface clustering on Ω;
8: Update contraction and clustering constraints;
9: **until** (no contractible node in $\mathcal{G}$);
10: $\mathcal{K} \leftarrow \mathcal{G}$;
11: Optionally refine skeleton $\mathcal{K}$;
Note: C_i is the ith cluster. N_V is the number of vertices of $\mathcal{M}$. $N_{seeds} = 200$ was found to work well for all tested models, r is the cluster reduction ratio, experimentally set to 0.7. No contractible node in $\mathcal{G}$ means that there is no non-topological triangle in $\mathcal{G}$.

Our contraction idea is enlightened by the observation that AVCD can simplify the input mesh and generate a medial structure [17]. ACVD mimics CVD via k-means clustering of mesh elements (vertices) to coarsen the input mesh. However, when ACVD is used in a straightforward way, it produces a medial *surface*-like mesh as the result, as do other Voronoi-based methods. For example, Fig. 2 shows medial structures for similar hand models computed using typical Voronoi-based approaches, including the *power crust* approach [12] and ACVD [17]: in both cases a *three-dimensional* medial skeleton structure or surface results. Our goal is to produce a *one-dimensional* curve skeleton, as again shown in Fig. 2. We thus have to revise AVCD.

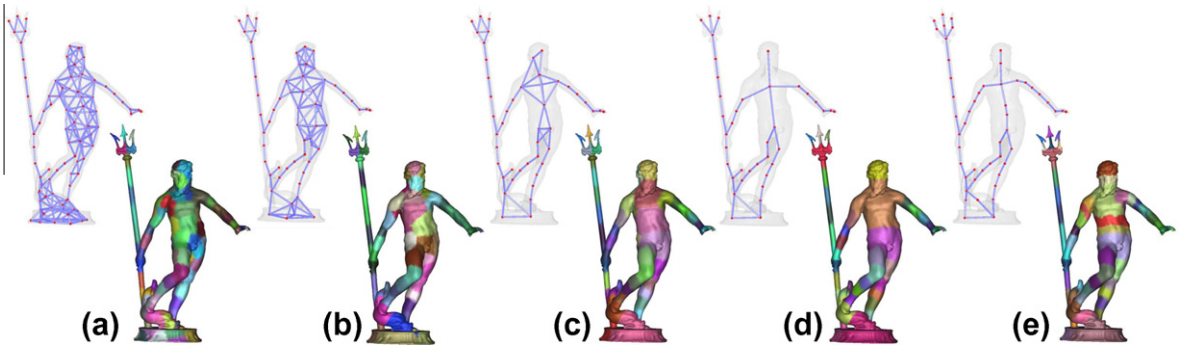


Fig. 1. From (a) to (d): Four iterations of the coupled graph extraction and surface clustering on a Neptune model. The resulting curve skeleton can be further refined if necessary to give a denser sampling (e).

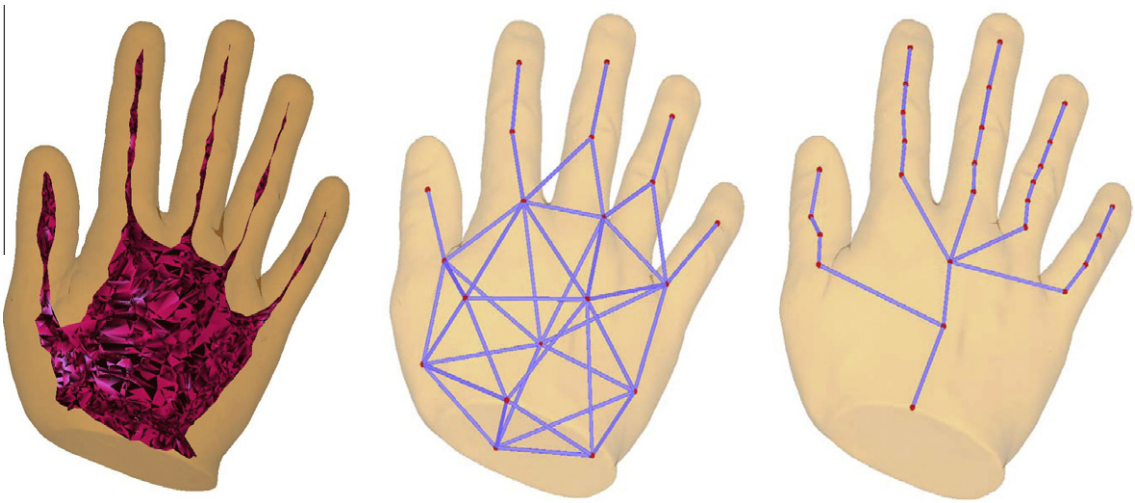


Fig. 2. Hand model medial structure, using, left to right: power crust [12] (image produced by Amenta et al. [12]), ACVD [17] and our approach.

3.2. Coupled graph contraction and surface clustering

The graph contraction and surface clustering are coupled steps, as illustrated in Fig. 3. The position of each node in the skeleton graph is computed as the (3D) center of mass of its corresponding surface cluster. The edges of the graph are induced by the adjacency relations of the surface clusters. In each iteration, we first perform skeleton graph contraction using a subset of the current graph nodes as seeds. During contraction, graph nodes are associated with the surface clusters obtained in the last iteration. A contraction of two graph nodes causes their associated surface clusters to be merged. The new surface clusters are then optimized via surface clustering, which reallocates some triangles to different clusters. The skeleton graph (both its nodes and edges) is then revised to agree with the optimized surface clusters. The purpose of this optimization process is to adjust the contracted graph to provide a better distribution of nodes.

To perform skeleton graph contraction, we utilize a revised form of ACVD on $\mathcal{G}$ in which we minimize the following energy:

$$F_{\mathcal{G}} = \sum_{i=0}^{n-1} \left(\sum_{n_i \in C_j} \omega_i \| n_i - c_j \|^2 \right), \tag{1}$$

where n_i is a graph node within the Voronoi cell C_j with center of mass c_i, the weight of node n_i is $\omega_i = 1/\sum_{j \in N_1(i)} A_{ij}$, $N_1(i)$ is the 1-ring neighborhood of triangles of node n_i and A_{ij} is the area of the corresponding triangle. If a node has no neighboring triangle, its weight is set to 10^6. Clustering is performed using the classical Lloyd relaxation method, interleaving between cluster growing and center relocation. The process is bootstrapped with a set of seed points and the initial clusters associated with each seed point. The initial clusters are obtained by grass-fire region growing from each seed. We then iteratively check for each boundary edge $e_{ij} = (n_i, n_j)$ whether its two end nodes lie in different clusters: $n_i \in C_1$ and $n_j \in C_2$. We then perform one of the following three operations according to which decreases the energy $F_{\mathcal{G}}$ most: (1) merge n_i into cluster C_2, (2) merge n_j into cluster C_1, or (3) keep the current configuration. When there is no further boundary edges can be updated, the contraction step is finished and we move to surface clustering. On each outer iteration the number of seeds reduces by a factor r; we experimentally take $r = 0.7$ in our implementation for all models.

The surface clustering uses a similar process to the above except that it operates over the clusters Ω on the original mesh. The energy function to be optimised is that

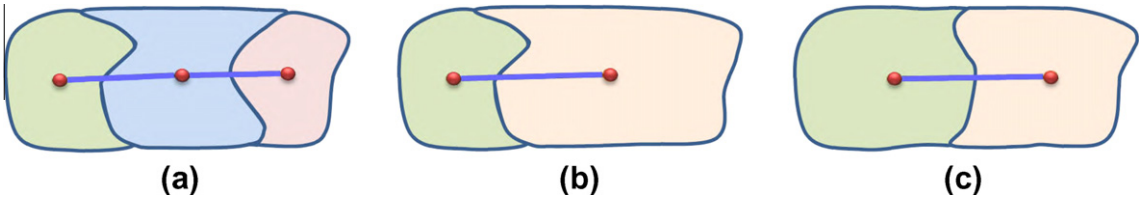


Fig. 3. Coupled graph contraction and surface clustering. Nodes (colored circles) of the skeleton graph are located at the centers of mass of corresponding surface clusters (shaded regions). Edges (blue segments) are induced by adjacency relations of the clusters. Clustering is first performed on the skeleton graph (a). The surface clusters are inherited and merged during skeleton graph contraction (b). We then optimize the surface clusters and update the skeleton graph, along with the new clusters (c). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in Eq. (2), which has similar form to Eq. (1) (see later), as both are variations of basic ACVD definition. Coupled contraction and clustering are iterated until no contractible nodes remain, i.e., all remaining nodes are selected as seeds and no mesh triangles can be removed due to the topological constraints.

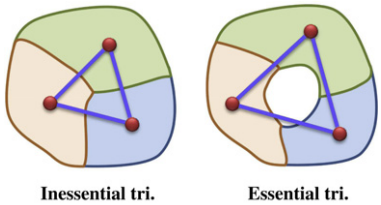
Note that directly performing ACVD over the skeleton graph would simply coarsen the graph. In order to eventually make it converge to an approximately-centered skeleton, we need to add constraints to both the graph contraction and surface clustering processes. Firstly, in order to avoid over-contraction, we prevent certain nodes of the skeleton graph from being removed, both to avoid shrinkage of extremities of the skeleton, and to avoid undesirable topological changes. Secondly, in order to make the graph lie near the center of the interior of the input surface, we constrain surface clustering to favor increasing eccentricity of the center of mass of the resulting clusters.

3.2.1. Constraints on graph contraction

A common issue in most contraction-based skeleton extraction approaches is over-contraction, where parts of the skeleton corresponding to surface details (e.g. thin structures) may be lost [15,42]. Addressing this issue is especially important in our algorithm since over-clustering may lead to incorrect topology. The two examples in Fig. 4 show unwanted shrinkage of the extremities of the skeleton, and loss of topological details respectively, both of which are caused by over-contraction: the skeleton does not go all the way down the legs of the horse, and topological handles of the Heptoroid model are merged. Corresponding correct results can be found in Fig. 6.

To avoid both kinds of over-contraction, we identify certain critical nodes and include them as seeds for clustering during each iteration step, which prevents them from being removed during contraction. Identifying such critical nodes is in general a non-trivial task. We resort to a heuristic approach.

Since the skeleton graph is computed by Delauney triangulation of the ACVD, its basic elements are triangles. Some triangles surround a topological handle, so we call them *essential triangles* (shown on the far right). Most triangles are, however, formed from three neighboring surface clusters, and we refer to them as *inessential triangles* (shown on the near right). The goal of controlled graph contraction is to remove all inessential triangles while meeting any further constraints.



We first identify all essential triangles by looking at the vertices between neighboring clusters. If a vertex is shared by at least three mutually neighboring clusters, then the triangle is inessential, otherwise it is essential. We ensure that *every* vertex of an essential triangle is selected as a seed. For inessential triangles, we make sure that exactly *one* vertex is selected for each triangle. In practice, to select seeds from inessential triangles, we maintain a priority queue for all unselected graph nodes in decreasing order of node degree. We select the top node in the priority queue and remove all its 1-ring neighboring nodes from the queue. Then we update the priority queue and repeat the above process until no node is left in the queue. If the seeds selected so far are fewer than the required number (recall that the number of seeds in the current step is 0.7 of the number in the previous one), the remaining seeds are selected randomly.

Further constraints are imposed to prevent shrinkage of the free ends of the skeleton, such as at the extremities of the legs of the horse model. We do this by ensuring that nodes with only one or two incident edges in the skeleton graph are also selected as seed nodes. Such isolated edges do not have any incident triangles, and thus belong to a 1D structure representing part of a skeleton. Selecting such nodes as seeds protects these edges from being further contracted.

3.2.2. Eccentricity control on surface clustering

The output skeleton should be as central as possible in the shape. To meet this goal, eccentricity control is imposed during surface clustering: surface clusters determines the positions of the skeleton nodes. We do so by constraining the growing direction of the surface clusters. Specifically, surface clustering optimizes the following energy function:

$$F_{\mathcal{G}} = \sum_{i=0}^{n-1} \left(\sum_{n_i \in C_j} \omega_i \| n_i - c_j \|^2 \right) + 1/d^2(c_j, C_j) \qquad (2)$$

where $d(c_j, C_j)$ is the Euclidean distance from a cluster C_j's center of mass c_j to the surface patch of C_j. The second

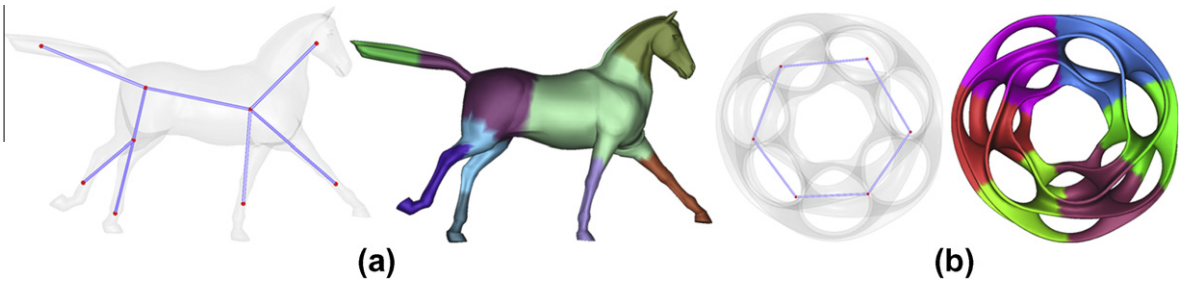


Fig. 4. Results without constraint. The horse legs are over-shrunk (a) and the genuses of the Heptoroid model are lost (b).

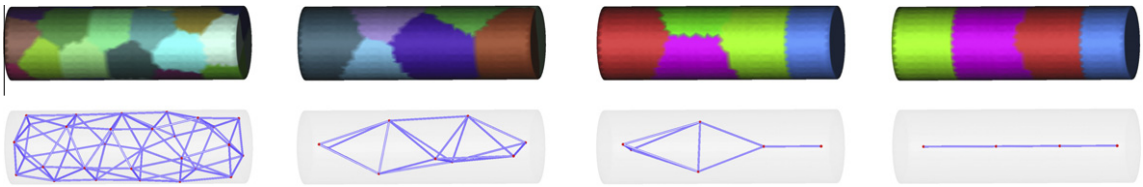


Fig. 5. Four iterations in the coupled graph extraction and surface clustering on a cylinder. Note how the surface clustering forms a serial of cylindrical clusters.

energy term is highly nonlinear. Fortunately, it can be optimized with by Lloyd relaxation. Specifically, in each iteration of Lloyd's algorithm, merging of boundary edges is not only determined by minimization of ACVD energy, but also by maximization of the distance from the current center to the surface cluster. Note that an approximate point to surface distance suffices for this purpose. In our implementation, we simply find the closest vertex on the surface mesh using a *k*-D-tree and compute the distance as the minimum distance from the center to all 1-ring neighbouring triangles of the closest vertex. Since skeleton nodes should be located inside the surface mesh, the point to surface distance should only take into account interior distance. This is approximately achieved by filtering the back-facing triangles (w.r.t. the center) based on normal. Our simple approach to eccentricity control is merely one possible solution. Other more complicated constraints can be used given within the optimization framework of Lloyd relaxation.

Fig. 5, shows that the effect of eccentricity control is to cause anisotropic growth of the surface clusters. Ultimately, the input cylinder is decomposed into a series of smaller cylinders whose centers of mass can be concatenated into a skeleton. A similar phenomenon can also be observed for more general shapes like those shown in this paper: a skeleton can be locally seen as the axis of a generalized cylinder corresponding to a particular surface cluster. This is further confirmed by the important observation that a curve skeleton can be locally thought of as a generalized rotational symmetry axis of a shape [8].

3.3. Skeleton refinement

We can optionally perform geometric refinement to upsample the skeleton by inserting further graph nodes into the skeleton graph. To do so, given two neighboring graph nodes, we extract all the mesh edges shared by the two surface clusters of the two nodes to form a new cluster. Then we compute the center of mass of the new cluster and insert the node (along with the corresponding surface cluster) inbetween the two graph nodes. To optimize the location of the newly inserted node, we perform local optimization of the newly inserted cluster by minimizing the energy in Eq. (1). However, differently from the procedure in Section 3.2, we perform grassfire growing where we grow the boundary one ring at a time instead of one

Table 1
Timings and statistics from our skeleton extraction experiments.

Shape	Number of vertices	Iterations	Skeletonization time (s)	Refinement time (s)
Fertility	9551	6	18	24
Neptune	14024	5	23	21
Pegasus	14990	9	58	87
Children	18114	7	53	56

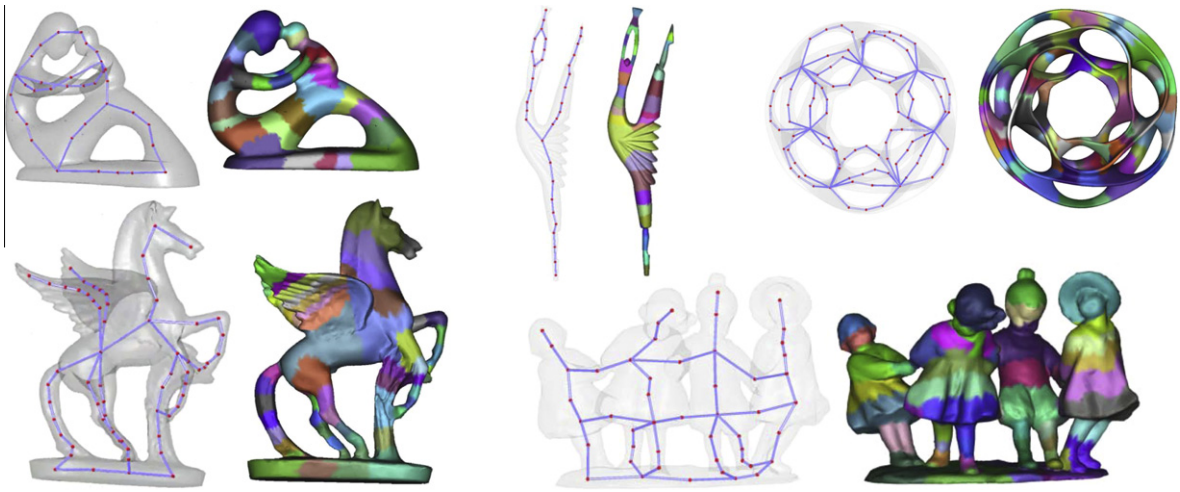


Fig. 6. A gallery of results for various input shapes, including Fertility, Dancer, Heptoroid, Pegasus, Dancing Children (top left to bottom right). Our method extracts high quality, approximately-centered, skeletons even for complex models with high-genus and flat regions.

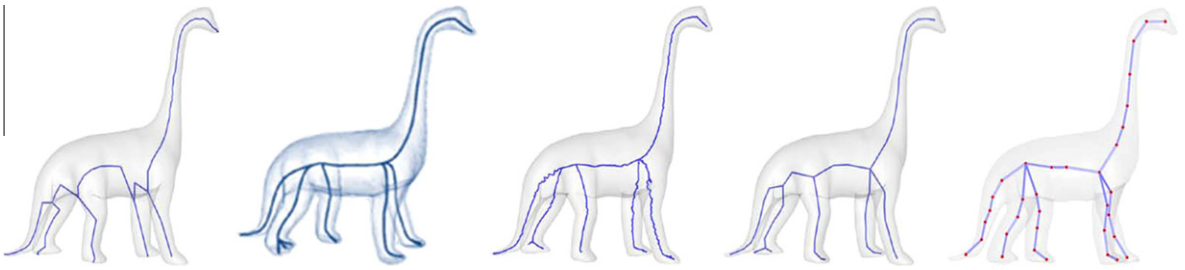


Fig. 7. Skeletons extracted by different algorithms. From left to right: potential field [26], gradient vector flow [29] (image after [29]), medial geodesic function [31] (image after [31]), Laplacian contraction [15], and our method.

Fig. 8. Skeletons extracted by different algorithms. From left to right: gradient vector flow [29] (image courtesy of [29]), shrinking and thinning [42] (image after [42]), domain connected graph [27] (image after [27]), simultaneous shape decomposition and skeletonization [37] (image after [37]), and our method.

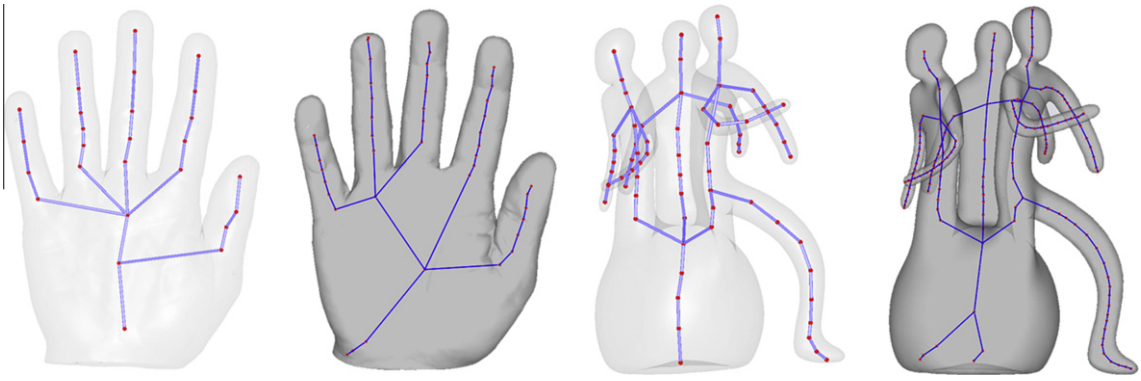


Fig. 9. Further comparisons of our method (left) with Laplacian contraction [15] (right). Differences can be seen in the palm of the Hand model, and the spherical base of the Momento model.

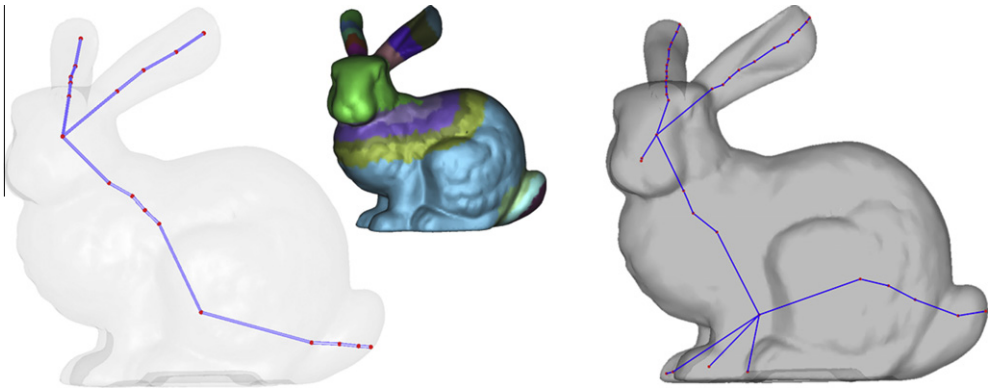


Fig. 10. Our method has missed the small protrusions of the forelegs of the Bunny (left) which are captured by the method in [15] (right).

triangle at a time. This is because we hope to keep the cluster in the shape of a ring to avoid introducing incorrect topology. Fig. 1e demonstrates the effect of geometric refinement.

4. Results and discussions

We next present results of skeleton extraction and surface clustering on a variety of input 3D shapes, as well as giving a comparison with other state-of-the-art skeleton extraction methods, and discussing the properties of the skeletonizations produced. Finally, we show how the clusters produced by our approach can also be used in skinning deformation.

4.1. Capability and speed

Fig. 6 demonstrates the functionality of our method with a gallery of 3D shapes. The results demonstrate that our method can correctly capture the genus of the input shape and produce an approximately centered skeleton within each shape.

All results shown in this paper were produced with the same parameter settings given earlier. Table 1 reports the running time of our algorithm on four models with various mesh densities, using a PC with an Intel Core2Quad 2.4 GHz CPU and 2 GB memory.

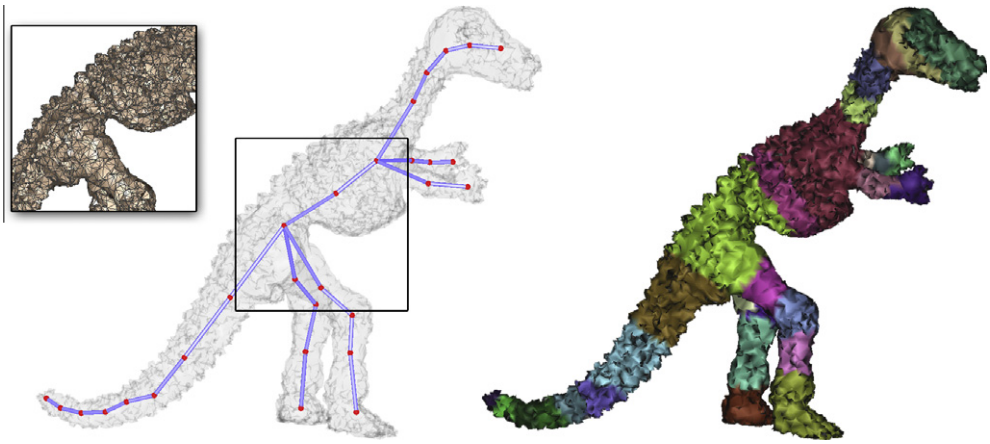


Fig. 11. Robustness to Gaussian noise. The curve skeleton and surface clusters are still reasonable.

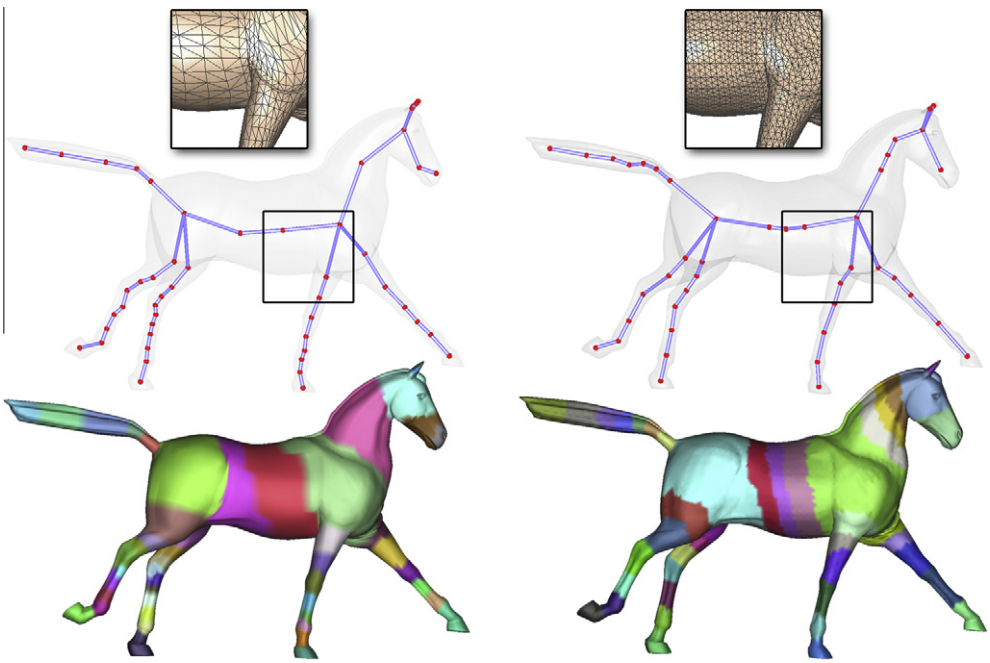


Fig. 12. Results for two differently meshed Horses. The geometry of the skeleton produced by our method is insensitive to changes in shape tessellation, even if the number of skeleton segments varies somewhat.

4.2. Comparison

We next compare our method with other state-of-the-art algorithms. Fig. 7 shows results for a dinosaur using methods based on potential field [26], gradient vector flow [29], medial geodesic function [31], and Laplacian contraction [15], as well as ours. Fig. 8 shows the results for a horse using methods based on gradient vector flow [29], shrinking and thinning [42], domain connected graph [27], simultaneous shape decomposition and skeletonization [37], and ours. The results show that our method can achieve results competitive with the state-of-the-art approaches, both volumetric [26,29,31,42], and geometric [15,27,37].

Since the mesh contraction approach of [15] is most similar to our method, we make a further comparison; we have used closed manifolds as their method imposes this requirement. See Fig. 9. Their method tends to produce unwanted skeleton branches towards non-protruding corners, e.g. the side of the base of the Hand, or the base of the Momento. Our method does not suffer from this defect, due to the good sampling properties of ACVD and the use of centers of mass of clusters as skeleton points. On the other hand, this difference is not always to our advantage, as can be seen in Fig. 10—we may miss some short skeleton segments in slightly protruding regions. More importantly, our algorithm can perform skeletonization on point clouds as explained in the following, unlike [15] which can only handle meshes.

4.3. Skeletonization properties

We now discuss whether our algorithm possesses the following features, which are agreed widely [1,2] to be

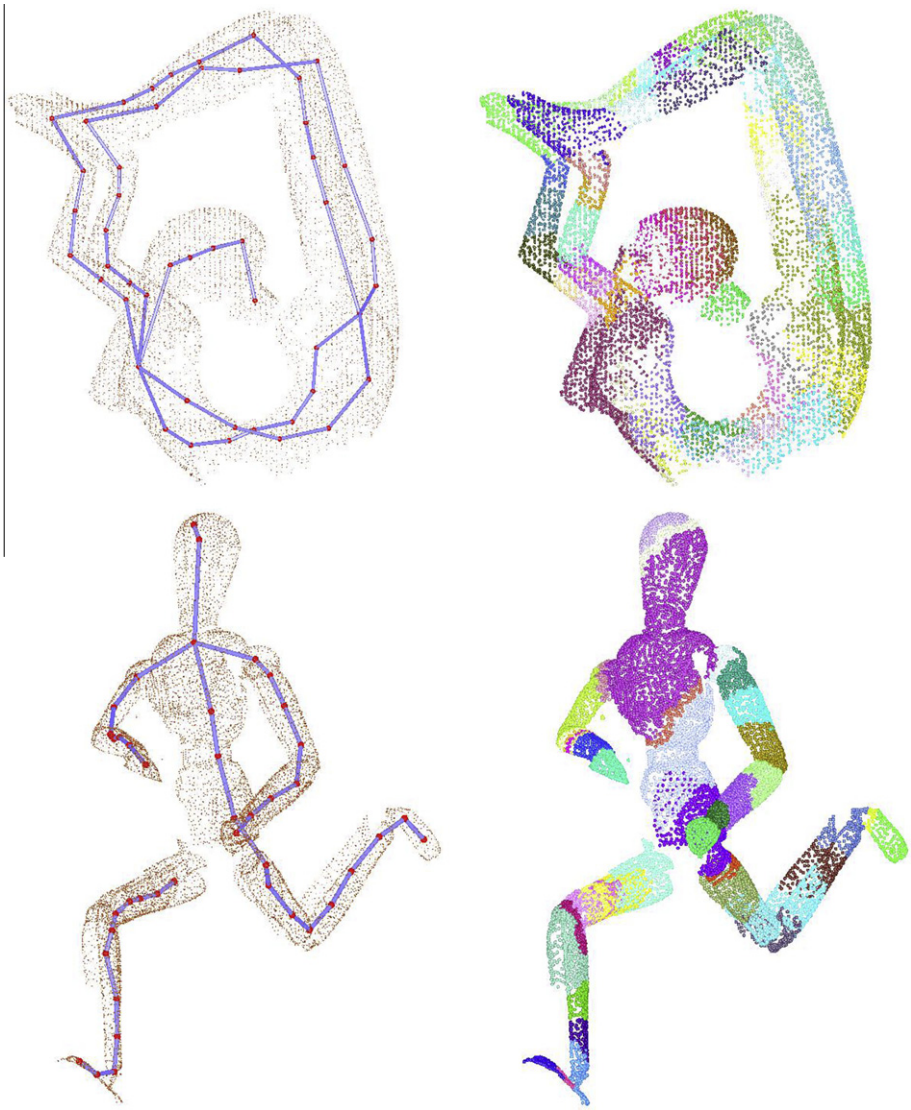


Fig. 13. Results of skeleton extraction and point cloud clustering on two real scans. Our method obtains plausible skeletons for point cloud with large missing data.

the properties that an ideal curve skeleton method should possess:

- *Centeredness:* During skeletonization iteration, centeredness is encouraged by the eccentricity control on surface clustering. Optional skeleton refinement can further improve centeredness a little.
- *Connectedness:* The curve skeleton is extracted by graph contraction, which never breaks the skeleton graph into unconnected parts. Therefore, connectedness is guaranteed.
- *Homotopy:* Topology preservation is guaranteed by essential traingle constraints during graph contraction.
- *Robustness:* Our method is insensitive to geometric noise and variations in input tessellation and its resolution. A key feature of our coupled clustering approach is that surface clustering provides regularization for skeleton graph contraction. This helps to make the graph contraction very robust even with badly distributed initial seed points. Figs. 11 and 12 demonstrate the robustness of our algorithm against geometric noise (Gaussian) and mesh tessellation changes.
- *Reliability:* Reliability of the output can be easily checked, using the visibility test for each boundary location on the original shape. Even for shapes with large gaps, e.g. the hand in Fig. 14, our algorithm produces a plausible curve skeleton (unlike [15], for example).
- *Generality:* It is straightforward to extend our ideas to deal with point cloud data. It suffices to construct an approximate surface connecting the points with their neighbors found by k-NN search, filtered to have normal consistency, and use that to initialize skeleton construction. Due to the insensitivity to the graph and its connectivity, our algorithm can compute plausible skeletons for point clouds of real scans (with missing data): see Fig. 13.
- *Uniqueness:* The final skeleton is not unique in the sense of being independent of the input geometric representation, and, for example, the final skeletons may have differing numbers of segments. Nevertheless, there final geometry will be almost the same.

4.4. Application: deformation

Perhaps the most widespread application of skeletons is skeleton-driven deformation, a key component of skinning animation. We show in Fig. 14 that skeletons extracted with our method, although only approximately-centered, can be used to produce visually pleasing deformation results. The deformation in this figure was produced using the dual quaternion technique developed in [43].

4.5. Limitations

Firstly, graph contraction is guided by heuristic criteria and we cannot provide a theoretical guarantee on certain properties of the extracted skeletons. For example, our method produces only approximately centered skeletons; nevertheless this issue can be easily addressed using the skeleton embedding technique described in [15]. Although

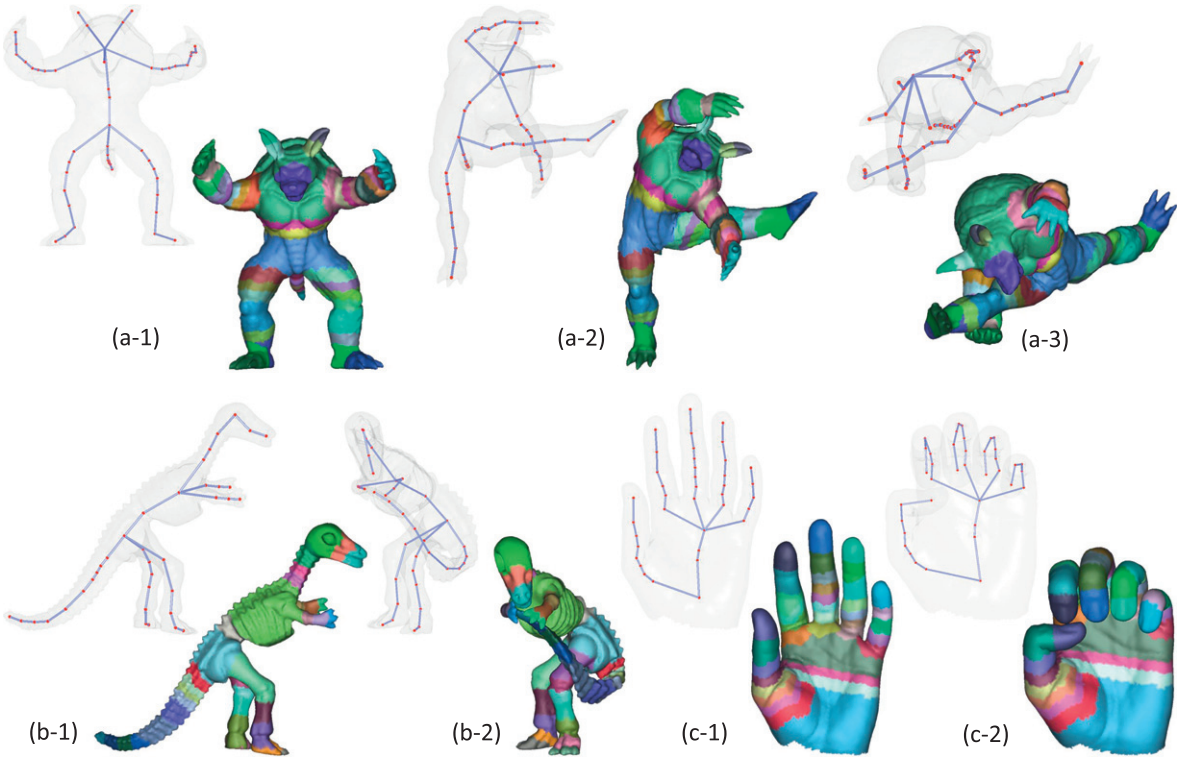


Fig. 14. Skinning deformation of the Armadilo, Dinosaur, and Hand (with hole) driven using the extracted skeletons. The input configurations for the three models are (a-1), (b-1) and (c-1), respectively. The extracted skeletons are sufficient to produce visually-pleasing skinning deformations.

we have provided a practical solution to the problem of coupled skeleton extraction and surface decomposition, more work is needed on its theoretical aspects, and the relation of our algorithm to other Voronoi-based approaches.

Another limitation of our method is that our method may overlook small geometric protrusions due to how the clustering works. Fig. 10 shows such an example on the Bunny model. Our method fails to produce skeleton points in the forelegs of the Bunny, as the surface cluster on the body incorrectly subsumes the forelegs. A possible solution here is to further constrain surface clustering using segmentation boundaries found by the minima rule [38]. Finally, our method does not extend to 2D, because surface clustering, which aims to form cylindrical clusters [8], does not have an analog for 2D contours.

5. Conclusions

We have presented a novel, practical algorithm for curve skeleton extraction for 3D shapes. Our algorithm is based on coupled processes of graph contraction on the skeleton graph and surface clustering on the input mesh triangles. Surface clustering serves to ensure uniformity during skeleton graph contraction, leading to an approximately-centered skeletonization. Our experiments on various shapes demonstrate that our algorithm possesses many desirable properties: connectedness, homotopy, reliability, and robustness. It can readily be used with point cloud data. A further advantage is that we also output surface clusters associated with the skeleton, for use in applications such as skinning deformation.

Acknowledgments

We would like to thank Sebastien Valette, Nicu D. Cornea, and Oscar Kin-Chung Au for sharing their source code. All test shapes are from http://www-roc.inria.fr/gamma/download/ or the AIM@SHAPE Shape Repository. This work was supported by NSFC (Nos. 60970094, 61103084, 61202333, and 61272334).

References

[1] N.D. Cornea, D. Silver, P. Min, Curve-skeleton properties, applications, and algorithms, IEEE Transactions on Visualization and Computer Graphics 13 (3) (2007) 530–548.
[2] S. Biasotti, D. Attali, J.-D. Boissonnat, H. Edelsbrunner, G. Elber, M. Mortara, G.S. Baja, M. Spagnuolo, M. Tanase, R. Veltkamp, Skeletal structures, in: L. Floriani, M. Spagnuolo (Eds.), Shape Analysis and Structuring, Mathematics and Visualization, Springer, Berlin Heidelberg, 2008.
[3] N. Amenta, S. Choi, Voronoi methods for 3d medial axis approximation, in: K. Siddiqi, S.M. Pizer (Eds.), Medial Representations, Computational Imaging and Vision, vol. 37, Springer, Netherlands, 2008, pp. 223–239.
[4] S. Yoshizawa, A.G. Belyaev, H.-P. Seidel, Free-form skeleton-driven mesh deformations, in: ACM Symposium on Solid Modeling and Applications, 2003.
[5] I. Baran, J. Popović, Automatic rigging and animation of 3d characters, ACM Transactions on Graphics (Proc. SIGGRAPH) 26.
[6] H.-B. Yan, S. Hu, R.R. Martin, Y.-L. Yang, Shape deformation using a skeleton to drive simplex transformations, IEEE Transactions on Visualization and Computer Graphics 14 (3) (2008) 693–706.
[7] Q.-Y. Zhou, T. Ju, S.-M. Hu, Topology repair of solid models using skeletons, IEEE Transactions on Visualization and Computer Graphics 13 (4) (2007) 675–685.
[8] A. Tagliasacchi, H. Zhang, D. Cohen-Or, Curve skeleton extraction from incomplete point cloud, ACM Transactions on Graphics 28 (2009) 71:1–71:9.
[9] O.K.-C. Au, C.-L. Tai, D. Cohen-Or, Y. Zheng, H. Fu, Electors voting for fast automatic shape correspondence, Computer Graphics Forum 29 (2010) 645–654.
[10] J.P. Lewis, M. Cordner, N. Fong, Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation, in: Proc. SIGGRAPH, 2000.
[11] A. Manzanera, T. Bernard, F. Preteux, B. Longuet, Medial faces from a concise 3d thinning algorithm, in: International Conference Computer Vision, 1999, pp. 337–343.
[12] N. Amenta, S. Choi, R.K. Kolluri, The power crust, in: ACM Symposium on Solid Modeling and Applications, 2001, pp. 249–266.
[13] S. Tari, J. Shah, Local symmetries of shapes in arbitrary dimension, in: International Conference on Computer Vision, 1998, pp. 1123–1129.
[14] X. Li, T.W. Woon, T.S. Tan, Z. Huang, Decomposing polygon meshes for interactive applications, in: Symposium on Interactive 3D Graphics and Games, 2001, pp. 35–42.
[15] O.K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, T.-Y. Lee, Skeleton extraction by mesh contraction, ACM Transactions on Graphics 27 (2008) 44:1–44:10.
[16] W. Jiang, K. Xu, Z.-Q. Cheng, R.R. Martin, G. Dang, Curve skeleton extraction by graph contraction, in: Computational Visual Media Conference, 2012.
[17] S. Valette, J.-M. Chassery, Approximated centroidal Voronoi diagrams for uniform polygonal mesh coarsening, Computer Graphics Forum 23 (3) (2004) 381–389.
[18] A. Bucksch, R. Lindenbergh, M. Menenti, SkelTre: robust skeleton extraction from imperfect point clouds, The Visual Computer 26 (10) (2010) 1283–1300.
[19] R. Ogniewicz, M. Ilg, Voronoi skeletons: theory and applications, in: Proc. Computer Vision and Pattern Recognition, 1992, pp. 63–69.
[20] N. Amenta, M. Bern, Surface reconstruction by Voronoi filtering, in: Proceedings of the Fourteenth Annual Symposium on Computational Geometry, 1998, pp. 39–48.
[21] G. Bertrand, G. Malandain, A new characterization of three-dimensional simple points, Pattern Recognition Letters 15 (1994) 169–175.
[22] L. Liu, E.W. Chambers, D. Letscher, T. Ju, A simple and robust thinning algorithm on cell complexes, Computer Graphics Forum 29 (7) (2010) 2253–2260.
[23] J.-H. Chuang, C.-H. Tsai, M.-C. Ko, Skeletonization of three-dimensional object using generalized potential field, IEEE Transaction on Pattern Analysis and Machine Intelligence 22 (2000) 1241–1251.
[24] I. Bitter, A.E. Kaufman, M. Sato, Penalized-distance volumetric skeleton algorithm, IEEE Transactions on Visualization and Computer Graphics 7 (2001) 195–206.
[25] W.-C. Ma, F.-C. Wu, M. Ouhyoung, Skeleton extraction of 3d objects with radial basis functions, in: Proc. Shape Modeling International, 2003, pp. 207–214.
[26] N.D. Cornea, D. Silver, X. Yuan, R. Balasubramanian, Computing hierarchical curve-skeletons of 3d objects, The Visual Computer 21 (11) (2005) 945–955.
[27] F.-C. Wu, W.-C. Ma, R.-H. Liang, B.-Y. Chen, M. Ouhyoung, Domain connected graph: the skeleton of a closed 3d shape for animation, The Visual Computer 22 (2) (2006) 117–135.
[28] L. Shapira, A. Shamir, D. Cohen-Or, Consistent mesh partitioning and skeletonisation using the shape diameter function, The Visual Computer 24 (4) (2008) 249–259.
[29] M.S. Hassouna, A.A. Farag, Variational curve skeletons using gradient vector flow, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 2257–2274.
[30] S. Katz, A. Tal, Hierarchical mesh decomposition using fuzzy clustering and cuts, ACM Transactions on Graphics (Proc. SIGGRAPH) 22 (2003) 954–961.
[31] T.K. Dey, J. Sun, Defining and computing curve-skeletons with medial geodesic function, in: Eurographics Symposium on Geometry Processing, 2006, pp. 143–152.
[32] A. Sharf, T. Lewiner, A. Shamir, L. Kobbelt, On-the-fly curve-skeleton computation for 3d shapes, Computer Graphics Forum 26 (3) (2007) 323–328.

[33] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, Z. Su, Point cloud skeletons via Laplacian-based contraction, in: Proc. Shape Modeling International, 2010, pp. 187–197.
[34] V. Pascucci, G. Scorzelli, P.-T. Bremer, A. Mascarenhas, Robust on-line computation of Reeb graphs: simplicity and speed, ACM Transactions on Graphics 26.
[35] M. Hilaga, Y. Shinagawa, T. Kohmura, T.L. Kunii, Topology matching for fully automatic similarity estimation of 3d shapes, in: Proc. SIGGRAPH, 2001, pp. 203–212.
[36] G. Aujay, F. Hétroy, F. Lazarus, C. Depraz, Harmonic skeleton for realistic character animation, in: ACM SIGGRAPH/Eurographics symposium on Computer animation, 2007, pp. 151–160.
[37] J.-M. Lien, J. Keyser, N.M. Amato, Simultaneous shape decomposition and skeletonization, in: ACM symposium on Solid and physical modeling, 2006, pp. 219–228.
[38] A. Shamir, A survey on mesh segmentation techniques, Computer Graphics Forum 27 (6) (2008) 1539–1556.
[39] S. Schaefer, C. Yuksel, Example-based skeleton extraction, in: Eurographics symposium on Geometry processing, 2007, pp. 153–162.
[40] Y. He, X. Xiao, H.S. Seah, Example based skeletonization using harmonic one-forms, in: Proc. Shape Modeling International, 2008, pp. 53–61.
[41] N. Hasler, T. Thormählen, B. Rosenhahn, H.-P. Seidel, Learning skeletons for shape and pose, in: Symposium on Interactive 3D Graphics and Games, 2010, pp. 23–30.
[42] Y.-S. Wang, T.-Y. Lee, Curve-skeleton extraction using iterative least squares optimization, IEEE Transactions on Visualization and Computer Graphics 14 (4) (2008) 926–936.
[43] L. Kavan, S. Collins, J. Žára, C. O'Sullivan, Skinning with dual quaternions, in: Proc. of Symposium on Interactive 3D Graphics and Games, 2007, pp. 39–46.
[44] A. Tagliasacchi, I. Alhashim, M. Olson, H. Zhang, Mean Curvature Skeletons, Computer Graphics Forum 31 (5) (2012) 1735–1744.