DOI: 10.1111/cgf.13906

COMPUTER GRAPHICS *forum*
Volume 00 (2020), number 00 pp. 1–22

Curve Skeleton Extraction From 3D Point Clouds Through Hybrid Feature Point Shifting and Clustering

Hailong Hu,[1,2] Zhong Li,[1,3] Xiaogang Jin,[4] Zhigang Deng,[5] Minhong Chen[3] and Yi Shen[3]

[1]Faculty of Mechanical Engineering & Automation, Zhejiang Sci-Tech University, Zhejiang, Hangzhou, China
huhailonga@163.com, lizhong@zstu.edu.cn
[2]School of Science, Zhejiang A&F University, Zhejiang, Hangzhou, China
[3]School of Science, Zhejiang Sci-Tech University, Zhejiang, Hangzhou, China
zjyh-cmh@163.com, yshen@zstu.edu.cn
[4]State Key Lab of CAD&CG, Zhejiang University, Zhejiang, Hangzhou, China
jin@cad.zju.edu.cn
[5]Department of Computer Science, University of Houston, Houston, TX, USA
zdeng4@uh.edu

Abstract

Curve skeleton is an important shape descriptor with many potential applications in computer graphics, visualization and machine intelligence. We present a curve skeleton expression based on the set of the cross-section centroids from a point cloud model and propose a corresponding extraction approach. We first provide the substitution of a distance field for a 3D point cloud model, and then combine it with curvatures to capture hybrid feature points. By introducing relevant facets and points, we shift these hybrid feature points along the skeleton-guided normal directions to approach local centroids, simplify them through a tensor-based spectral clustering and finally connect them to form a primary connected curve skeleton. Furthermore, we refine the primary skeleton through pruning, trimming and smoothing. We compared our results with several state-of-the-art algorithms including the rotational symmetry axis (ROSA) and L_1-medial methods for incomplete point cloud data to evaluate the effectiveness and accuracy of our method.

Keywords: curve skeleton, point cloud, hybrid feature point, spectral clustering

ACM CCS: I.3.5 Computing methodologies: Computer graphics

1. Introduction

Skeleton of an object is an intuitive and effective abstraction of its topological and geometric properties [CSM07, SP08]. It has been widely used in a wide range of geometric tasks, including shape analysis, shape decomposition, model retrieval, computer animation, virtual reality and 3D printing [TDS*16, BAS14, LD14].

Many previous efforts have been conducted to extract curve skeletons. A curve skeleton, also called the line-like representation of a 3D object, is a simplified 1D representation of the original shape, consisting of curves only [JST16]. In spite of its simplicity, there does not exist an unanimously accepted formal definition of the curve skeleton [Cor07]. Recently, Dey and Sun [DS06] proposed a mathematical definition of curve skeleton based on the medial geodesic function. However, it is quite difficult to compute the curve-skeleton exactly as they defined, which is based on the medial axis and medial geodesic function approximation. An ideal curve skeleton is expected to have the following properties: Homotopic, invariant under isometric transformations, thinness, centredness, smoothness, componentwise differentiation, reliability and robustness [CSM07]. Many existing curve skeleton extraction algorithms are designed for watertight surface meshes, such as topological thinning [Pal08], distance field transformations [ZT99, HF09, JST16, SJT14], centroidal Voronoi tessellation [LLW12], Reeb graph construction [GSBW11], surface contraction via mean curvature flow [SYJT13, CK11, ATC*08], etc. In some cases, a watertight mesh is not given as input, but a partial point cloud coming from a 3D scanner. In these cases, the previous algorithms could not be used. Therefore, researchers have proposed several skeleton extraction

Zhong Li is the corresponding author.

algorithms from raw scan data, including the generalized rotational symmetry axis (ROSA)-based method [TZCO09], the L_1-medial skeleton extraction algorithm based on random sampling [HWCO*13] and the combination of the distance field and L_1-median skeleton extraction from raw scanned point cloud data [SPJX18]. How to obtain a centred skeleton with topology preservation from incomplete or noisy point cloud data remains a challenge.

In this paper, we propose a curve skeleton expression based on the centroid set of the cross-sections of a 3D object, and then further introduce a corresponding skeleton extraction from a point cloud model. Different from existing methods, our method has the following contributions.

1. We search for the feature points of a point cloud model according to the local extremes of both geometric features (i.e. mean curvatures) and the distance field substitution. Here, the purpose of using the distance field substitution is to acquire more feature points to construct skeletal points for robust skeleton completion. It does not require voxelization [ZT99] and can be easily implemented based on the distances from the points inside the shape to the three faces of the axially aligned bounding box (AABB).
2. We optimize the skeleton-guided normal direction for each feature point, and introduce the relevant facet to search for the relevant points of each feature point. The hybrid feature point shifting, introduced in our work, helps to accurately construct skeletal points.
3. We provide a novel tensor-based spectral clustering method to classify skeleton points, which ensures the resulting curve skeleton to retain more details with componentwise differentiation.

Compared to Tagliasacchi *et al.*'s method [TZCO09], our skeleton extraction from a point cloud model does not require *any prior assumptions* about its geometry (e.g. cylindrical) or topology. And, compared to the random sampling algorithm by Huang *et al.* [HWCO*13], our hybrid feature points are often sufficient to accurately describe the shape of a point cloud model, even with significantly missing data.

The remainder of this paper is organized as follows. Section 2 describes recent related works. Section 3 presents our curve skeleton definition and the overview of its extraction method. Sections 4–6 provide the details of our skeleton extraction algorithm, including hybrid feature points and their shifting in Section 4, skeleton point computation through tensor-based spectral clustering and region-growing clustering algorithm in Section 5 and skeleton points connected to form the primary skeleton and skeleton refinements in Section 6. Section 7 provides our experimental results, comparisons and limitation discussion. Finally, Section 8 concludes the paper with future work.

2. Related Work

Blum [B*67] defined the skeleton of a model based on the medial axis transform (MAT) in 1967. From then on, numerous methods have been developed to extract skeletons from 3D models. Existing skeleton extraction methods can be roughly categorized as: volumetric methods, geometric methods for mesh models and geometric methods for point cloud models.

Volumetric methods for mesh models extract skeletons by making use of a regularly partitioned voxelized discrete representation. They commonly include topological thinning and distance field operations. For example, Palágyi [Pal08] proposed a parallel thinning algorithm by applying a computed search-table. Zhou and Toga [ZT99] used a voxel-code algorithm for distance field transformation and then connected and smoothed the extracted skeleton. Hassouna and Farag [HF09] applied gradient vector flows via two different energy functions to extract the skeleton that is insensitive to noise. Jalba *et al.* [JST16] contracted the boundary into the surface skeleton or curve skeleton counterpart for voxel models. Sobiecki *et al.* [SJT14] performed a comprehensive comparison on various curve and surface skeleton extraction methods for voxel models. Generally, these methods can produce topology-preserving skeletons but may contain redundant branches. Furthermore, it is often difficult for them to produce quality skeletons for models with significant missing data.

Geometric methods for mesh models are another category of widely used skeleton extraction methods that only require the surface information of 3D models. For example, Näf *et al.* [NSK*97] extended the Voronoi graph and apply it to extract skeletons for 3D models. Lu *et al.* [LLW12] used Thiessen-polygons to extract the skeletons for simple mesh models. Ge *et al.* [GSBW11] modified the Reeb graph method based on Morse functions to generate skeletons. Sobiecki *et al.* [SYJT13] gave a review of boundary contraction methods. Chuang and Kazhdan [CK11] used fast mean curvature flows via finite-element tracking to extract skeletons. There exist other methods to extract skeletons such as generalized-potential field-based method [CTK00] and visible repulsive force-based method [WML*03]. These methods can keep the skeleton with centredness, but their computational complexity is often high, and the resulting skeletons may not be sufficiently smooth. Recently, Au *et al.* [ATC*08] employed Laplacian smooth functions to contract an input mesh model to obtain its curve skeleton, which is robust and insensitive to noise. However, all the above methods are generally designed to extract skeletons from 3D mesh models, and they cannot be directly applied to unoriented point cloud models, because most of them utilize the edges and normals of the input mesh model.

Besides extracting skeletons from a single mesh model, some other efforts have been focused on extracting skeletons (especially animation skeletons) from mesh examples (or animated meshes). For example, Kirk *et al.* [KOF05] extracted skeletons from marker-based mocap data, utilizing the temporal coherence between mocap frames. However, limited to marker motion capture data, this method can only obtain the joint locations between two rigid body parts and fail to handle the blending between bodies. After that, a number of approaches have been developed to extract animation skeletons [SY07, DATTS08, HTRS10, LD14] or bone transformations [JT05, KSO10, LD12] from a set of example poses.

Geometric methods for point cloud models. To extract the skeletons from point cloud models, Cao *et al.* [CTO*10] extended the Laplacian contraction–based skeleton extraction method from mesh models to point cloud models. However, in order to construct the adjacent relationship through the k-nearest neighbours (*KNN*) and Laplacian matrix, the input point cloud model is required to be sufficiently dense and noise-free. Both Bucksch *et al.* [BLM10]

and Natali *et al.* [NBPF11] extended the Reeb graph reconstruction to extract skeletons from point cloud models. Despite their certain successes, the above methods still fall short of handling point cloud models with noise or incomplete data.

Sharf *et al.* [SLSK07] proposed a skeleton extraction method based on the evolution of a deformable model for both meshes and point set objects. Livesu *et al.* [LGS12] proposed a curve skeleton reconstruction method of 3D shapes based on the visual hull concept. It can work equally well on polygonal meshes, voxel models and point clouds, and is robust against noises and missing data. Moreover, they first proposed the approach including the perceptual core extraction, branch collapsing and loops recovery, which has become the common topological operation for the skeleton extraction [TDS*16]. Li *et al.* [LLZM10] extracted the skeleton for slender tubular shapes. Livny *et al.* [LYO*10] proposed an automatic approach to extract skeletons for tree structures, in particular, it can handle trees with multiple overlapping, without segmentation. Bremer *et al.* [BRW13] also presented a fully automated modular workflow to extract skeletons for tree structures. Jalba *et al.* [JKT13] extracted and regularized skeletons from point cloud models using a new GPU-based geodesic tracing technique. The resulting skeletons by the above methods are topology-preserving, but their qualities are highly dependent on the input point cloud data.

Tagliasacchi *et al.* [TZCO09] proposed a ROSA method to effectively extract skeletons from incomplete point cloud models. The main problem is that it assumes the shapes are cylindrical and need accurate normal vectors of points, pre-filter noise and outliers. Huang *et al.* [HWCO*13] presented an L_1-medial skeleton algorithm to extract and smooth the skeletons of point cloud models. It can generate a feasible skeleton from incomplete missing data. However, the resulting skeleton by this method may be poor when the data are significantly missing or the initial set of sample models are insufficient to express the shape complexity. Song *et al.* [SPJX18] proposed a distance field guided L_1-median skeleton extraction method from raw scanned point cloud data. In their method, voxelization is the key operation that may influence the quality of the resulting skeleton. Recently, Lu *et al.* [LCY*18] proposed an unsupervised articulated skeleton extraction from point set sequences captured by a single depth camera, which can guarantee the visual quality and accuracy. Compared to the methods in [LCY*18] and [SPJX18], our method does not need the process of non-rigid point set registration and voxelization, which not only improves the performance but also reduces the memory footprint.

3. The Preliminary of Our Method

Tagliasacchi *et al.* [TDS*16] surveyed the existing definitions and properties of various types of 3D skeletons. There does not exist a universal definition on curve skeletons. Some widely accepted definitions of a curve skeleton are based on medial axis, centres of maximally inscribed balls, L_1-medial, etc. In this paper, we first capture the cross-sectional planes that are perpendicular to the skeleton along with pairs of antipodal points, and then use the set of cross-section centroids to construct the curve skeleton of a 3D object. The antipodal points include the feature point and its relevant point, which are defined in Section 4. Note that although there are infinite planes passing through the two points, we directly use the two antipodal points to calculate the centroid of the cross-section plane in order to avoid the construction of the cross-section plane. This process can be implemented because we can use curvature and distance field substitution to obtain hybrid feature points, and use the normal optimization to locate the relevant point of each hybrid feature point. Then the cross-section centroid is determined by the midpoint of the feature point and its relevant point.

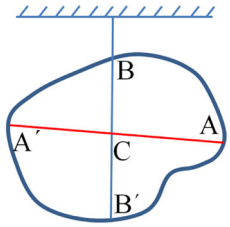


Figure 1: *Centroid approximation according to the two-force equilibrium axiom.*

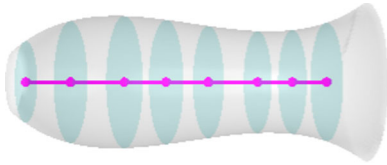


Figure 2: *A curve skeleton is composed of the cross-section centroids.*

Given a thin and flat plate that occupies a 2D planar region D in the *xy*-plane, we can approach the centroid of an object according to the two-force equilibrium axiom [CFD*92]. Namely, if it is hung twice from a string, then its centroid will be located at the intersection of the two extended lines, as illustrated in Figure 1.

Analogously, given a model $\Omega \in R^3$ with boundary $\partial\Omega$, we define its curve skeleton S_Ω as follows:

$$S_\Omega = \{x \in CSP_i | x = \text{centroid}(CSP_i), i = 1, 2, \ldots\},$$

where CSP_i is a cross-sectional plane, and centroid(CSP_i) denotes the centroid of a 2D planar region in CSP_i. For example, in Figure 2, the magenta points are the centroids of the corresponding cross-sections and the curve connecting these centroids is regarded as the curve skeleton of the model.

For a 3D point cloud model, it is difficult to find the cross-sections perpendicular to an unknown axis (i.e. the curve skeleton), so directly applying the above skeleton definition to extract skeletons is technically infeasible. Instead, we propose an approximation method, where a cross-section is constructed based on feature points and the extracted relevant points, and its centroid is determined through the feature point shifting.

The main steps of our method are described as follows (illustrated in Figure 3): (a)–(c) obtain hybrid feature points from a point cloud model based on mean curvatures and distance field substitution; (d) compute the relevant point of each feature point and set the shift-step to obtain the centroids of cross-sections; (e) obtain skeleton points

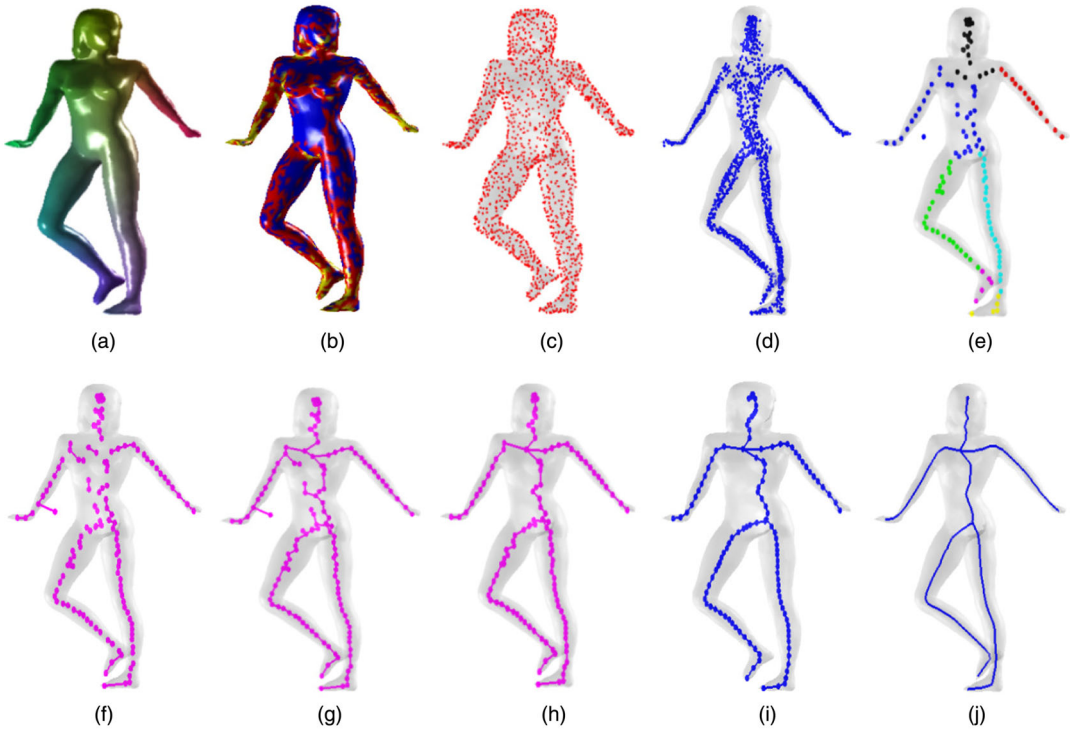


Figure 3: *The pipeline of our skeleton extraction method. (a) Compute the distance field substitution, where the colour (r, g, b) is represented by the distance field substitution (D_1, D_2, D_3), D_i is the normalized distance from this point to the ith benchmark plane of AABB, (b) compute the mean curvatures map, visualized as a heat map, curvatures from small to large are represented by blue, red and yellow, respectively, (c) compute feature points, (d) compute the approximation of centroid points, (e) extract the skeleton points, (f) construct the initial primary adjacency list, (g) obtain the primary connected skeleton, (h) prune and trim the skeleton, (i) smooth the skeleton and (j) obtain the final skeleton.*

by tensor-based spectral clustering and region growing clustering; (f)–(g) construct the initial primary adjacency list and iteratively add edges to form a primary connected skeleton; (h)–(j) refine the primary skeleton by pruning, trimming and smoothing, and obtain the final curve skeleton.

Note that the ROSA method [TZCO09] needs to determine skeleton points by computing local optimal cutting planes and the relevant neighbourhood N_i according to a Mahalanobis distance–based graph. In our method, the computation of skeleton points depends on the computed relevant points that are obtained based on the positions of feature points and their normal directions. Therefore, our method does not need to compute the optimal cutting planes and relevant neighbourhood.

In our method, cross-sectional planes are auxiliary and our method does not need to explicitly compute the cross-sectional planes. Actually, after we determine the relevant point of a given feature point, the skeletal point can then be obtained according to the shifting of the feature point.

4. Initial Skeletal Points Construction

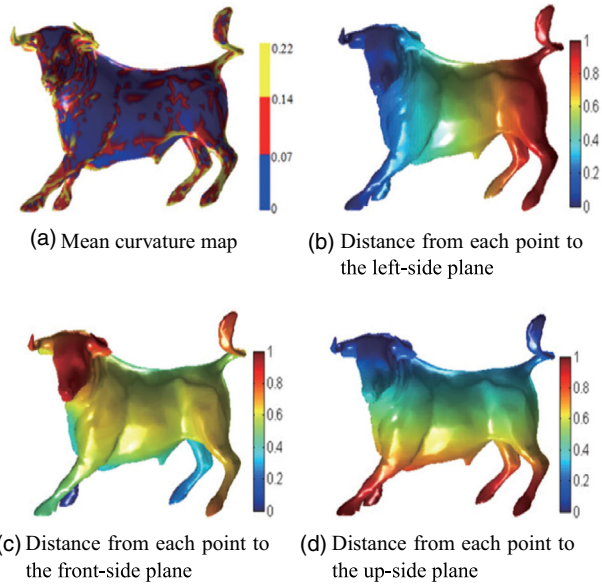
To obtain initial skeletal points, we perform a series of operations: hybrid feature points construction, normal vectors estimation and optimization and the computation of relevant facets and relevant points, as detailed below.

4.1. Hybrid feature points construction

The feature points of a point cloud model are traditionally captured by their geometric feature variances. As mean curvature (*MC*) can capture the local geometry property on the surface, we compute the mean curvatures [DHKL01] of all the points by computing the convolved covariance matrices of the Voronoi cells of the point cloud, which has been provably robust in the presence of noise [MOG11]. The mean curvatures can be visualized by different colours shown in Figure 4(a).

Another method for capturing feature points will make use of the distance field. The straightforward way of computing the distance field of a 3D model is to voxelize it and compute the minimal distance of each interior voxel to its boundary. The distance is normally computed by the Euclidean metric, chess metric, block metric, etc. The minimal distances of all the voxels together constitute the distance field. However, it is computationally expensive to compute a high-resolution distance field, which limits its application for point cloud models [SPJX18].

Different from previous methods, we introduce a novel *distance field substitution* method which is computed from each point inside



(a) Mean curvature map (b) Distance from each point to the left-side plane

(c) Distance from each point to the front-side plane (d) Distance from each point to the up-side plane

Figure 4: *Colour-coded representations of (a) the mean curvatures and (b)–(d) distance field substitution of a 3D model. Curvature from small to large is represented by blue, red and yellow, respectively.*

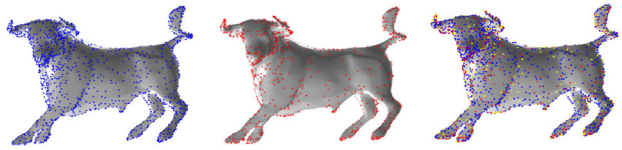
the shape to the three orthogonal facets of the minimum AABB of the model [BHP01]. As this distance does not relate to the voxelization of a point cloud and can be computed according to AABB, it is easily implemented for point cloud models. For a given point, we use this distance field substitution to capture more feature points, besides geometric feature points captured by the *MC*. Our distance field substitution first constructs the AABB of the point cloud model, and then chooses three facets that are orthogonal to each other from the six facets of the AABB, as the benchmark planes (denoted as π_i, $i = 1, 2, 3$).

Specifically, for each point p of an input point cloud model, we compute and normalize its distances to the benchmark planes, respectively, as follows:

$$D_i = \frac{distance(p, \pi_i)}{\max\limits_{p \in Point cloud}(distance(p, \pi_i))}, i = 1, 2, 3. \quad (1)$$

To this end, each point can be represented by a three-tuple (D_1, D_2, D_3), and we call these three-tuples as the distance field substitution for the point cloud model. If (D_1, D_2, D_3) is visualized through the colour values of (r, g, b), respectively, we can generate an intuitive heat-map representation of the distance field, as illustrated in Figures 4(b)–(d).

After computing the mean curvatures *MC* and the distance field substitution *D*, we can obtain the local extremes of *MC* and *D*. These corresponding points fP_s are called the *hybrid feature points* that reflect the topological and geometric features of the model [ZH04]. In our work, considering the point cloud model is discrete, we compute the local extremes of *D* and *MC* via the *KNN* algorithm. That is, for a point *P* in a point cloud, if one of the three minimal distances to the three benchmark planes is the minimum or maximum, or if the mean curvature of *P* is also the minimum or maximum among the *k* neighbours of *P*, then this point *P* is regarded as a feature point. We visualize hybrid feature point examples with separate subfigures showing the local min–max curvature points and distance points in different colours in Figure 5.



(a) Local min–max curvature points (Blue colour) (b) Local min–max distance points (Red colour) (c) Hybrid feature points (The same points with local min–max curvature points and distance points are colored in yellow)

Figure 5: *Hybrid feature point examples showing the min–max curvature points and min–max local distance points in different colours.*

4.2. Normal vectors estimation and optimization

Our method also needs to estimate the normal vectors of hybrid feature points. Specifically, we first determine the neighbours of each feature point using the *KNN* algorithm, estimate its normal through principal component analysis (PCA) [HLDZ17] and finally adjust all the normal directions of a point cloud model by employing the minimum-spanning tree algorithm [PR02]. For each point *P* and its neighbouring points in a point cloud model, we project these points into a public tangent plane at *P*, and construct a graph using Delaunay triangulation, where the edges of each triangle form the graph edges, and their weights are set as the lengths of the edges.

After the normal vector estimation for each feature point, we first search for its *relevant point* according to the normal vector, and then find the centroid of each cross-section containing the feature point and its relevant point. For each feature point p_i of a dense point cloud model, we identify a set of points from the entire point cloud whose normals are opposite to the normal of the feature point. Here, the opposite normal is judged by a dot product with a threshold. After that, among all the points in the identified point set, we further find a particular point, rP_i, such that the distance from rP_i to the feature point is minimal. In this work, we call rP_i as the relevant point of p_i. Note that in some cases spurious relevant points may exist for a given feature point, we just need to remove wrong relevant points and determine one correct relevant point (refer to Section 4.3).

Considering the normal direction of a feature point has a great impact on the position of the relevant point, and the normals of the surrounding feature points also influences the normal estimation of a given feature point, we introduce an optimization method for the skeleton-guided normal vector estimation of each feature point.

For the computed normal vector $\{N(p_i)|i = 1, 2, \ldots, n\}$, we construct the following quadratic energy function to optimize and obtain new normal vectors $\{N'(p_i)|i = 1, 2, \ldots, n\}$ for skeleton

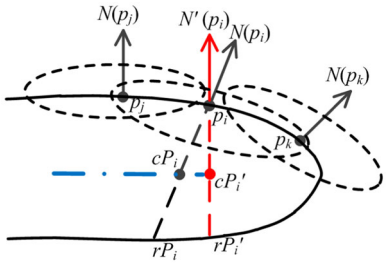
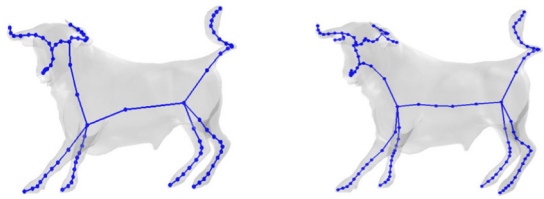


Figure 6: *Skeleton optimization based on normal direction optimization.*



(a) Skeleton without the normal optimization

(b) Skeleton with the normal optimization

Figure 7: *The skeleton comparison with and without the normal optimization for the ox model.*

extraction:

$$E = \sum_{i=1}^{n} (\langle N'(p_i), N(p_i)\rangle^2 + \lambda L(N'(p_i))^2), \tag{2}$$

where $\langle N'(p_i), N(p_i)\rangle$ is the dot product between two normal vectors $N'(p_i)$ and $N(p_i)$, which prefers that the new normal vector is close to the original normal vector; λ is a weight coefficient which is used to control the degree of the normal adjustment of p_i, and we normally set λ in the range of [0, 2]; $L(N'(p_i))$ is added to adjust the normal of p_i according to its local normal variance. We can describe it as a generalized Laplacian form for the point cloud model, namely,

$$L(N'(p_i)) = \sum_{j \in I_i} s_j \frac{1}{\|p_j - p_i\|} (N'(p_i) - N(p_j)), \tag{3}$$

where I_i is the KNN neighbourhood of p_i, $\|p_j - p_i\|$ is the distance between points p_j and p_i, $s_j = a_j / (\sum_{j \in I_i} a_j)$ is the normalized weight of each point in the neighbourhood, $a_j = \frac{1}{\sum_{k \in I_j} \angle(N(p_k), N(p_j))}$ ensures that for a given point, when the normal variance in its neighbourhood is small, s_j is set as a large value otherwise a small value. The above optimization process actually uses this weight setting to let the normal of a feature point close to the normal of its local region as much as possible, which helps to more accurately capture skeletal points.

To minimize the above quadratic energy function E (Equation 2), we compute the partial derivatives of the E with respect to the unknown variables $N'(p_i), i = 1, 2, \ldots, n$, and make it equal to zero, respectively. We thus obtain a system of linear equations with respect to $N'(p_i), i = 1, 2, \ldots, n$, which can be solved to obtain the updated normal vectors $\{N'(p_i) | i = 1, 2, \ldots, n\}$ for each feature point.

We give an example to show the result using different normal directions for skeleton extraction in Figure 6. For a point p_i of a point cloud model, point rP_i is its relevant point according to the original normal estimation on point p_i. For the point p_j, the normal variance in its neighbourhood is smaller compared to that of point p_k, so the optimized normal $N'(p_i)$ will be closer to $N(p_j)$ than $N(p_k)$. Through the above optimization process, rP_i' is the relevant point by the improved normal estimation on point p_i. We found the centroid position (cP_i') between the feature point and its relevant point is more reasonable than the original cP_i for skeleton extraction.

For some regions of a model with rich local geometric details, our extracted skeleton can be improved by the above normal optimization. We show a comparison of the skeleton results with and without the normal vector optimization in Figure 7. For the head and ear parts of the ox model, the skeleton with the normal optimization is more accurate than the result without the normal optimization. In addition, the extracted skeleton with the normal optimization can well keep the centredness property of the skeleton. For the body part of the ox model, the skeleton with the normal optimization is closer to the centre than the result without the normal optimization.

4.3. Computing relevant facets and relevant points

We determine the relevant point of a hybrid feature point through the introduction of *relevant facets*. For a facet S_j of an AABB, we denote its normal vector as $\overrightarrow{N(S_j)}$, and for a feature point fP_i inside the bounding box, its normal vector is denoted as $\overrightarrow{N(fP_i)}$. If the angle between the two normal vectors $\overrightarrow{N(S_j)}$ and $\overrightarrow{N(fP_i)}$ is smaller than $\pi/2$, we define the plane S_j is a relevant facet with respect to the point fP_i, which is denoted as rF.

Proposition 1. *Each feature point always has one, two or three relevant facet(s). (Please refer to Appendix A for its detailed explanation.)*

Proposition 2. *When searching for the relevant point of a feature point, the method by judging the opposite normal direction may create redundant spurious relevant points. But our method by judging relevant facets can remove more spurious relevant points, and then determine the accurate relevant point through the shortest distance. (Please refer to Appendix B for its detailed explanation.)*

Based on the concept of the above relevant facets, we can obtain the relevant point of a feature point efficiently. For a given feature point fP_i, assuming its normal vector is $\overrightarrow{N(fP_i)}$ and its obtained relevant facet is rF_i, we define a point rP_i in the model as the *relevant point* of fP_i, if it satisfies all of the following conditions: (1) Its normal vector is opposite to $\overrightarrow{N(fP_i)}$; (2) the distance from this point to the relevant facets rF_i is larger than the distance from fP_i to rF_i; (3) it is the closest point to fP_i among all the points satisfying both (1) and (2).

According to the above condition (1), we first search for all the points (denoted as a set R) whose normal vectors are opposite to $\overrightarrow{N(fP_i)}$. However, certain exceptions still could occur: In the example illustrated in Figure 8(c), the black point is a feature point fP_i, but the green point whose normal is opposite to $\overrightarrow{N(fP_i)}$ is not our desired result. In order to eliminate such a case, we add the condition (2) based on the distance from the point to its relevant facet, instead of the distance from fP_i to the feature point. In this way, we can obtain a subset $sR \in R$ and finally select the closest point to fP_i from sR according to the condition (3). Therefore, we can determine the correct relevant point rP_i (the red point in Figure 8c) for the feature point fP_i. To enforce the above conditions, we solve a constrained optimization problem and obtain the relevant point as follows:

$$rP_i = \arg\min_P \|P - fP_i\|,$$

s.t.

$$\begin{cases} \cos\angle(\overrightarrow{N(P)}, \overrightarrow{N(fP_i)}) = -1 \\ distance(P, rF_j) > distance(fP_i, rF_j),\ j = 1, 2, 3, \\ P \in Pointcloud, \end{cases}$$

where the point rP_i is the relevant point for the feature point fP_i.

In our implementation, considering the complexity and non-regularity of a point cloud model, it is difficult to find the point P which exactly satisfies $\cos\angle(\overrightarrow{N(P)}, \overrightarrow{N(fP_i)}) = -1$ for each feature point fP_i, so we replace it by $\cos\angle(\overrightarrow{N(P)}, \overrightarrow{N(fp_i)}) < \theta$, where θ is a small threshold.

In addition, at narrow parts with a sharp corner such as the toe tip (e.g. the blue point in Figure 8c), our method may fail to find the relevant points for some feature points. In this case, if the distance between the feature point and its relevant point is larger than a given threshold, we do not compute the skeletal point for this feature point and its relevant point. The threshold is set as the average distance from all feature points to their relevant points in a local *KNN* neighbourhood.

Finally, after determining the relevant point rP_i for each feature point fP_i, we compute the shifting step by $step_i = 1/2\|fP_i - rP_i\|$, and then make fP_i to shift the step along the opposite direction of its skeleton-guided normal vector to determine the approximated centroid cP_i.

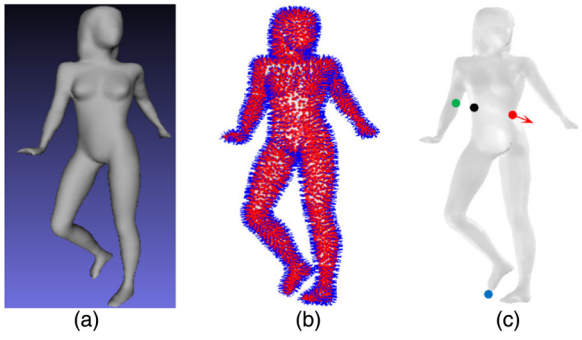


Figure 8: *Computing the relevant points and normal vectors (b) of a 3D shape (a) and (c). The black point is a feature point, the green point is a spurious relevant point, the red point is our desired point (relevant point) and the arrow line is its normal vector.*

5. Skeletal Points Construction

Often, the obtained centroid points (from the above step) are over-dense to extract the skeleton of the 3D object, as shown in Figure 3(d). Therefore, we first cluster these points to groups using a tensor-based spectral clustering. Then, we employ a region growing clustering algorithm to iteratively reduce the centroid points in each group. The retained points are used to determine the skeletal points of the curve skeleton.

5.1. Tensor-based spectral clustering for centroid points

Over the years many clustering algorithms have been developed, including K-means, hierarchical clustering, DBSCAN method, etc. Compared to these methods, spectral clustering [BN03] has some nice properties (e.g. clustering in fewer dimensions is more computationally efficient; the affinity matrix is constructed for preserving the local features and topology of the model) and has been successfully used for a variety of applications. But in the conventional spectral clustering method, the distance matrix for clustering is normally constructed according to the distance property. Recently, Wu *et al.* [WBG16] proposed a general tensor spectral co-clustering method for high-order data and it can be used for partitioning higher-order networks or complex structures [BGL15]. Ma *et al.* [MWF*10] proposed an effective point cloud segmentation through spectral clustering using both distance and normal direction. When clustering the centroid points in this work, we found we can obtain more desired clustering results if additional properties, such as the variance of normals and the curvatures of the centroid points, are considered.

For the above reason, we propose a tensor-based spectral clustering method by extending the existing matrix-based spectral cluster algorithm. Namely, we not only use the distance property between centroid points, but also consider the variance of normals and the curvature information as the reference to construct the feature tensor, and finally apply the tensor-based spectral clustering to group centroid points. The main steps of our clustering are described below.

1. We first construct the affinity matrix W^t by $W^t(i,j) = e^{-metric(cP_i, cP_j)^2/2\sigma^2}$, where $\sigma = 1/m^2 \sum_{i,j}^{m} metric(cP_i, cP_j)$, $t = 1, 2, 3$, and m is the number of points. For $t = 1$, $metric(cP_i, cP_j) = \|cP_i - cP_j\|$ is the Euclidean distance. When $t = 2$, $metric(cP_i, cP_j) = \angle(N_i, N_j)$ is chosen as the small angle difference of normals N_i and N_j at cP_i and cP_j, respectively. When $t = 3$, $metric(cP_i, cP_j) = |curvature(cP_i) - curvature(cP_j)|$, which is set as the small curvature difference between cP_i and cP_j. Note that the normal and curvature of point cP_i (or cP_j) are inherited from its feature point or its relevant point.
2. Normalize W^t to $L^t = D^{-\frac{1}{2}} W^t D^{-\frac{1}{2}}$, where $D = diag(d_1, d_2, \ldots, d_m)$, and $d_i = \sum_{j=1}^{m} W^t_{ij}$.
3. Form a tensor T by L^1, L^2, L^3, and unfold T into $T^{(n)}$ by mode-n unfolding, $n = 1, 2, 3$.

4. $T^{(n)}, n = 1, 2, 3$ are three different matrices and each one is separately decomposed by the singular value decomposition (SVD), the column vectors (called left singular vectors) of the obtained unitary matrix are the principal components of the data set $T^{(n)}$ [WRR03]. The singular values are $\lambda_i^{(n)}$ and the corresponding left singular vectors are $\phi_i^{(n)}, i = 1, 2, \ldots, m$. We first normalize the singular values $\lambda_i^{(n)}$ such that they are between 0 and 1, and then modify them by multiplying with the weighted values $\omega^{(n)}, n = 1, 2, 3$. When setting the weight values $\omega^{(n)}, n = 1, 2, 3$, we assume the distance metric has more influence than the variance of normal vectors and the curvature for the singular values $\lambda_i^{(n)}$, so we normally set $\omega^{(1)}$ to a value in [0.6, 1], and $\omega^{(2)}$, $\omega^{(3)}$ are set to smaller values, which are in [0, 0.4]. Finally, we sort the modified $\omega^{(n)}\lambda_i^{(n)}(i = 1, 2, \ldots, m)$ in a descending order and choose the top-ranked K singular vectors to form the spectral space $V = [\phi_1, \phi_2, \ldots, \phi_K] \in R^{m\times K}$.
5. Representing cP_i by $y_i \in V$, we cluster $Y = \{y_i \in V | 1 \le i \le m\}$ using the K-means clustering, and the corresponding centroid points are then clustered.

In the traditional spectral clustering, the value of K (i.e. the number of the clusters) is experimentally set by the user in advance. Here, K is adaptively determined according to the matrix perturbation theory [CLX*12] in our tensor-based spectral clustering, which reports that the larger the eigengap is, the more stable the subspace formed by the chosen eigenvectors is. Similarly, we suppose $b_i^{(n)} = \omega^{(n)}\lambda_i^{(n)} - \omega^{(n)}\lambda_{i+1}^{(n)}(i = 1, 2, \ldots, m-1)$ is the difference of singular values, if $|b_i^{(n)} - b_{i+1}^{(n)}|$ is the first extreme maximum, then the subindex i is chosen as the cluster number K.

Compared to traditional spectral clustering and other clustering methods such as *KNN* algorithm, the above tensor-based spectral clustering can help us obtain the curve skeleton with more correct branches and guarantee the componentwise differentiation property. Figure 9 shows an example to illustrate the effectiveness of our tensor spectral clustering, where Figure 9(a) shows the result by the traditional spectral clustering, Figure 9(b) shows the result by the distance and normal direction [MWF*10] and Figure 9(c) shows the result by our tensor-based spectral clustering. For the ox model in Figure 9, although the spectral clustering with distance and normal direction can segment four legs, the head region with rich details is not effectively segmented, while there exists oversegmentation in the body part. In contrast, our method can obtain satisfactory segmentation results. For example, there are three clusters in the head of the ox model, four clusters in the leg regions and one cluster in the body part of the ox model.

(a) Traditional spectral clustering with distance (b) Spectral clustering with distance and normal [MWF*10] (c) Our spectral clustering with distance, normal and curvature

Figure 9: *Clustering comparison among different spectral clustering methods.*

5.2. Region growing clustering for centroid points in the same group

After the above tensor-based spectral clustering, some clusters (groups) may have more than sufficient number of centroid points, which could cause unnecessary redundancy for the curve skeleton. We use the region growing clustering algorithm [MOG11] to reduce the centroid points in each cluster. We first sort the centroid points in each group G in a descending order (normally by the z-coordinate), and then set the point with the maximum z-coordinate as the first seed point. We search for its k nearest points to form the first sphere region r_1 by the *KNN* algorithm, and compute the centre C_1 of r_1. k represents the number of different centroid points in each region of a given group. It is an important parameter in this region growing clustering, which influences the density of skeletal points. In our experiments, k is experimentally set to 8. Second, we look for the furthest point in r_1 from C_1 and choose it as the next seed point. We search for the corresponding k nearest points excluding the previously processed points and form the second region r_2, and so on. Therefore, the group G will be divided into $[\|G\|/k]$ regions and the residual points are assigned to the closest regions, where $\|G\|$ denotes the number of points in the set G, $[x]$ is a rounding function that rounds the value of x to the nearest integer smaller than or equal to x. Finally, we update the centre C of each region and use it to represent the points within the corresponding region. We use sP_i to denote the centre of a region in the same group, then $\{sP_i\}$ forms the skeletal point set sP and represents the resulting regions, which are shown in Figure 3(e).

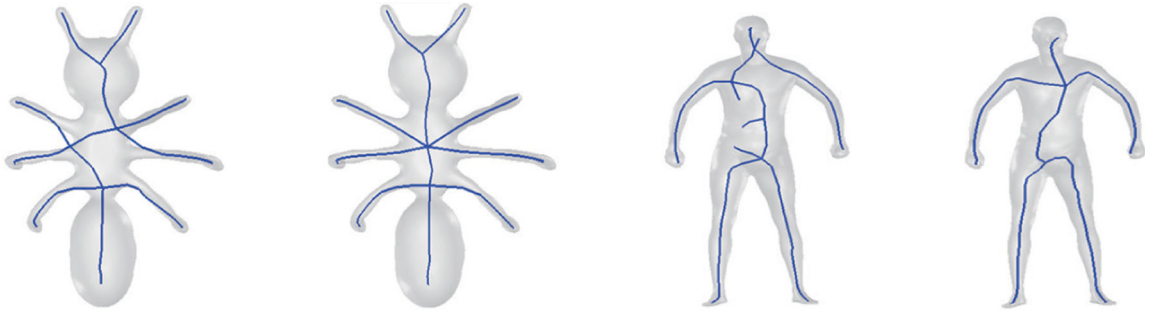
It is noteworthy that if $\|G\| \le k$, that is, the number of points in G is fewer than or equal to k, it is not processed by the above region growing clustering and each point in G will be retained as a skeletal point. In order to extract the desired curve skeleton, the above process is iteratively called to reduce the skeletal points from the overdense centroid points to a suitable number. We set the termination condition of the iterations as satisfying the condition that the density D of points in G is below a threshold τ (in our experiments, τ is set as 20).

Note that the coordinates of points in different point cloud models may have different values in scale. In order to ensure the density condition is suitable for all 3D models, we use the following Equation (4) to regularize the volume of group G:

$$V = \frac{\frac{4}{3}\pi r^3}{vol(Box)}, \tag{4}$$

where r is the radius of group G and $vol(Box)$ is the volume of the bounding box enclosing the point cloud model. Because the sphere of group G and the AABB of the model have been constructed in the above region growing clustering step and the distance field substitution, we can calculate this regularized volume easily. Then the density of points in G is computed by $D = \|G\|/V$, which describes the number of points per unit volume.

In the above two-steps clustering, the first step is to divide the approximated centroid points into different groups by the spectral



(a) Skeleton without spectral clustering (b) Skeleton with spectral clustering (c) Result without *KNN* clustering (d) Result with *KNN* clustering

Figure 10: *The comparison with and without the spectral clustering for the spider model, and the comparison with and without the KNN clustering for human model.*

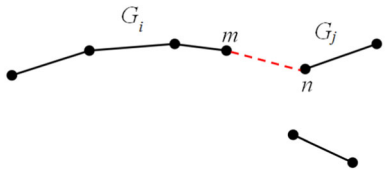


Figure 11: *A connected component is merged with its closest component by connecting the points m and n, where m is on G_i, and n is on G_j.*

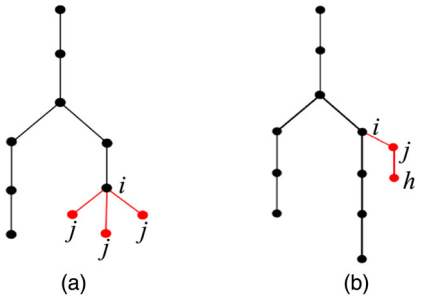


Figure 12: *The spurious branch pruning: the red edges are spurious branches.*

clustering. Without this step, it could be difficult to keep the correct shape of the skeleton. We give examples in Figures 10(a) and (b). The second step is to use the *KNN* clustering to reduce the centroid points in the same group to represent the curve skeleton. Without this step, the skeleton may produce some trivial branch chains because of the redundant centroid points. We show examples in Figures 10(c) and (d). If instead of *KNN* clustering, tensor clustering was used again inside each group, it is hard to obtain the clustering result and would cause the centroid points be less centred.

6. Constructing the Connected Skeletal Graph

After obtaining the skeletal points set sP, we employ the idea of [LGS12] to perform skeletal graph connection, skeleton pruning,

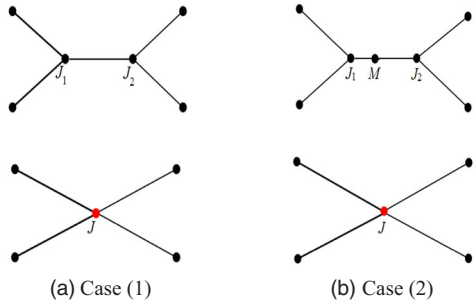


Figure 13: *Junction points are trimmed.*

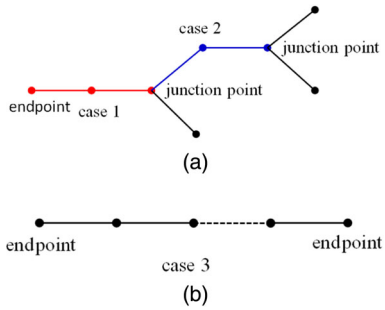


Figure 14: *Three cases of chains: the red chain in (a) is the case 1, the blue chain is the case 2 and the chain in (b) is the case 3.*

overjunctions merging, skeleton smoothing and loop closure. We first construct the primary adjacency relationships among the skeletal points. For each skeletal point in sP, we search for its nearest point and then connect them to obtain the initial primary adjacency list E, shown in Figure 3(f), which is composed of connected components $G_1, G_2, \ldots, G_t$.

Considering a given connected component G_i, we search for its closest connected component G_j, and find the minimum Euclidean distance between two points $m \in G_i$ and $n \in G_j$, connect them to merge G_i and G_j into one connected component, as shown in Figure 11. The edge (m, n) is added into the initial primary

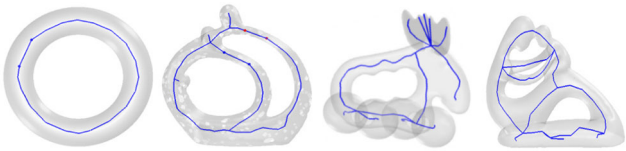


Figure 15: *Curve skeletons extracted from shapes with various genus. Note that in the leftmost (first) case, one blue point is the endpoint of the chain, the other blue point is the closest endpoint on this chain. Here, feature points are extracted according to distance field substitution. In the second case, two blue points and two red points are similar endpoints. The numbers of points from four models (from left to right) are 4800, 12 750, 24 013 and 5000, respectively. The sampling densities (from left to right) are 17.8934, 28.7827, 29.3909, 19.9357, respectively.*

Table 1: *The default values of all the parameters in our algorithm.*

Parameter	k	λ	θ	τ	$\omega^{(1)}$	$\omega^{(2)}$	$\omega^{(3)}$	α
Default value	8	1.2	-0.5	20	0.7	0.3	0.3	0.1

adjacency list E. We iteratively repeat this process until all the connected components are merged into one connected component G, which forms the primary skeletal graph, as shown in Figure 3(g).

6.1. Skeleton pruning

In the primary skeletal graph, there could exist spurious branches whose edge lengths are small around the endpoint area of the skeleton graph. We detect spurious branches based on the lengths of the edges with one endpoint. For example, considering an edge $E(i, j)$ shown in Figure 12(a), if node j is an endpoint whose degree is 1 and the regularized distance of $E(i, j)$ is smaller than a threshold α, then this edge will be pruned. Here, the regularized distance is computed as $length(E(i, j))/\max\{distance(u, v)\}$, where u and v are two arbitrary points in the skeleton, which guarantees that α is suitable for other parts of 3D models. We also delete the edges according to the length from the endpoint to the closest junction point. For example, for the edges $E(i, j)$ and $E(j, h)$ shown in Figure 12(b), if node i is a junction and node h is an endpoint, and the regularized distance from i to h is smaller than a threshold α, then the edges $E(i, j)$ and $E(j, h)$ are removed. The completely pruned skeleton graph as an example is shown in Figure 3(h).

6.2. Overjunctions merging

In the primary skeletal graph, there could exist some overjunctions due to redundant skeletal points. We merge some edges and points related to junction points. For example, considering an edge $E(i, j)$ shown in Figure 13(a), if junctions J_1 and J_2 are directly adjacent, we replace them by J whose coordinate is computed as $J = (J_1 + J_2)/2$. For an intermediary point M which is connected by junctions J_1 and J_2, we replace J_1, J_2 and M by J whose coordinate is computed by $J = (J_1 + J_2 + M)/3$, which is shown in Figure 13(b).

6.3. Curve skeleton smoothing

After the above steps, for a point cloud model with noise, some skeletal points (in particular, those added skeletal points in the above steps) may not be centred. Therefore, we need to adjust their positions to obtain a smooth curve skeleton. Inspired by the previous work by Au *et al.* [ATC*08] that employs Laplacian contraction to obtain a smooth curve skeleton for a mesh model, we use a point-based Laplacian smoothing strategy [Tau95] to smooth a skeletal point V_i in the curve skeleton, described below.

$$\begin{cases} V_1' = V_1 \\ V_i' = \frac{1}{4}V_{i-1} + \frac{1}{2}V_i + \frac{1}{4}V_{i+1},\ i = 2, 3, \ldots, N-1 \\ V_N' = V_N. \end{cases} \tag{5}$$

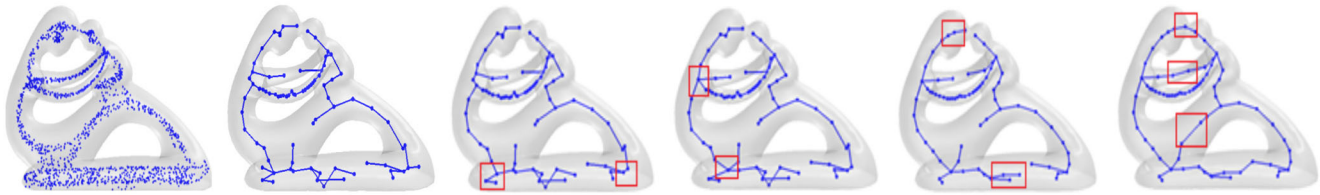
In the connected skeletal graph, we decompose it to some branch chains using the Depth First Search (DFS) algorithm starting from a junction point (if not existing, from the first endpoint). There exist a total of three cases of chains as shown in Figure 14: (1) Starting from a junction point and ending on an endpoint; (2) starting from a junction and ending at another junction and (3) if the curve skeleton is composed of only one chain (i.e. non-existence of any junctions), starting from the first endpoint and ending on the other endpoint. We smooth each chain according to the above modified Laplacian formula and obtain the smoothed curve skeleton.

Note that the above smoothing process may leave some original skeletal points off-centred. During the above clustering and over-junction merging steps, we only smooth those skeletal points with the changed positions, and do not apply the smoothing steps to junction points connecting different groups (resulted from the spectral clustering step). This can help us to keep the centredness of the curve skeleton as much as possible.

6.4. Loop forming for models with non-zero genus

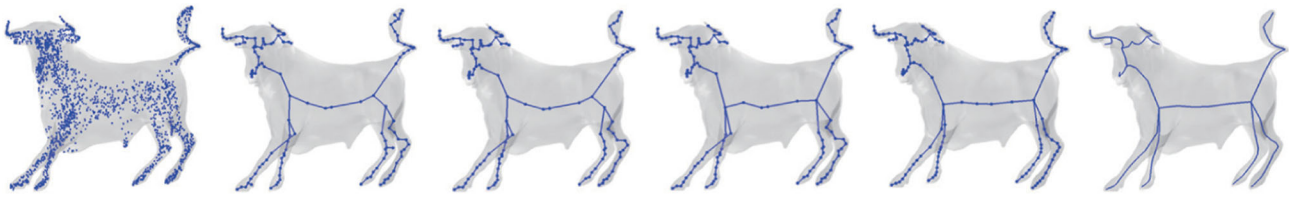
Given a model with non-zero genus, in order to extract its topologically-equivalent curve skeleton, we need to add some edges to form loops on the pruned and merged curve skeleton. For a point cloud model with one genus, we form a loop according to the closest distance between the endpoints of chains in the primary skeleton graph. For a point cloud model whose genus is greater than 1, each endpoint of the chain is judged whether it is connected to other endpoints of chains according to the distance. Experimentally, if the distance between this pair of endpoints is not larger than twice of the average edge length in its *KNN* neighbourhood, we connect them to form the skeleton with loops.

Figure 15 shows the results of several point cloud models with non-zero genus by our method. In this figure, the first is a point cloud model with one genus, and its curve skeleton is extracted correctly. While others include complex shapes with high genus, and our method can also extract their reasonable curve skeletons. For a fertility model shown in Figure 15, we provided corresponding results of different steps in Figure 16. Besides, we use another ox model to show the results of different steps during our skeleton extraction in Figure 17.



(a) Overly dense skeleton points (b) Initial skeleton (c) Skeleton after pruning (d) Skeleton after merging of branching nodes (e) Skeleton after more aggressive smoothing (f) Skeleton after topological fixing (for genus > 0)

Figure 16: *Results of different steps for the fertility model (some variances are shown in the red rectangles).*



(a) Overly dense skeleton points (b) Initial skeleton (c) Skeleton after pruning (d) Skeleton after merging of branching nodes (e) Skeleton after progressively more aggressive smoothing (f) Curve skeleton after removing skeleton point

Figure 17: *Results of different steps for the ox model.*

Table 2: *Skeleton extraction time comparison of the Laplacian mesh contraction [ATC*08] and our method for the three models in Figure 23.*

Models	Point number	Feature points number	Skeleton points number	Laplacian Mesh contraction [ATC*08] time (s)	Our method time (s)
Dinosaur	23 984	922	89	16.32	15.38
Bunny	5000	362	49	2.86	1.97
Hand	1557	515	84	1.12	1.65

Table 3: *Skeleton extraction time comparison of the L_1-medial method [HWCO*13] and our method for the models in Figure 21.*

Models	Point number	Feature points number	Skeleton points number	L_1-medial time (s)	Our method time (s)
Woman	44 230	425	71	16.90	7.95
Rhino	79 934	1590	68	210.86	85.86

7. Experimental Results and Discussion

We tested our skeleton extraction algorithm on a variety of point cloud models, including noisy models (the second model in Figure 15 and the last two models in Figure 18), models with holes (the last four models in Figure 18), models with high genus (Figure 15) and other models in Figure 18. We also analysed various properties of the extracted curve skeletons, including correctness, centredness, homotopic and componentwise differentiation, efficiency, reliability and robustness and invariant under different poses or deformations. Finally, we compared our algorithm with state-of-the-art curve skeleton extraction methods.

Table 1 lists the default values of the parameters in our approach. Note that although we extracted the curve skeletons directly from input point cloud models, for a more intuitive visualization, we rendered the models with their corresponding mesh representations in a transparent colour, and showed their final skeletons in blue. For all the tested models, we referred to the local density of each point as in the L_1-medial method [HWCO*13], and computed the average of the local densities of all the points as the sampling density.

7.1. Discussion

Correctness and centredness properties: For a densely sampled point cloud model, our method captures hybrid feature points through geometric features and distance field substitution, performs the skeleton-guided normal vector optimization, tensor-based spectral clustering and removes spurious relevant points. These steps help us to obtain the correct curve skeleton. Also, we smooth

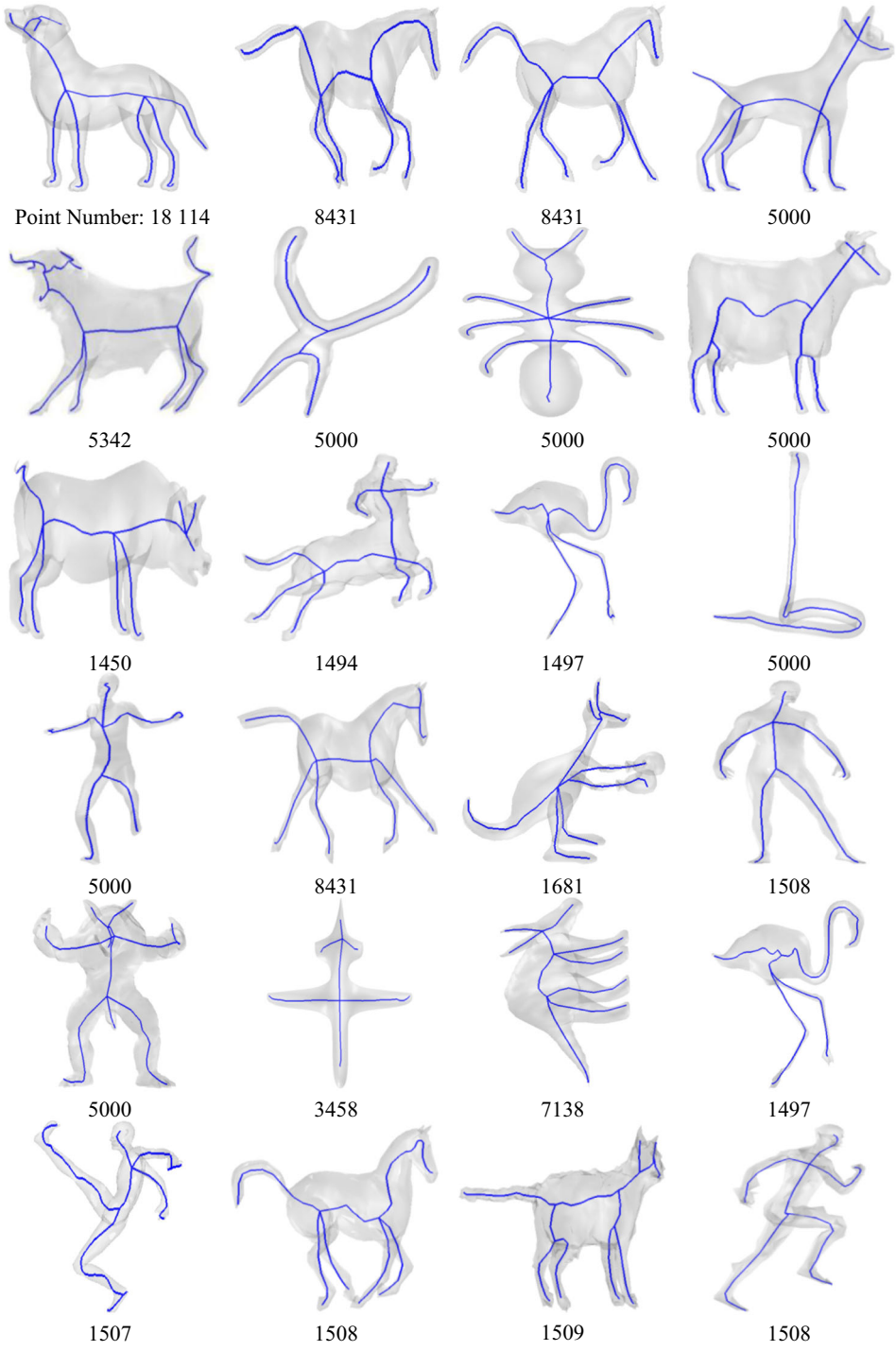
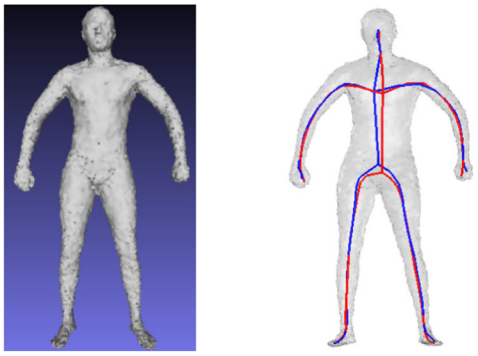


Figure 18: *The extracted skeletons by our method. The numbers of points from all the models vary from 2K to 20K. The average of the sampling densities for all the models is 48.1941.*

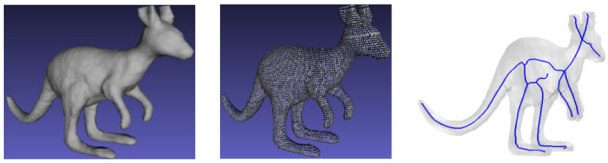
those skeletal points whose positions are changed in the clustering and merging processes, so we can keep the resulting curve skeleton centred as much as possible. We provide the quantitative analysis about the centredness of the resulting skeletons in the following comparisons section.

The noise or missing data of an input point cloud model influences the quality of the resulting skeletons by our method, for example, the skeleton is sometimes off-centred or out of the shape. When noise or the amount of the missing data is not significant, more hybrid feature points and other mentioned measurements in



(a) Man model with noise (b) Curve skeletons

Figure 19: *Skeletons extracted from a man model with noise (a). In (b), the red skeleton is extracted from the model without noise. The blue skeleton is extracted from the model with noise. The number of points is 15 000, and the sampling density is 60.0636.*



(a) A kangaroo model (b) Raw scan (c) Curve skeleton

Figure 20: *The skeleton (c) of a kangaroo model. (a) is the mesh model and (b) is the corresponding point cloud model. The number of points is 9673, and the sampling density is 45.8795.*

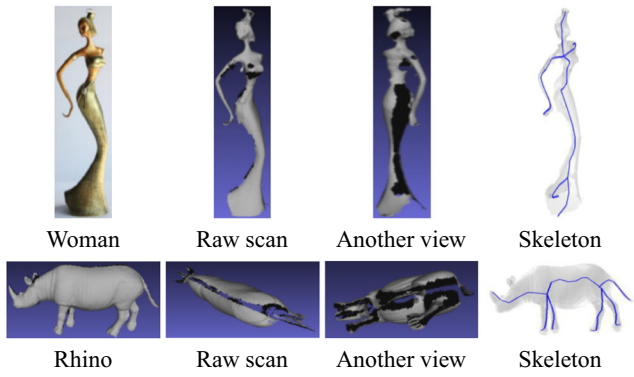


Figure 21: *Example curve skeletons extracted from incomplete point cloud models. The number of points on the woman model is 21 097, and its sampling density is 102.1090. The number of points on the rhino model is 79 934, and its sampling density is 66.4723.*

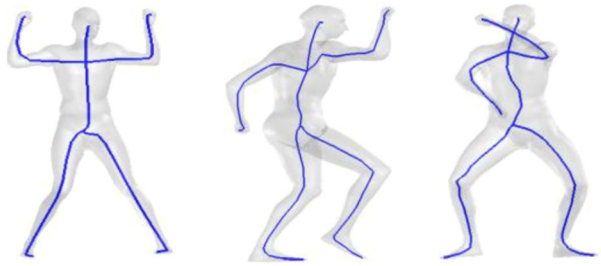


Figure 22: *Invariance under different poses or deformations. The number of points is 15 000, and the sampling density is 58.9850.*

our method help to keep the correctness and centredness of the resulting skeleton. For a man model with noise in Figure 19 and other models with missing data in Figure 21, we can see the extracted curve skeletons correctly keep the shapes of the models.

Homotopic and componentwise differentiation: Since in our method each skeletal point is extracted as the average of its local feature point and relevant point, it can represent the local shape of the input point cloud model. So, the resulting curve skeleton can preserve the topology of the input model. Meanwhile, we use the tensor-based spectral clustering for these skeletal points, so that the clustered skeletal points can reflect the details of the shape. As shown in Figure 20, the skeleton captures the correct topological relationship (such as limbs and connections) between different body parts, even though different parts of the model have significant differences in terms of size and shape.

Computational efficiency: The configuration of our experimental computer was an Intel core i7-4710MQ cpu@2.50 GHz with 8 GB RAM. In order to test the computational efficiency of our method, we compared the skeleton extraction time of the Laplacian mesh contraction [ATC*08] with that of our method for the three models in Figure 23, listed in Table 2. We cannot find significant difference between our method for a point cloud model and the Laplacian mesh contraction [ATC*08] for its corresponding mesh model. We also compared our method with the L_1-medial method [HWCO*13] for two point cloud models in Figure 21, listed in Table 3. In the original works of [SPJX18, HWCO*13], the authors reported that the L_1-medial method [HWCO*13] is faster than the ROSA method [TZCO09] and the distance field guided L_1-median method [SPJX18], while our method is significantly faster than the L_1-medial method [HWCO*13] (refer to Table 3). For the complexity of the algorithm, although we add the normal optimization and the tensor-based spectral clustering, the computational efficiency of our skeleton extraction method is higher than the L_1-medial method because our method does not need iterations to obtain the approximated centroid points. Note that the running times in Tables 2 and 3 were not reported on the optimized code for all the methods including the L_1-medial method, the ROSA method and our approach. If the code is further optimized, the running times for all the three methods would be reduced. Note that computational scalability is not the main goal of our method, and we are aware that some existing methods can achieve a better scalability (e.g. [JKT13]).

Generality and robustness: In order to verify the generality and robustness of our skeleton extraction method, we tested it on some point cloud models with incomplete data and models with significant missing regions, as shown in Figure 21 (also refer to Figures 25–28), the first column shows the origin models, and the second and third columns show the raw scans with different views (the black regions are caused by significant missing data), the extracted curve skeletons, shown in the fourth column, are clean and

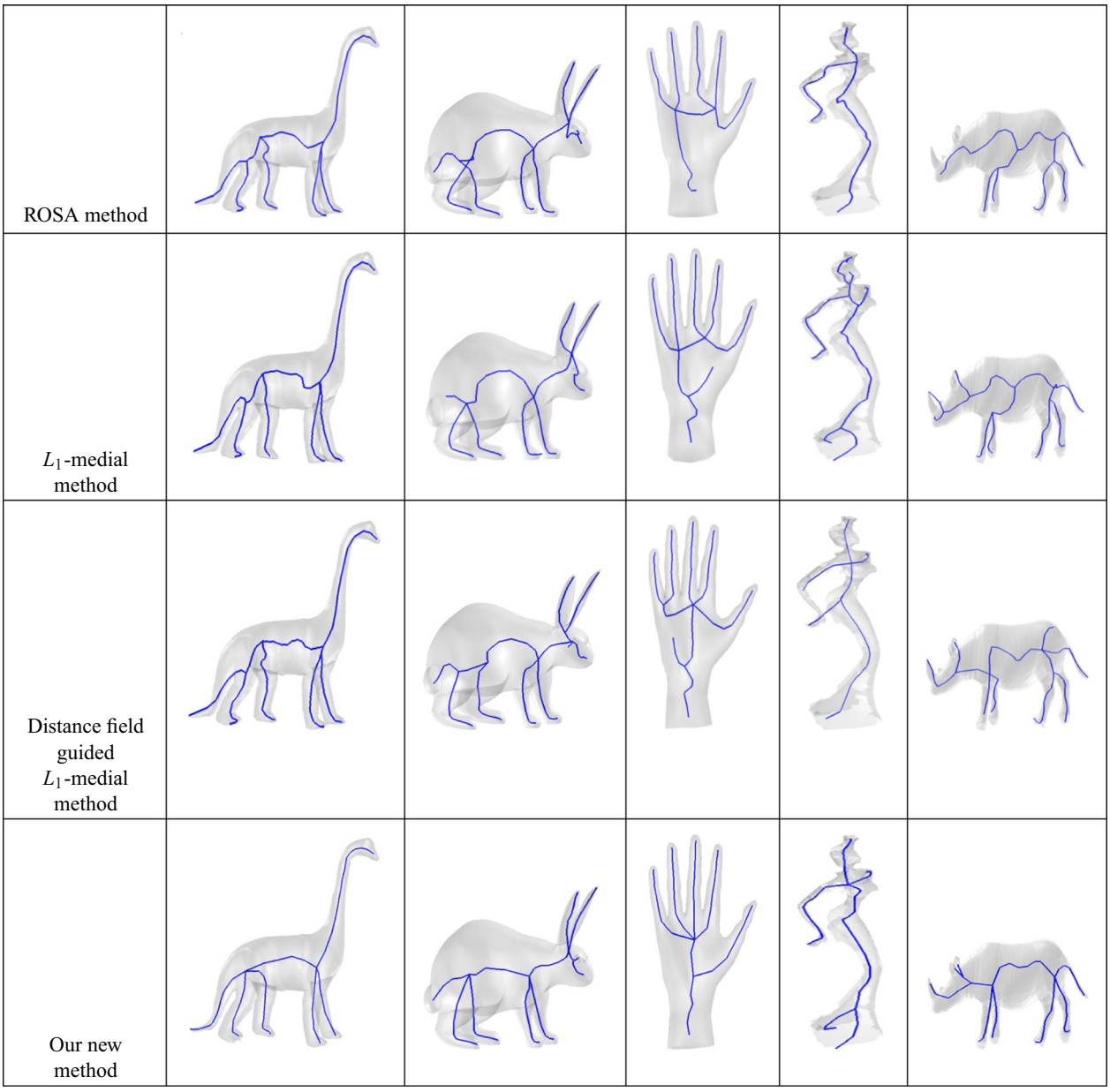


Figure 23: *Comparisons between our method and three state-of-the-art skeleton extraction methods. The number of points on the dinosaur model is 23 984, and its sampling density is 59.5170. The number of points on the bunny model is 5000, and its sampling density is 28.8505. The number of points on the hand model is 1557, and its sampling density is 25.3355. The number of points on the woman model is 21 097, and its sampling density is 102.1090. The number of points on the rhino model is 79 934, and its sampling density is 66.4723.*

correct. Note that the missing data still have influence on the resulting skeleton. For example, for the woman model in Figure 21, the skeleton cannot locally express the breasts due to the missing data in the breast regions. For the rhino model, the skeleton cannot locally express small ears because of the missing data in that area. Besides, the missing data also cause some parts of the curve skeleton off-centred.

For a point cloud model with significant missing data, we extract feature points by not only geometric feature but also distance field substitution. As long as the missing data on the cross-sectional contour is fewer than 50%, our method can typically find the corresponding relevant points for feature points. So, we can determine the corresponding skeletal points and form the curve skeleton for the input model.

(a) L_1-medial (b) Distance field guided L_1-median (c) Our method

Figure 24: *Comparisons among the three methods (L_1-medial [HWCO*13], Distance field guided L_1-median [SPJX18], our method) for a dog model. The number of points is 18 114, and the sampling density is 102.1069. Note that the results by the L_1-medial method and the distance field guided L_1-median method are taken from the original papers.*

The noise on a point cloud model could cause the resulting skeletal point positions off-centred, in particular, during the feature extraction and relevant point determination steps in our algorithm. However, this issue can be to a certain extent fixed or at least alleviated through the skeleton refinement step in our algorithm.

Invariance under different poses or deformations: Our skeleton extraction approach is invariant under different poses or deformations for input point cloud models, which is achieved by our skeleton-guided normal vector estimation and feature points shifting steps. As shown in Figure 22, for the three different poses of a human model, their curve skeletons are extracted correctly and topologically-equivalent.

In addition, other properties such as smoothness and thinness can be achieved by our skeleton smoothing, point cloud simplification and feature points shifting processes. Our skeletons extracted by our approach satisfy most of the properties proposed by Cornea *et al.* [CSM07].

7.2. Comparisons

We compared our results with three state-of-the-art skeleton extraction algorithms: the ROSA method [TZCO09], the L_1-medial method [HWCO*13] and the distance field guided L_1-median method [SPJX18], for point cloud models without missing data. The comparison results for five test models are shown in Figure 23. All these skeleton extraction methods can reasonably preserve the topologies of the models. Their difference is that, for the dinosaur model, the curve skeleton by our method has better connections between the tail and the body compared to the other three methods; for the bunny model, the head part of the curve skeleton by our method is more approximate to the shape than the other three methods and for the hand model, the curve skeleton by our method does not have spurious branches, and is more closer to the centre and better represents the shape of the object. For the woman model, the curve skeleton by our method has correct connections and branches; for the rhino model, the curve skeleton by our method has better centredness and no spurious branches than the other three methods. Note that the centredness of the skeleton is important for many practical applications. Our method also cannot always satisfy this property. For example, for the bunny and hand models in Figure 23, all the methods cannot produce perfectly centred skeletons.

We also compared our approach with the three chosen methods for models with incomplete and significant missing data. As shown in Figure 24, compared to the L_1-medial method [HWCO*13] and the distance field guided L_1-median method [SPJX18], we found the skeletons by our method had more details in the head of the dog model (i.e. two ears are represented by two skeletal branches). The results of an incomplete lady model with complex shapes are presented in Figure 25. We can see that the skeleton by our method is better than both the ROSA method and the L_1-medial skeleton method, since some errors exist in the blue-box regions, by contrast, the skeleton by our method does not have such errors.

For an incomplete dinosaur model in Figure 26, we found that the curve skeleton by our method is as good as that by the L_1-medial method. But when the missing data are increased on a human model as shown in Figure 27, we found our method can still keep the property of the skeleton while the L_1-medial method gradually extracts spurious branches and wrong parts of the skeleton. We also tested the results when we removed points with 10%, 20% and 30%, respectively, in a uniform way from the original human model. We found our method can still extract satisfactory skeleton results, as shown in Figure 28. For a model with complex shape and large missing regions, our method can achieve more satisfactory skeleton results than existing methods mainly because it uses hybrid feature points not only from geometric information but also from the distance field substitution for skeleton construction.

We also designed a quantitative criterion for the centredness evaluation of the curve skeleton [Cor07]. First, the curve skeleton is uniformly sampled. For each sample point O_m, a plane perpendicular to its skeletal direction is constructed. The skeletal direction at the sample point O_m can be calculated through the neighbouring sample points of O_m. Then, the intersection points with the point cloud model are found by uniformly emitting rays from O_m in this plane, and the distance between each intersection point C_i and O_m is calculated. We employ a tolerance to determine whether a point P_j in the point cloud model is in the plane perpendicular to the curve skeleton at the sample point O_j. If the angle φ between the line P_jO_j and the plane is smaller than a threshold (10° in our experiments), P_j is considered in the plane for the intersection computation. For each point C_i in the intersection set, if another point C_j is found in the intersection set to satisfy that C_i, O_m and C_j are almost on a straight line, the shorter and longer distances in $|C_iO_m|$ and $|C_jO_m|$ can be set as $lmin$ and $lmax$, respectively, then $LC_i = lmin/lmax$ is recorded as the centredness degree about the point C_i. The average value of all LC_i is calculated as the centredness degree of the skeleton point O_m. Finally, the average value of the centredness degree of all skeleton sampling points is defined as the centredness degree of this skeleton. Therefore, the larger the calculated centredness degree, the more centred the skeleton. Note that if point O_i is a skeletal junction point, we compute the centredness degree of its neighbouring skeletal point in each skeletal branch, respectively, and then take the average of the centredness degrees in all skeletal branches as the centredness degree at the junction point (see Figure 29).

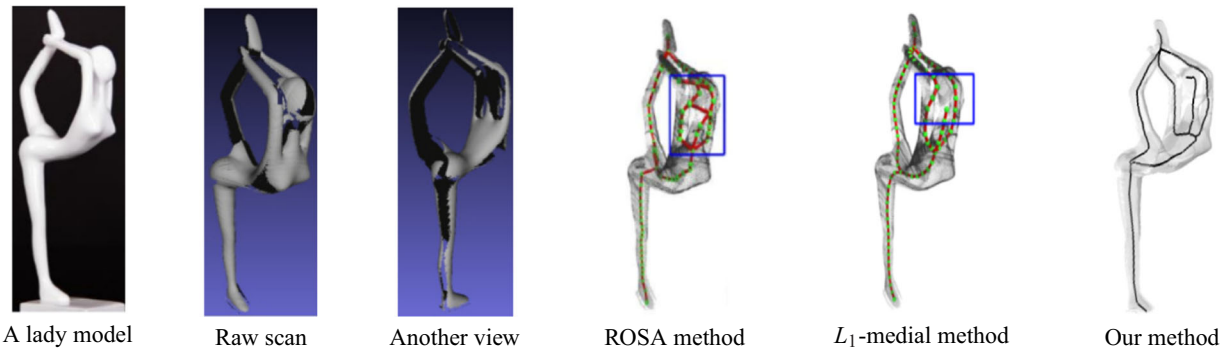


Figure 25: *Comparisons among the three methods (ROSA [TZCO09], L_1-medial [HWCO*13] and our method) for an incomplete lady model. The number of points is 41 238, and the sampling density is 138.6774. Note that the results by the L_1-medial method and the ROSA method are taken from the original papers.*

Figure 26: *Comparison between the L_1-medial [HWCO*13] method and our method for an incomplete dinosaur model. The number of points is 57 210, and the sampling density is 114.1270. Note that the result by the L_1-medial method is taken from the original paper.*

Table 4: *The centredness results of the five models in Figure 23.*

	Dinosaur	Bunny	Hand	Woman	Rhino
ROSA method	0.766808332	0.803718848	0.906167574	0.792531224	0.745892033
L_1-medial method	0.772488796	0.808567351	0.921160137	0.803814563	0.739299560
Distance field guided L_1-medial method	0.774012757	0.830179174	0.915827802	0.791945545	0.720106851
Our method	0.836542762	0.840244916	0.925785687	0.824963467	0.771208026

Table 5: *The centredness of the human model shown in Figure 27 with different missing data.*

	Original model with points: 15 000	Model with points: 13 377 Missing data: 10.82%	Model with points: 11 992 Missing data: 20.05%	Model with points: 8849. Missing data: 41.01%
L_1-medial method	0.837643214	0.755438790	0.742154331	0.703428233
Our method	0.849560674	0.858944282	0.843579911	0.832942352

Taking a cylinder model as the ground truth, we compared the curve skeleton and the centredness degree between our method and state-of-the-art methods in Figure 30. We reported the calculated centredness results in Table 4 and showed the advantage of our method by comparing it with different algorithms for five models in Figure 23. Besides, we provided the centredness values in Table 5 for the human model shown in Figure 27 with different missing data. We found that our obtained skeleton is optimally centred with respect to the original model, compared to the skeletons extracted by the other methods. For the original human model and model with Gaussian noise ($\sigma = 0.003$) as shown in Figure 19, we also gave the centredness result with 0.851510321 and 0.831485361, which

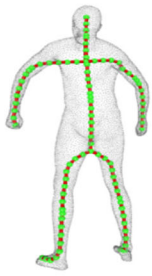
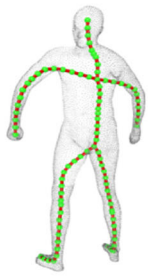
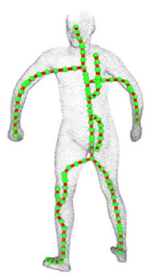
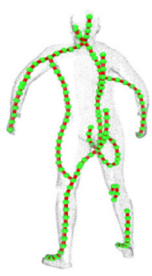
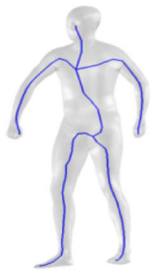
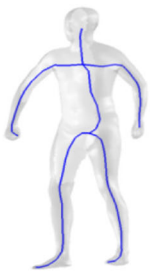
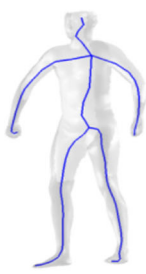
	Original model with points: 15000	Model with points: 13377. Missing data: 10.82%	Model with points: 11992. Missing data: 20.05%	Model with points: 8849. Missing data: 41.01%
Human model				
Skeleton by L_1-medial method [HWCO*13]				
Skeleton by our method				

Figure 27: *Comparison between the L_1-medial method and our method for a human model with different missing data. Here, the number of points from original model is 15 000, the sampling density is 63.3401.*

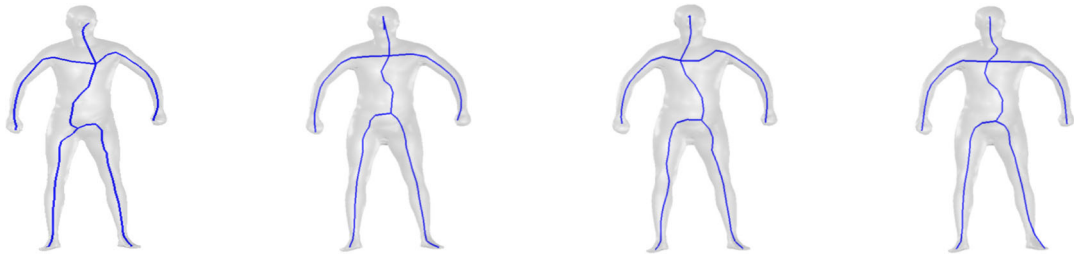
indicates that the centredness of our extracted skeleton is less influenced by noise.

7.3. Limitations

Our current method is suitable for dense point cloud models and has some limitations. The skeletonization from point clouds, especially from raw data with significantly missing regions, is an ill-posed problem by nature. Our method cannot completely solve this open problem. In particular, how to eliminate or bound the violation of the centredness of the skeleton and preserve the topology is still an unresolved challenge for practical applications. Our work provides an effective alternative for skeleton extraction and can obtain relatively satisfactory skeletons, compared to some existing methods.

Generally, if the missing data as a patch on the contour of a cross-section is large than a half of data on that contour, the relevant point of a given feature point will probably not be found according to the opposite normal from the feature point, our method cannot guarantee the correctness of its corresponding skeletal point. For example, for the human model shown in Figure 27, if the missing data are more than 50%, our method also cannot generate a reasonable curve skeleton.

Another limitation of our current method is that it is not applicable for flat and thin models due to the shape of the cross-sections or a low sampling resolution. For such models, some relevant points could be misidentified during the relevant point determination step according to the opposite normal from the feature point in our algorithm, which could lead to incorrect skeletons at the end. For example, there are some spurious skeletal branches in the wing part



(a) Original model with 15,000 points

(b) Model with 10% points removed uniformly

(c) Model with 20% points removed uniformly

(d) Model with 30% points removed uniformly

Figure 28: *Skeleton results for the human model with different percentage of missing data.*

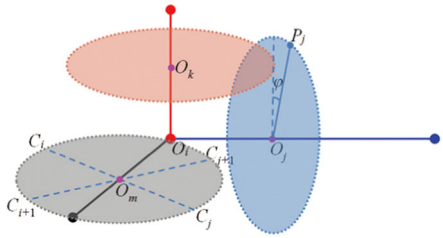


Figure 29: *Centredness degree computation at the skeletal junction point, and the intersection between the plane perpendicular to the skeletal direction and the point cloud model.*

of a bird model, shown in Figure 31. For such models, we need to combine our method with interactive user operations to solve the problem.

8. Conclusion and Future Work

In this paper, we present a novel hybrid feature point-based approach to extract curve skeletons for point cloud models. This is achieved through a series of operations: identify hybrid feature points by combining geometry features and distance field substitution, shift the hybrid feature points according to the skeleton-guided normal directions and the relevant points to locate the centres of an input model and then simplify them by the tensor-based spectral clustering to determine skeletal points, construct the connected curve skeleton and finally prune, trim and smooth it to generate the final skeleton. Through many experiments and comparisons, we show that our method is robust and can efficiently handle point cloud models with noise and significant missing data.

Based on the extracted skeletons from point cloud models, many tasks in digital geometry processing can be further explored. For example, the extracted skeletons can be potentially used for geodesic computation, incomplete surface repair, model mapping, shape analysis, etc, which will be our future work.

Acknowledgements

This work was supported by National Natural Science Foundation of China under Grant No. 11671009 and 11801513, Zhejiang Provincial Natural Science Foundation of China under Grant No. LY19F010014 and LZ19A010002. Xiaogang Jin was supported by the National Natural Science Foundation of China (Grant Nos. 61972344, 61732015) and the Key Research and Development Program of Zhejiang Province (No. 2018C01090). Zhigang Deng was in part supported by NSF IIS-1524782.

Appendix A

Proposition 1. Each feature point always has one, two or three relevant facet(s).

Explanation. For a given bounding box and a feature point in this box, there are three cases for the relevant facet. If the normal vector of a feature point is coincident with the normal of one facet of the box, then this facet is the only relevant facet. If the normal vector of a feature point is parallel with two normal-opposite facets of the box, then there are two relevant facets. If the normal vector of a feature point is not parallel with any facet of the box, then there are three facets as the relevant facets corresponding to the feature point. Figure A1 shows three cases of the relevant facets.

Now let us further assume the number of the relevant facets is more than three for a given feature point, there must be two relevant facets in the six facets of a box whose normal directions are opposite, so we observe that the sum of the two angles between the normal of a given feature point and the normals of two relevant facets is π. According to the definition of relevant facets, the angle between the normal of a feature point and the normal of any relevant facet is smaller than $\pi/2$, this causes a contradiction to the above observation. Therefore, each feature point has at most three relevant facets.

Appendix B

Proposition 2. When searching for the relevant point of a feature point, the method by judging the opposite normal direction may create redundant spurious relevant points. But our method by judging relevant facets can remove more spurious relevant points, and then determine the accurate relevant point through the shortest distance.

Explanation. We consider three cases according to the number of relevant facets.

Method	ROSA method	L_1-medial method	distance field guided L_1-medial method	Our method
Model and skeleton				
Centeredness	0.980422325	0.979861725	0.979247365	0.985610647

Figure 30: *Comparisons between our method and state-of-the-art methods for a ground truth example (i.e. a cylinder model). The number of points is 5043, and the sampling density is 16.2223.*

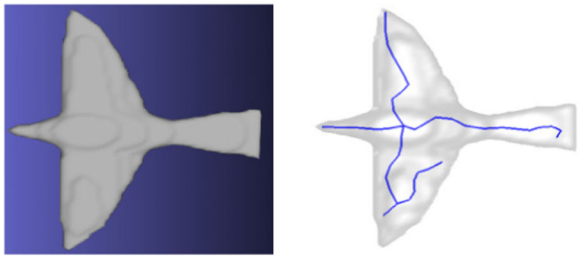


Figure 31: *Skeleton extraction from a bird model. The number of points is 6188, and the sampling density is 48.5677.*

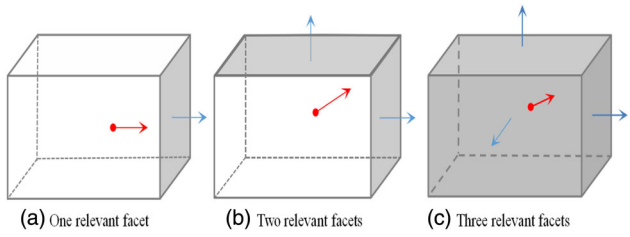


Figure A1: *Three cases of the relevant facets, the red point and red arrow are a feature point and its normal vector, the blue arrow is the normal vector of a relevant facet.*

Case (1): The feature point f has only one relevant facet (for example, the left facet of the bounding box is assumed to be the relevant facet). For convenience, the relevant feature and relevant facet are shown from the top view of the bounding box, which are illustrated in Figure A2(a). The bounding box is then divided into two parts s_1 and s_2 by the plane β crossing the feature point f and parallel to the direction of the relevant facet.

It is obvious that the spurious relevant points of p_2 and p_3 in s_1 will be found by the judgement only according to the opposite normal direction. While in our method, the spurious p_2 and p_3 in the left part s_1 can be eliminated if the distance from the relevant point to the relevant facet is smaller than the distance between the feature point and the relevant facet. And the relevant point p_1 in the right part s_2 can be found by the opposite normal direction and the shortest distance.

Case (2): The feature point f has two relevant facets (for example, the left and front facets of the bounding box are assumed to be the relevant facets). For convenience, relevant feature and relevant facet are shown from the top view of the bounding box, which are illustrated in Figure A2(b). The bounding box is then divided into four parts s_1, s_2, s_3 and s_4 by the planes β_1 and β_2 crossing the feature point f and parallel to each relevant facet.

Intuitively, the spurious relevant points p_2 and p_3 will be found by the judgement only according to the opposite normal direction.

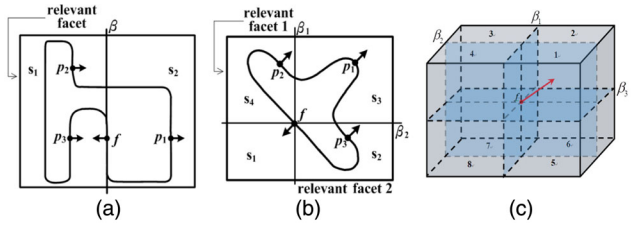


Figure A2: *The positions relation between the feature point, relevant points and relevant facets.*

In our method, the spurious p_2 in s_4, p_3 in s_2 and other spurious relevant points in s_3 can be eliminated if the distance from the relevant point to the relevant facets is smaller than the distance between the feature point and the relevant facets. And, the relevant point p_1 in s_3 can be found by the opposite normal direction and the shortest distance.

Case (3): The feature point f has three relevant facets (for example, the upper, front and right facets of the bounding box are assumed to be the relevant facets, which are illustrated in Figure A2(c). At the feature point f, The bounding box is then divided into eight octants 1–8 by the planes β_1, β_2, β_3, crossing the feature point f and parallel to each relevant facet.

Assuming the normal direction of the feature point f is located in the first octant 1, it is obvious that those spurious relevant points located in the 1–6 or 8 octants can be eliminated according to the rule that the distance from the relevant point to the relevant facets are smaller than the distance between the feature point and the relevant facets. Only one relevant point in the octant 7 can be found by the opposite normal direction and the shortest distance.

References

[ATC*08] Au O. K.-C., Tai C.-L., Chu H.-K., Cohen-Or D., Lee T.-Y.: Skeleton extraction by mesh contraction. *ACM Transactions on Graphics 27*, 3 (2008), 44.

[B*67] Blum H.: A transformation for extracting new descriptors of shape. In *Models for the Perception of Speech and Visual Form*. MIT Press, Cambridge, MA (1967), pp. 362–380.

[BAS14] Bærentzen J. A., Abdrashitov R., Singh K.: Interactive shape modeling using a skeleton-mesh co-representation. *ACM Transactions on Graphics 33*, 4 (2014), Article No. 132.

[BGL15] Benson A. R., Gleich D. F., Leskovec J.: Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining* (Vancouver, BC, Canada, 2015), SIAM, pp. 118–126.

[BHP01] Barequet G., Har-Peled S.: Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms 38*, 1 (2001), 91–109.

[BLM10] Bucksch A., Lindenbergh R., Menenti M.: Skeltre. *Visual Computer 26*, 10 (2010), 1283–1300.

[BN03] Belkin M., Niyogi P.: Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation 15*, 6 (2003), 1373–1396.

[BRW13] Bremer M., Rutzinger M., Wichmann V.: Derivation of tree skeletons and error assessment using LiDAR point cloud data of varying quality. *ISPRS Journal of Photogrammetry and Remote Sensing 80* (2013), 39–50.

[CFD*92] Carello C., Fitzpatrick P., Domaniewicz I., Chan T.-C., Turvey M.: Effortful touch with minimal movement. *Journal of Experimental Psychology: Human Perception and Performance 18*, 1 (1992), 290–302.

[CK11] Chuang M., Kazhdan M.: Fast mean-curvature flow via finite-elements tracking. *Computer Graphics Forum 30*, 6 (2011), 1750–1760.

[CLX*12] Chen X. S., Li W., Xu W. W.: Perturbation analysis of the eigenvector matrix and singular vector matrices. *Taiwanese Journal of Mathematics 16*, 1 (2012), 179–194.

[Cor07] Cornea N. D.: *Curve-Skeletons: Properties, Computation and Applications*. Rutgers University, New Jersey, USA, 2007.

[CSM07] Cornea N. D., Silver D., Min P.: Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization & Computer Graphics*, 3 (2007), 530–548.

[CTK00] Chuang J.-H., Tsai C.-H., Ko M.-C.: Skeletonisation of three-dimensional object using generalized potential field. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 11 (2000), 1241–1251.

[CTO*10] Cao J., Tagliasacchi A., Olson M., Zhang H., Su Z.: Point cloud skeletons via Laplacian based contraction. In *Proceedings of Shape Modeling International Conference* (Aix-en-Provence, France, 2010), IEEE, pp. 187–197.

[DATTS08] De Aguiar E., Theobalt C., Thrun S., Seidel H.-P.: Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum 27*, 2 (2008), 389–397.

[DHKL01] Dyn N., Hormann K., Kim S.-J., Levin D.: Optimizing 3D triangulations using discrete curvature analysis. *Mathematical Methods for Curves and Surfaces 1* (2001), 135–146.

[DS06] Dey T. K., Sun J.: Defining and computing curve-skeletons with medial geodesic function. In *Proceedings of Symposium on Geometry Processing* (Cagliari, Sardinia, 2006), vol. 6, pp. 143–152.

[GSBW11] Ge X., Safa I. I., Belkin M., Wang Y.: Data skeletonization via Reeb graphs. In *Proceedings of Advances in Neural Information Processing Systems* (Granada, Spain, 2011), pp. 837–845.

[HF09] Hassouna M. S., Farag A. A.: Variational curve skeletons using gradient vector flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 12 (2009), 2257–2274.

[HLDZ17] Hu H., Li Z., Dong H., Zhou T.: Graphical representation and similarity analysis of protein sequences based on fractal interpolation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 14*, 1 (2017), 182–192.

[HTRS10] Hasler N., Thormählen T., Rosenhahn B., Seidel H.-P.: Learning skeletons for shape and pose. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Washington, DC, USA, 2010), ACM, pp. 23–30.

[HWCO*13] Huang H., Wu S., Cohen-Or D., Gong M., Zhang H., Li G., Chen B.: L1-medial skeleton of point cloud. *ACM Transactions on Graphics 32*, 4 (2013), 65:1–65:8.

[JKT13] Jalba A. C., Kustra J., Telea A. C.: Surface and curve skeletonization of large 3D models on the GPU. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 6 (2013), 1495–1508.

[JST16] Jalba A. C., Sobiecki A., Telea A. C.: An unified multiscale framework for planar, surface, and curve skeletonization. *IEEE Transactions on Pattern Analysis and Machine Intelligence 38*, 1 (2016), 30–45.

[JT05] James D. L., Twigg C. D.: Skinning mesh animations. *ACM Transactions on Graphics 24*, 3 (2005), 399–407.

[KOF05] Kirk A. G., O'Brien J. F., Forsyth D. A.: Skeletal parameter estimation from optical motion capture data. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (San Diego, CA, USA, 2005), vol. 2, IEEE, pp. 782–788.

[KSO10] Kavan L., Sloan P.-P., O'Sullivan C.: Fast and efficient skinning of animated meshes. *Computer Graphics Forum 29*, 2 (2010), 327–336.

[LCY*18] Lu X., Chen H., Yeung S.-K., Deng Z., Chen W.: Unsupervised articulated skeleton extraction from point set sequences captured by a single depth camera. In *Proceedings of Thirty-Second AAAI Conference on Artificial Intelligence* (New Orleans, LA, USA, 2018).

[LD12] Le B. H., Deng Z.: Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics 31*, 6 (2012), 199:1–199:10.

[LD14] Le B. H., Deng Z.: Robust and accurate skeletal rigging from mesh sequences. *ACM Transactions on Graphics 33*, 4 (2014), 84:1–84:10.

[LGS12] Livesu M., Guggeri F., Scateni R.: Reconstructing the curve-skeletons of 3D shapes using the visual hull. *IEEE Transactions on Visualization and Computer Graphics 18*, 11 (2012), 1891–1901.

[LLW12] Lu L., Lévy B., Wang W.: Centroidal Voronoi tessellation of line segments and graphs. *Computer Graphics Forum 31*, 2 (2012), 775–784.

[LLZM10] Li G., Liu L., Zheng H., Mitra N. J.: Analysis, reconstruction and manipulation using arterial snakes. *ACM Transactions on Graphics 29*, 6 (2010), 152:1–152:10.

[LYO*10] Livny Y., Yan F., Olson M., Chen B., Zhang H., El-Sana J.: Automatic reconstruction of tree skeletal structures from point clouds. *ACM Transactions on Graphics 29*, 6 (2010), 151:1–151:8.

[MOG11] Mérigot Q., Ovsjanikov M., Guibas L. J.: Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics 17*, 6 (2011), 743–756.

[MWF*10] Ma T., Wu Z., Feng L., Luo P., Long X.: Point cloud segmentation through spectral clustering. In *Proceedings of the 2nd International Conference on Information Science and Engineering* (Hangzhou, China, 2010), IEEE, pp. 1–4.

[NBPF11] Natali M., Biasotti S., Patanè G., Falcidieno B.: Graph-based representations of point clouds. *Graphical Models 73*, 5 (2011), 151–164.

[NSK*97] Näf M., Székely G., Kikinis R., Shenton M. E., Kübler O.: 3D Voronoi skeletons and their usage for the characterization and recognition of 3D organ shape. *Computer Vision and Image Understanding 66*, 2 (1997), 147–161.

[Pal08] Palágyi K.: A 3D fully parallel surface-thinning algorithm. *Theoretical Computer Science 406*, 1–2 (2008), 119–135.

[PR02] Pettie S., Ramachandran V.: An optimal minimum spanning tree algorithm. *Journal of the ACM 49*, 1 (2002), 16–34.

[SJT14] Sobiecki A., Jalba A., Telea A.: Comparison of curve and surface skeletonization methods for voxel shapes. *Pattern Recognition Letters 47* (2014), 147–156.

[SLSK07] Sharf A., Lewiner T., Shamir A., Kobbelt L.: On-the-fly curve-skeleton computation for 3D shapes. *Computer Graphics Forum 26*, 3 (2007), 323–328.

[SP08] Siddiqi K., Pizer S.: *Medial Representations: Mathematics, Algorithms and Applications*. Springer Science & Business Media, Dordrecht, 2008.

[SPJX18] Song C., Pang Z., Jing X., Xiao C.: Distance field guided l_1-median skeleton extraction. *Visual Computer 34*, 2 (2018), 243–255.

[SY07] Schaefer S., Yuksel C.: Example-based skeleton extraction. In *Proceedings of Symposium on Geometry Processing* (Barcelona, Spain, 2007), pp. 153–162.

[SYJT13] Sobiecki A., Yasan H. C., Jalba A. C., Telea A. C.: Qualitative comparison of contraction-based curve skeletonization methods. In *Proceedings of International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing* (Uppsala, Sweden, 2013), Springer, pp. 425–439.

[Tau95] Taubin G.: A signal processing approach to fair surface design. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (Los Angeles, CA, USA, 1995), ACM, pp. 351–358.

[TDS*16] Tagliasacchi A., Delame T., Spagnuolo M., Amenta N., Telea A.: 3D skeletons: A state-of-the-art report. *Computer Graphics Forum 35*, 2 (2016), 573–597.

[TZCO09] Tagliasacchi A., Zhang H., Cohen-Or D.: Curve skeleton extraction from incomplete point cloud. *ACM Transactions on Graphics 28*, 3 (2009), 71:1–71:9.

[WBG16] Wu T., Benson A. R., Gleich D. F.: General tensor spectral co-clustering for higher-order data. In *Advances in Neural Information Processing Systems* (Barcelona, Spain, 2016), Neural Information Processing Systems Foundation, Inc. (NIPS), pp. 2559–2567.

[WML*03] Wu F.-C., Ma W.-C., Liou P.-C., Laing R.-H., Ouhyoung M.: Skeleton extraction of 3D objects with visible repulsive force. In *Proceedings of Computer Graphics Workshop* (Hualien, Taiwan, 2003), pp. 124–131.

[WRR03] Wall M. E., Rechtsteiner A., Rocha L. M.: Singular value decomposition and principal component analysis. In *A Practical Approach to Microarray Data Analysis*. Springer, Boston, MA (2003), pp. 91–109.

[ZH04] Zhou Y., Huang Z.: Decomposing polygon meshes by means of critical points. In *Proceedings of 10th International Multimedia Modelling Conference* (Brisbane, Australia, 2004), IEEE, pp. 187–195.

[ZT99] Zhou Y., Toga A. W.: Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics 5*, 3 (1999), 196–209.