The Visual Computer manuscript No.
(will be inserted by the editor)

Ichitaro Yamazaki · Vijay Natarajan · Zhaojun Bai · Bernd Hamann

Segmenting Point-sampled Surfaces

Abstract Extracting features from point-based representations of geometric surface models is becoming increasingly important for purposes such as model classification, matching, and exploration. In an earlier paper, we proposed a multiphase segmentation process to identify elongated features in point-sampled surface models without the explicit construction of a mesh or other surface representation. The preliminary results demonstrated the strengths and potential of the segmentation process, but the resulting segmentations were still of low-quality, and the segmentation process could be slow. In this paper, we describe several algorithmic improvements to overcome the shortcomings of the segmentation process. To demonstrate the improved quality of the segmentation and the superior time efficiency of the new segmentation process, we present segmentation results obtained for various point-sampled surface models. We also discuss an application of our segmentation process to extract ridge-separated features in point-sampled surfaces of CAD models.

Keywords point sets · sampling · features · geodesic distance · normalized cut · topological methods · spectral analysis · multiphase segmentation · hierarchical segmentation

1 Introduction

Point primitives support both simple and flexible modeling of complex shapes and have been widely used to represent various surface models [1,2]. In recent years, the number and usage of high-resolution point-sampled surface models have been rapidly increasing due to improvements in digital scanning technology. In order to classify, match, and explore such a large number of high-resolution point-sampled surface models, an efficient method to extract *features* that distinguish the surface models is becoming increasingly important. In fact, point-set segmentation to extract features of such surfaces in the absence of connectivity information has been studied extensively in the graphics community, and its motivations are well-established [3–5]. In addition, the study of point-set segmentation has been extended to higher dimensions [6], which further motivates the development of an efficient segmentation process. In this paper, we present a process to identify two types of geometric features from pointed-sampled surfaces, namely, *elongated* features such as the legs of a horse and fingers of a hand, and *ridge-separated* features such as the faces of a CAD model. These features are natural choices for applications such as model classification, matching, and exploration.

In an earlier paper [7], we introduced a multiphase segmentation process to extract elongated features in point-sampled surfaces without the explicit construction of a mesh or other surface representation. The preliminary results demonstrated the potential of the segmentation process, but the resulting segmentation was still of low-quality, and the segmentation process could be slow. In this paper, we describe several algorithmic improvements to overcome the shortcomings of the segmentation process. The segmentation results of various point-sampled surface models are presented to demonstrate that the new algorithm not only leads to significantly improved quality of the segmentation, but also reduces the time to compute the segmentation by a factor of up to five. We also discuss an application of our segmentation process to identify ridge-separated features in point-sampled surfaces of CAD models.

The rest of this paper is organized as follows; In Section 2, we first review the previously developed and published methods, besides the one proposed in [7], which are closely related to ours. In Section 3, we review the shortcomings of the earlier segmentation process [7] and

I. Yamazaki · Z. Bai · B. Hamann
Department of Computer Science, University of California, One Shields Avenue, Davis, CA 95616, U.S.A.
E-mail: {yamazaki, bai, hamann}@cs.ucdavis.edu

V. Natarajan
Department of Computer Science and Automation, Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore, 560012, India
E-mail: vijayn@csa.iisc.ernet.in

summarize the algorithmic improvements that are presented in this paper. After discussing the details of the algorithmic improvements in Sections 4 and 5, we present in Section 6 a detailed analysis of the storage and run time requirements of the new algorithm. In Section 7, we discuss an application of our segmentation process to CAD models. In Section 8, we conclude the paper.

2 Related work

Segmenting a surface model into its distinct parts is crucial for several applications, such as modeling [8], metamorphosis [9,10], compression [11], simplification [12], retrieval [13,14], collision detection [15], texture mapping [16], and skeleton extraction [17,18]. Besides our earlier work [7], numerous surface segmentation methods have been developed based on techniques from computer vision [19], load partitioning in finite element methods (FEM) [20], point set clustering in statistics [21], and machine learning [22]. In this section, we review some of the segmentation methods that are closely related to ours, and point out the differences.

Based on their objectives, segmentation methods can be broadly classified into two categories: patch-type and part-type methods [23]. Patch-type methods obtain segments that are topological disks [12,24–26], whereas part-type methods partition a surface into segments that correspond to features [15,18,27–34]. Our methods compute part-type segmentations.

Zhang et al. [33] proposed a feature-based approach for computing a patch-type mesh segmentation for surface parameterization. Their approach identifies a feature by growing a region from a local maximum of the average geodesic distance function and searching for a feature boundary which results in an abrupt increase in size of the surrounded region. Another effective patch-type segmentation method, called multi-chart image geometry method (MCIGM), was proposed by Sander et al. [35]. It is based on a k-means algorithm to minimize the global cost of segmentation, where the cost of assigning a face to a segment is measured by the angle between the normal of the face and average normal of all the faces assigned to the segment. Yamauchi et al. [36] showed that using mean-shift to cluster faces before applying MCIGM makes the method robust against the noise in the input and results in a high-quality segmentation.

Katz and Tal [18] proposed a part-type mesh segmentation method that is based on a k-means algorithm. The cost of assigning a face to a segment is measured by the geodesic distance to the representative face of the segment. The local distance between two connected faces is computed as a weighted sum of their Euclidean and angular distances. Several other successful part-type segmentation methods have been developed based on a watershed technique [30,37]. This technique locates the negative curvature minima that correspond to segmentation boundaries by simulating the accumulation of water into basins. Even though the segmentation methods discussed so far share some similarity with ours, they all assume that a surface is explicitly represented by meshes. Hence, they cannot be applied directly to the point-sampled surface models, in which the connectivity information is not available. On the other hand, our methods operate directly on the input points. For example, the first phase of our segmentation approach to identify features is similar to the watershed technique, but we explicitly identify the saddle points of a discrete function defined over the input points, and assign each point to a segment based on the gradient flow induced by the discrete function

Dey et al. [38] proposed a region-growing part-type approach to segment point-sampled surfaces. Their method first identifies the local maxima of a discrete function defined over explicitly-computed 3D meshes. These local maxima represent distinct features, and input points are assigned to a feature based on the flows induced over the meshes. The first phase of our segmentation process is similar to this region-growing approach, but we operate directly on the input points. An advantage to working in the lower dimension (of the surface, in comparison to that of the 3D meshes) is that our segmentation process is efficient. As a result, our method is between two and eleven times as fast as theirs, while generating segmentations that are highly similar to those of Dey et al. in terms of quality

Spectral analysis of an affinity matrix has been used to segment images [19] and meshes [29,39]. We extend these ideas to collect points that together describe a feature on a surface model.

3 Contributions

We first outline the multiphase segmentation process proposed in [7] to extract elongated features in a point-sampled surface:

1. *Supernode extraction.* Based on the topology of the input points, we first identify sets of points that belong to a common feature. This step is done by constructing a *discrete function* and an associated *gradient flow field* over the input points. Points that flow into a common local maximum of the discrete function belong to a common feature and are represented by a *supernode*. For efficiency, the discrete function is computed from uniformly-sampled surface points. This phase sets the stage for performing hierarchical segmentation in an efficient manner by coarsening the input points into supernodes. In this initial phase, we work with an intrinsic dimension (i.e., a two-dimensional surface) of the point set, which is typically lower than the dimension of the embedding space (i.e., three in the case of scanned surfaces

in three-dimensional space). An advantage to working in the lower dimension is that our segmentation process is efficient.
2. *Hierarchical segmentation.* We bisect the set of the supernodes while ensuring that supernodes belonging to a common feature remain together. A near-optimal bisection is computed using a spectral analysis of a weighted graph that represents the relation between supernodes. This second phase can be applied directly to the input points, but computing a near-optimal bisection is significantly faster when it is applied to the smaller set of supernodes. Repeated application of this bisection results in a hierarchical segmentation of the supernodes.
3. *Surface segmentation.* We construct a segmentation of the input points from the segmentation of the supernodes. Previous phases ensure that features lie in individual segments.

We now summarize the properties and shortcomings of the segmentation process using the criteria proposed by Attene et al. [40]. Some of the relevant properties are:

- *Type of segmentation.* A segmentation of a point-sampled surface model is computed to extract elongated features without the explicit construction of a mesh or other surface representation.
- *Hierarchy.* A hierarchical segmentation of supernodes supports multiple views of the input surface at various levels of detail.
- *Sensitivity to pose.* Segmentation results are independent of the poses of surface models since the segmentation process is based on geodesic distances and uniform sampling of points.

Shortcomings of the method described above include:

- *Extracting correct segments.* The method failed to identify significant features that were captured by a single supernode in some models.
- *Segment boundaries.* Leakage of segments beyond the feature boundaries was observed.
- *Control parameters.* The quality of segmentation depended strongly on the number of sample points. To obtain a good segmentation, this parameter needed to be tuned for each model. A large number of sample points were required for some models.
- *Asymptotic complexity.* The multiphase segmentation process works directly on the point primitives that represent a surface. This can lead to an efficient use of storage and computing resources. Specifically, the memory complexity is $O(n)$, and the run time complexity is $O(un\log(n))$, where u and n are the numbers of sample and input points, respectively. Unfortunately, for some models, a large number of sample points, u, may be needed, leading to a slow segmentation process.

In this paper, we describe several algorithmic improvements to address the shortcomings of the previous segmentation process. We list below our contributions that lead to significant improvements:

- *Extracting correct segments.* We describe a new weighted graph of supernodes that captures the connectivity within each supernode. This leads to a significant improvement in the quality of segmentation, where all elongated features can be extracted. An extension of the method to extract ridge-separated features is also discussed.
- *Segment boundaries.* We show that the growth of supernodes beyond their feature boundaries can be avoided by creating supernodes at saddle points and using local refinement techniques. Since the leakage of segments beyond feature boundaries was greatly reduced, a point-wise refinement technique [41,42] can be applied in a post-processing phase to obtain desirable geometric properties for the segment boundaries.
- *Control parameters.* Even though additional control parameters must be used, the user may have to adjust only two parameters to improve the quality of the segmentation for each model. The remaining control parameters are pre-determined to ensure high segmentation quality.
- *Asymptotic complexity.* The new algorithm is much less affected by the presence of noise in the input and requires a significantly smaller number of sample points. As a result, even though the asymptotic complexity of the new algorithm is the same as that of the previous algorithm, the time to compute the segmentation is greatly reduced.

In the following sections, we discuss all phases of the segmentation process in detail. In particular, we describe the shortcomings of the previous segmentation process, and the proposed algorithmic improvements. We also provide segmentation results of various point-sampled surfaces to demonstrate that the new algorithm greatly improves the quality of segmentation results, and reduces the time to compute the segmentation by a factor of up to five.

4 Supernode extraction

In the first phase of the segmentation process, we use ideas from *Morse Theory* to identify features in the underlying surface of an input point set. Morse Theory was originally developed to study the relationship between the shape of a space and critical points of smooth functions defined over the space [43,44]. Recently, it has been used to construct multi-resolution structures for the visualization of scalar data [45–47] and to remove noise from 2-manifolds [48]. In contrast to these approaches based on Morse Theory for smooth functions, we construct and analyze characteristics of a discrete function defined over the input point set.

4.1 Feature-identifying function

We construct the discrete function over the input points based on the concept of *centrality*, which was first introduced in the context of social networks to identify central people for transporting information within a network [49, 50]. The notion of centrality was used more recently by Hilaga et al. [51] in the context of shape matching to capture the skeletal and topological structures of 3D shapes, where the centrality of a point is defined as the average *geodesic distance* from the point to all points over the surface model.

Assuming that a sufficiently dense point-sampled surface is provided, the geodesic distance between two points on the surface can be approximated by their shortest path in a graph that connects all k-nearest points [6]. For example, we assume that our point set is dense enough for the k-nearest neighbor graph to identify two fingers of a hand model (Fig. 1), which are close to each other. This also implies that points that are associated with the same feature belong to a connected component of the graph. The disconnected components that belong to different features are identified in our hierarchical segmentation phase (Section 5). Based on this assumption, the centrality $\bar{f}(p)$ of a point p can be approximated as

$$\bar{f}(p) \approx \frac{1}{|P|} \sum_{q \in P} g(p,q),$$

where $|P|$ is the number of points in the input set P (i.e., $|P| = n$), $g(p,q)$ measures the shortest path between two points p and q in the k-nearest neighbor graph G, and if p and q are k-nearest neighbors of each other in G, then $g(p,q)$ is equal to their Euclidean distance $d(p,q)$.

To avoid the expensive computation of all-pair shortest path distances, we compute an approximate centrality value $f(p)$ as the average shortest path distance from the point p to uniformly distributed sample points U over the graph G, i.e.,

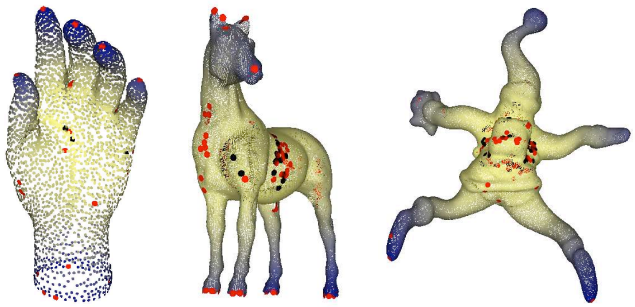
$$f(p) = \frac{1}{|U|} \sum_{q \in U} g(p,q), \tag{1}$$

where $|U|$ is the number of the sample points. Clearly, as $|U|$ approaches $|P|$, $f(p)$ approaches $\bar{f}(p)$.

In order to sample the points U uniformly over the graph G, we repeatedly sample a point p in the input set P that is furthest away from the points that are already in U. Specifically, for each point p in P, we first compute $h(p)$, which is the shortest path distance between the point p and its nearest point q already in U, i.e.,

$$h(p) = \min_{q \in U} g(p,q).$$

Then, we repeatedly sample a point p with the largest value $h(p)$ in the input set P.



hand$_c$ horse$_c$ santa$_c$

Fig. 1 Discrete function measuring approximate centrality values of points. Bluer colors correspond to larger function values. Red dots are the local maxima of the function and black dots are the local minima. Elongated features such as fingers, arms, and legs are represented by the local maxima that are located at the extremal points on the surface. Insignificant local maxima that do not represent any features are removed by a local refinement process (Section 4.3). For each model, we have a coarse and fine point set, which we denote with the subscripts c and f, respectively. Our segmentation approach obtains similar results for both sets.

To summarize, given the k-nearest neighbor graph G of the input point set P, we construct the discrete function f over P as follows[1]:

1. Initialize $h(p) = \infty$ and $f(p) = 0$ for all points $p \in P$.
2. Create a set U to store uniformly-sampled points from P, and initialize $U = \{\}$.[2]
3. Repeat steps 4-6 until $|U|$ becomes greater than $\sqrt{n}$.
4. Pick a point $p \in P$ with the largest value $h(p)$, breaking ties arbitrarily.
5. For all points $q \in P$,
   (a) compute the shortest path distance $g(p,q)$ in G,
   (b) update $f(q) \leftarrow f(q) + g(p,q)$, and
   (c) update $h(q) \leftarrow \min\{h(q), g(p,q)\}$.
6. Update $U \leftarrow U \cup \{p\}$.
7. Compute the average geodesic distance $f(p) \leftarrow f(p)/|U|$ for all $p \in P$.

After the completion of the above steps, $f(p)$ contains a value that approximates the centrality $\bar{f}(p)$. Fig. 1 shows the distribution of f for different surface models. We observed that the tip of an elongated feature is represented by a local maximum of f. Since our method computes a uniform sample of the input and approximate geodesic distance, our final segmentation results are sensitive to neither the uniformity of the input point distribution nor the pose of the surface model, which will be demonstrated later in Fig. 7.

We note that not all local maximum represent features in the underlying surface; in the presence of noise in the input, some local maximum may not represent a feature at all. The previous segmentation process [7]

[1] In our numerical experiments, k-nearest neighbors are computed using a kd-tree [52].

[2] The implementation maintains only the number of sample points.

is sensitive to the noise, and the quality of segmentation depends strongly on the number of sample points. In some models, a large number of sample points are needed, resulting in a slow segmentation process. The algorithmic improvements described in the rest of this paper make the segmentation process much less affected by the presence of noise in the input, reducing the number of required sampled points. As a result, the time to compute the segmentation is reduced by a factor of up to five (Section 6). We found that $\sqrt{n}$ sample points are sufficient to identify distinct features in all the surface models used in this paper.

4.2 Supernode extraction

We define *discrete gradient* $\widehat{\nabla} f(p)$ at a point p to be the steepest ascent of the function f from p to its k-nearest neighbor. Specifically, let $\hat{u}(p,q)$ be the unit vector pointing from p to its neighbor q,

$$\hat{u}(p,q) = \frac{q-p}{\|q-p\|},$$

and $z(p,q)$ be the *discrete gradient magnitude* given by

$$z(p,q) = \frac{f(q)-f(p)}{d(p,q)},$$

where the function f is given by (1), and $d(p,q)$ measures the Euclidean distance between the points p and q. Then, the discrete gradient $\widehat{\nabla} f(p)$ is given by

$$\widehat{\nabla} f(p) = z(p,q) \cdot \hat{u}(p,q),$$

where

$$q = \underset{q \in N(p)}{\arg\max}\ z(p,q),$$

and $N(p)$ is the set of the k-nearest neighbors of p. Each input point is then assigned to a local maximum of f by following the gradient flow field induced by $\widehat{\nabla} f$. Points assigned to a common local maximum collectively form a supernode. However, two shortcomings of the earlier segmentation process [7] must be resolved to ensure that the points in a supernode are associated with a common feature.

First, the discrete gradient field induced by $\widehat{\nabla} f(p)$ flows to a local maximum of f all the way from a local minimum of f. As a result, supernodes grow across the boundaries of their corresponding features (see Fig. 2). Note that these feature boundaries are identified by *saddle points*, where multiple supernodes merge at the largest centrality value. In order to prevent the growth of supernodes beyond their feature boundaries, we create a new supernode at a saddle point. All points lying within supernodes that merge at the same saddle point and whose approximate centrality values are smaller than that of the saddle point are assigned to the new supernode. The process of creating new supernodes is recursively applied to the newly-created supernodes. Fig. 3 shows the expanded set of supernodes.

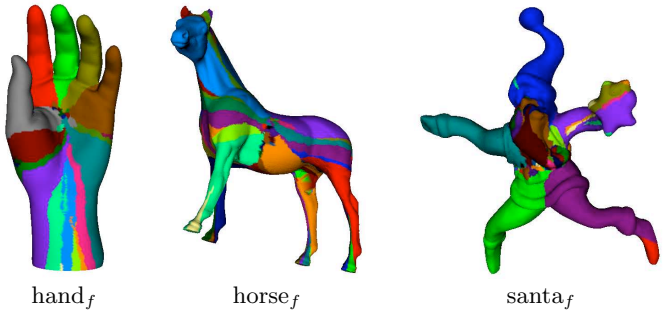


Fig. 2 Discrete gradient fields in surface models: the points flowing into a common local maximum collectively form a supernode. The surface meshes are added solely for the purpose of clearer illustration in the above and subsequent figures. The surface region formed by points in a supernode was generated by a simple mesh viewer. The surface region formed by the points in a supernode was generated using a simple mesh viewer for clearer illustration, and is shown in the same color. Since the gradient fields flow to a local maximum all the way from a local minimum, supernodes grow across the boundaries of distinct features. This is one of the disadvantages of our earlier work [7].

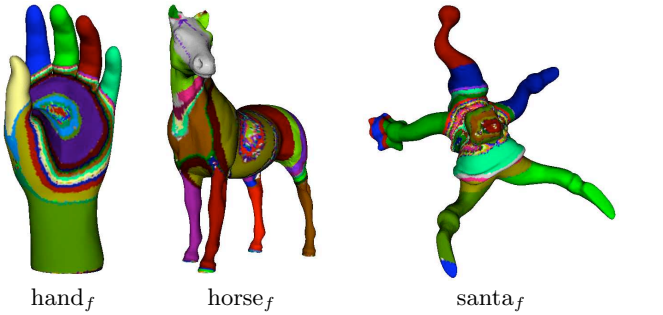


Fig. 3 New supernodes are created at saddle points. Growth of supernodes beyond the feature boundaries, as seen in Fig. 2, are avoided.

Second, in the k-nearest neighbor graph G, two points that lie within different features may be connected by *feature-crossing edges* even though their centrality values are far greater than those of the points near the corresponding feature boundaries. Note that these feature-crossing edges may exist in G even when the input points are dense enough to identify distinct features in the first phase of our segmentation process. These feature-crossing edges result in the creation of a saddle point and a new supernode instead of the continued propagation of two supernodes. Fig. 4 shows an example in which the feature-crossing edges cause an early termination of supernodes that otherwise would have identified the ring finger and middle finger. We observed that even if a point p is connected to points in different features by feature-crossing edges, the number of such edges is a small fraction of the total number of edges connecting the point p, i.e., most of the edges connect to the points within the same feature. Otherwise, the point p lies on its corresponding feature boundaries and should be identified as a saddle. There-

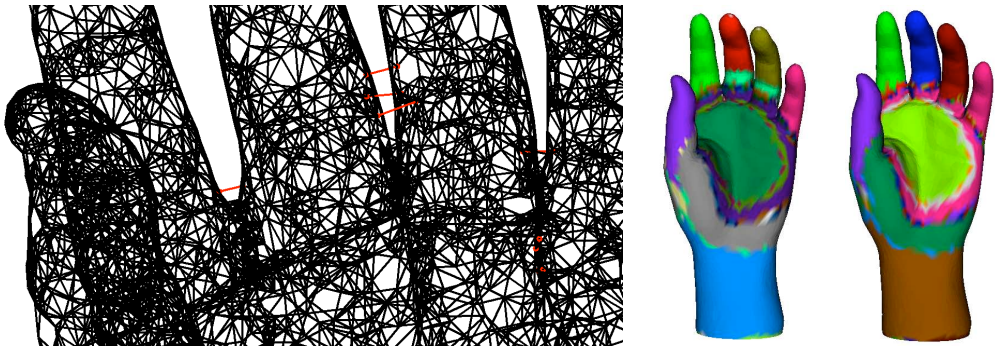


Fig. 4 Effect of feature-crossing edges. In the left figure, the edges in red are identified as the feature-crossing edges in the k-nearest neighbor graph of the coarse model of the hand. The quality of supernodes is improved by removing these edges as seen in the middle and right figures, which show the supernodes before and after feature-crossing edges are removed, respectively.

fore, in order to identify the feature-crossing edges and to efficiently extract correct surface features, we process points p in the input set P in descending order of $f(p)$ as follows:

1. If p is a local maximum, assign p to a new supernode.
2. Otherwise, assign p to a supernode using the following steps:
   (a) Let $M(p)$ contain all neighbors $N(p)$ of p that have been processed. In other words, if $q \in M(p)$, then $f(q) \geq f(p)$.
   (b) If the number of points in $M(p)$ from a supernode s_i is only a small fraction[3] of the total number of points in $N(p)$, then delete all points assigned to s_i from $M(p)$.
   (c) If the remaining points in $M(p)$ belong to a single supernode, then assign p to the supernode.
   (d) If the remaining points in $M(p)$ belong to multiple supernodes $s_1, \ldots, s_k$, then declare p a saddle point and assign p to a new supernode s. Any point processed in the future and assigned to one of the supernodes $s_1, \ldots, s_k$ will be assigned to s instead.

The above procedure successfully identifies a majority of the feature-crossing edges and dramatically improves the quality of the supernodes from the previous segmentation process. Fig. 5 shows the resulting supernodes.

4.3 Local refinement

Some of the supernodes computed above do not represent significant features in the underlying surface. To remove these insignificant supernodes, in this section, we describe a local refinement phase which was absent in the previous multiphase segmentation process. This additional phase not only improves the quality of the segmentation, but also reduces the segmentation time by reducing the number of supernodes.

[3] We use a threshold fraction of 0.3.

We first characterize each supernode s using three quantities:

1. The feature height $f_h(s)$, equal to the maximum difference in the approximate centrality values f between all pairs of the points in s, i.e.,
$$f_h(s) = \max_{p,q \in s} |f(p) - f(q)|. \tag{2}$$
2. The feature area $f_a(s)$, defined as
$$f_a(s) = \sum_{p \in s} \pi r(p)^2,$$
where the radius $r(p)$ is computed as
$$r(p) = \sum_{q \in N(p)} \frac{\|p - q\|_p}{|N(p)|},$$
and
$$\|p - q\|_2 = \sqrt{(f(p) - f(q))^2 + d(p,q)^2}. \tag{3}$$
3. The feature width $f_w(s)$, defined as
$$f_w(s) = \frac{f_a(s)}{f_h(s)}. \tag{4}$$

We observed that insignificant supernodes have either small feature areas or small feature heights; hence we merge them with their neighboring supernodes in two steps:

1. *Small supernodes are merged-up.* If the feature area $f_a(s)$ is less than a user-specified threshold[4] and the supernode s is created at a saddle point, then merge s back with supernodes that terminate at the saddle.
2. *Skinny supernodes are merged-down.* If the feature height $f_h(s)$ is less than a user-specified threshold[5] and the supernode s meets with another supernode at a saddle point, then merge s with the supernode created at the saddle point. Also, merge-down any small supernodes that have not been merged-up.

[4] Our threshold is 1% of the total feature area.
[5] Our threshold is 1% of the maximum feature height.

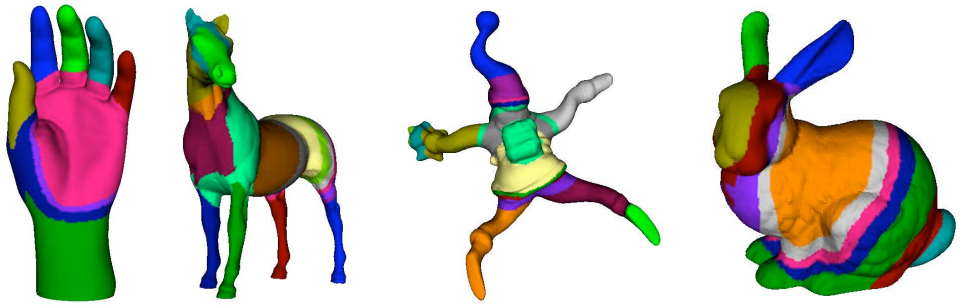


Fig. 5 Supernodes after local refinement. Insignificant supernodes are merged into connected supernodes.

We note that a supernode created at a saddle point can have disconnected regions. If a supernode contains disconnected regions even after the above local refinement steps, each of the disconnected regions is tested against the above refinement criteria, and then it is either merged with a neighboring supernode or becomes a supernode. Furthermore, after the feature-crossing edges are processed as described in Section 4.2, a small or skinny supernode may not be connected to any other supernode through a saddle point. In such a case, we merge these insignificant supernodes into a supernode that is connected point-wise in the k-nearest neighbor graph. Fig. 5 shows the supernodes after the local refinement steps.

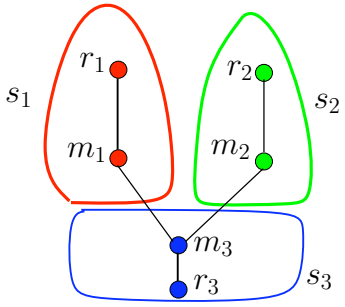


Fig. 6 Weighted graph representing the relation of supernodes. Two vertices are created for each supernode s_i: a representative vertex r_i and a member vertex m_i. There are edges between r_i and m_i, and between m_i and m_j if and only if s_i is a supernode created at a saddle point and s_j is one of the supernodes that terminates at the saddle. In the figure, s_3 is created at a saddle, where s_1 and s_2 merge.

5 Hierarchical segmentation

In this section, we first construct a weighted graph $G_s(V, E, W)$ based on the supernodes identified in Section 4.2; namely, each vertex in the vertex set V is a collection of input points that belong to a common feature and the weight $W(v_1, v_2)$ is large if two vertices v_1 and v_2 in V lie within similar features. We then present an algorithm to bisect the set of vertices V into two disjoint subsets V_1 and V_2 by computing a graph cut that minimizes the normalized cut value

$$NCut(V_1, V_2) = \frac{asso(V_1, V_2)}{asso(V_1, V)} + \frac{asso(V_1, V_2)}{asso(V_2, V)}, \tag{5}$$

where the association between the subsets V_1 and V_2 is given by

$$asso(V_1, V_2) = \sum_{v_1 \in V_i,\ v_2 \in V_2} W(v_1, v_2). \tag{6}$$

Minimizing the value $NCut$ results in a bisection in which vertices within a subset are similar, while those in different subsets are dissimilar. Thus, we avoid the bias toward small segments that are often favored when the cut is minimized without normalization. We recursively apply the bisection until $NCut$ is greater than a specified threshold. This bisection can be applied directly to the input points, but computing a near-optimal bisection is significantly faster when it is applied to the smaller set of the supernodes.

5.1 Weighted graph construction

In the previous segmentation process [7], the weighted graph G_s is constructed such that the supernodes are the vertices in the graph. However, with this weighted graph, some significant features could not be segmented out. This is because a feature in a point-sampled surface may be represented by a single supernode s_i (e.g., a finger in the model of the hand in Fig. 5). In such a case, this feature cannot be segmented out by minimizing $NCut$ because if $V_1 = \{s_i\}$ and $V_2 = V \setminus \{s_i\}$, then $asso(V_1, V_2)$ is equal to $asso(V_1, V)$ and $NCut(V_1, V_2)$ becomes its maximum value, i.e., $NCut = 2$. To avoid this problem, we propose a new weighted graph G_s that captures the connectivity within each supernode. In this new graph G_s, the vertex set V contains two vertices corresponding to each supernode s_i: a representative vertex r_i and a member vertex m_i. We then create two types of edges in G_s: between r_i and m_i, and between m_i and m_j. There are edges between all pairs r_i and m_i, but an edge between two member vertices m_i and m_j is added

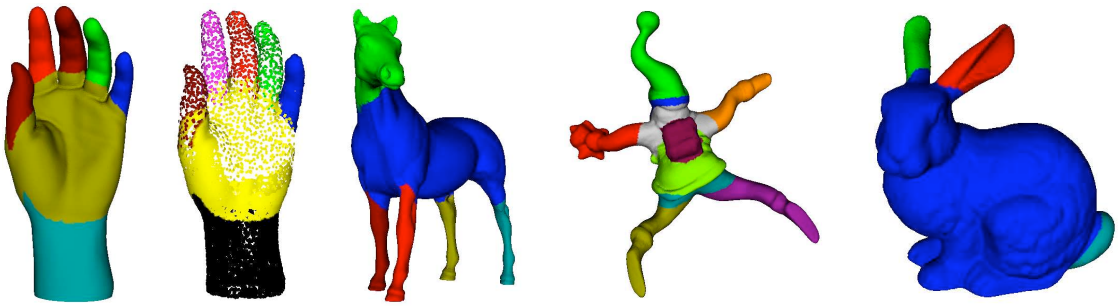


Fig. 7 Segmentation results after computing normalized cut for supernodes. The point distribution in hand$_n$ is skewed such that it is dense over the wrist and the little finger. This non-uniform distribution does not, however, affect the results.

to G_s if and only if s_i is a supernode created at a saddle point and s_j is one of the supernodes that terminates at the saddle. Fig. 6 shows an example of the graph G_s.

The weight of the edge between r_i and m_i measures the importance of the feature identified by s_i:

$$W(r_i, m_i) = e^{-\frac{1}{\beta f_w(s_i)}\left(1-\frac{conn(s_i,s_i)}{\max_{s_j\in V} conn(s_j,s_j)}\right)}, \tag{7}$$

where β is a user-specified scalar, the feature width $f_w(s_i)$ is given by (4), the connectivity between two supernodes s_i and s_j is defined as

$$conn(s_i, s_j) = \sum_{p\in s_i, q\in s_j\cap N(p)} \|p-q\|_2, \tag{8}$$

and the norm $\|\cdot\|_2$ is given by (3). Subsequently, even if a feature is represented by a single supernode, it can be extracted when its corresponding $W(r_i, m_i)$ is large.

The weight of an edge between m_i and m_j measures the similarity between the corresponding supernodes,

$$W(m_i, m_j) = e^{-\frac{d_1(m_i,m_j)}{\alpha_1}-\frac{d_2(m_i,m_j)}{\alpha_{ij}}}, \tag{9}$$

where α_1 and α_{ij} are scalars,

$$d_1(m_i, m_j) = \frac{|f_h(s_i) - f_h(s_j)|}{\max_{s_1,s_2\in V}|f_h(s_1) - f_h(s_2)|},$$

$$d_2(m_i, m_j) = 1 - \frac{conn(s_i, s_j)}{\max_{s_1,s_2\in V} conn(s_1, s_2)},$$

the feature height $f_h(s_1)$ is given by (2), and the connectivity $conn(s_i, s_j)$ is given by (8). The first component d_1 measures the relative significance of the supernodes. When s_i and s_j have similar feature heights, i.e., $f_h(s_i) \approx f_h(s_j)$, then $W(m_i, m_j)$ is large, which, in turn, facilitates the merging of the two supernodes. The second component d_2 measures how closely the two supernodes are connected to each other. The value for α_{ij} is chosen to be proportional to the ratio of the feature widths of the corresponding supernodes, namely

$$\alpha_{ij} = \alpha_2 \frac{\min\{f_w(s_i), f_w(s_j)\}}{\max\{f_w(s_i), f_w(s_j)\}},$$

so that small yet significant supernodes in the neighborhood of a wide supernode maintain their identity. The values of α_1 and α_2 are specified by a user. We note that, in the previous segmentation process, the similarity between the supernodes is measured simply by their approximated geodesic distance $g(s_i, s_j)$. With this simple similarity measure, a skinny feature may be connected to a large supernode with a small weight, and may be segmented out (e.g., the skinny features around the palm of the hand model in Fig. 5). The new similarity measure ensures that these supernodes have a small d_2 and are connected with a large weight, hence ensuring that they merge together.

5.2 Graph cut

Shi and Malik [19] showed that an approximate solution to minimize $NCut$ can be computed based on a spectral analysis of the *Laplacian matrix* $L = D - W$, where W is a matrix storing the edge weights in the graph G_s, and D is a diagonal matrix whose ith diagonal entry d_{ii} is given as

$$d_{ii} = \sum_{j\neq i} W(v_i, v_j).$$

Specifically, if y is the eigenvector corresponding to the second smallest eigenvalue λ of the generalized eigenvalue problem,

$$(D - W)y = \lambda D y, \tag{10}$$

then $NCut$ is approximately minimized by clustering all points p_i with approximately the same values y_i into a common subset:

$$v_i \in \begin{cases} V_1 & \text{if } y_i < \gamma, \\ V_2 & \text{otherwise.} \end{cases}$$

We identify the split value γ that minimizes $NCut$ from uniform samples of values that range between the smallest and largest elements in the eigenvector y. The recursive bisection of each subset results in a hierarchical segmentation of the vertex set V. The recursion terminates when $NCut$ is greater than a specified threshold.

Dataset	Data size n	m	l	t	Run time (sec) kNN	Cen	Snod	LRef	HSeg	PSeg	Total	Dey et al.
hand_c	4,000	39	12	7	0.00	0.09	0.02	0.20	0.01	0.00	0.32	2.23
horse_c	4,002	59	18	5	0.02	0.08	0.03	0.15	0.02	0.01	0.31	2.86
bunny_c	4,088	41	13	5	0.00	0.01	0.03	0.18	0.01	0.01	0.24	2.64
santa_c	5,002	55	15	9	0.01	0.12	0.03	0.20	0.03	0.00	0.39	2.64
hand_f	30,000	101	16	7	0.15	2.38	0.24	2.47	0.03	0.00	5.27	26.17
bunny_f	34,834	89	18	4	0.19	3.95	0.29	1.70	0.01	0.01	6.15	37.27
horse_f	40,002	237	19	6	0.17	3.96	0.66	12.30	0.01	0.01	17.11	42.18
santa_f	50,002	183	19	9	0.25	7.61	0.58	11.28	0.04	0.01	19.77	60.48
cube	9,602	21	6	6	0.20	0.39	0.04	0.20	0.02	0.01	0.68	−−
prism	28,674	146	8	8	0.14	3.09	0.31	2.48	0.04	0.01	6.07	−−
block	180,926	423	11	8	0.25	85.41	6.99	101.63	0.93	0.03	195.24	−−

Table 1 Performance data for each step of the segmentation process: kNN: k-nearest neighbor computation, Cen: approximate centrality computation, Snod: supernode identification, LRef: local refinement of supernodes, Hseg: similarity measure computation and hierarchical segmentation, and Pseg: construction of surface segmentation. Here, n is the number of input points, m is the number of supernodes, l is the number of supernodes after refinement, and t is the number of segments after hierarchical segmentation. For the segmentation of CAD models, kNN includes the computation of approximate normals and model transformation. k-nearest neighbors are computed using a kd-tree [52]. The first three steps as well as the last step, all of which work with point primitives, were implemented in C. All other steps were implemented in MATLAB. The last column of the table shows the time required by Dey et al.'s method [38] to compute segmentations of similar quality using default parameters. We do not provide the timing results of their method for the CAD models since it was not possible to compute a correct segmentation. We used a laptop PC with a 1.7GHz Intel Pentium M processor and 1GB RAM for our experiments.

Fig. 7 shows segmentation results obtained using the repeated application of the normalized cut. The segmentation results are similar to those obtained by the method of Dey et al. [38] shown in Fig. 9. For the models of the hand and santa, we used the choice of the scalars $\beta = 2.0$, $\alpha_1 = 2.0$, and $\alpha_2 = 0.5$. For the models of the horse and bunny, we found that decreasing the value of α_2 to 0.2 results in better segmentations. This can be explained by the fact that the models of the horse and bunny have features (i.e., the head and tail, respectively), which are not as elongated as the other features.[6] The thresholds used for the normalized cut were 0.07, 0.05, 0.23, and 1.2 for the models of the hand, horse, santa, and bunny, respectively. These two parameters (i.e., α_2 and the normalized cut threshold) are the only parameters that had to be adjusted.

6 Analysis

Memory requirement plays a crucial role in determining the efficiency of a segmentation method when the input point set is large. Our method requires the memory for storing Euclidean and geodesic distances between points for the computation of the approximate centrality values f in (1). However, only distances between certain pairs of points need to be stored. For example, computing geodesic distances requires the Euclidean distances between the neighboring points only. Thus, the memory requirement to store the Euclidean distances is $\Theta(kn)$, where k is the number of nearest neighbors considered for each point, and n is the number of input points. Geodesic distances between a point and $\sqrt{n}$ sample points are used to compute the approximate centrality values. However, only those geodesic distances from the current sample point to the rest of the points need to be stored, which results in $\Theta(n)$ storage complexity.

The computational complexities of the various steps of our method are:

k-nearest neighbor computation:	$O(kn\log(n))$
Approximate centrality computation:	$O(n^{3/2}\log(n))$
Feature identification:	$O(kn)$
Local supernode refinement:	$O(kmn)$
Hierarchical segmentation:	$O(tl^2)$
Construction of surface segmentation:	$O(n)$

Here, n is the number of input points, m is the number of supernodes, l is the number of supernodes after refinement, and t is the number of segments after hierarchical segmentation. Approximate centrality computation is clearly a computational bottleneck. However, our new algorithm uses $\sqrt{n}$ sample points, as opposed to the $c\sqrt{n}$ sample points used in our earlier one [7]. Since c can be as large as 60, the computational bottleneck of the segmentation process is, in practice, significantly reduced in the new method. Speed-ups of up to five were achieved for the models used in this paper. Table 1 summarizes our timing results. Local supernode refinement processes the input points, and it is currently implemented using simple arrays in MATLAB, which results in slow running time. The remaining computations process the set of supernodes, which is much smaller in size. As a result, our method is between two and eleven times faster than Dey

[6] It is possible for the representative vertex r_i and member vertex m_i of a same supernode s_i to be assigned to different segments. In this case, a simple local refinement can be applied to ensure that r_i belongs to the same segment as m_i. However in all of our experiments, we did not observe this event.

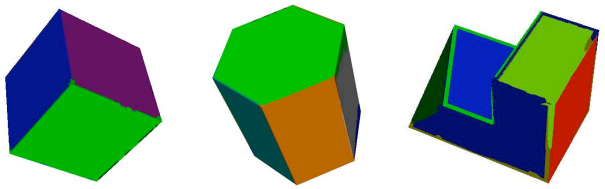


Fig. 8 Segmentation of CAD models. Edge lengths in the k-nearest neighbor graph and in the graph of supernodes are modified before applying the multiphase segmentation process.

et al.'s method [38] to compute segmentations of similar quality, as seen in the last column of Table 1 and Fig. 9.

7 Point-sampled CAD models

Our multiphase segmentation process can be applied to identify faces in point-sampled surfaces of CAD models. We incorporated two modifications to the segmentation process described in Sections 4 and 5: we modified edge lengths in the k-nearest neighbor graph G over the input points and those in the graph G_s representing the relation between supernodes.

Edge lengths in the graph G are modified to simulate a transformation of a CAD model that maps faces to elongated features and ridges to feature boundaries. Specifically, we shrink the lengths of edges connecting points that are equidistant from a ridge, especially the ones close to a ridge, and expand the lengths of remaining edges. This is done based on Laplacian smoothing of approximate normals of each input point. The new discrete function over the input points is constructed based on the centrality of the points in the transformed graph G. Supernodes that identify faces in the CAD model are extracted by applying the methods described in Section 4.

Following the ideas from Section 5.1, a weighted graph is constructed, where the vertex set contains two vertices representing each supernode. Similarity between supernodes is measured by comparing the orientation of the faces they represent. Supernodes that belong to a common face are identified by applying the normalized cut recursively to the weighted graph. Fig. 8 shows our segmentation results.

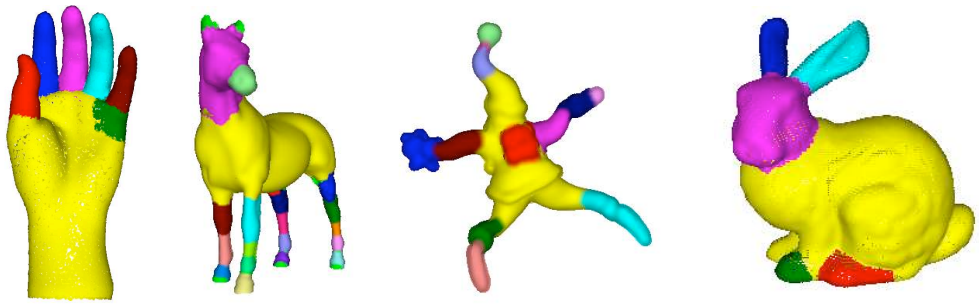
8 Conclusions

In this paper, we have described several algorithmic improvements for the multiphase segmentation process proposed in [7] to extract elongated features in a point-sampled surface without the explicit construction of a mesh or other surface representation. In comparison to the previous algorithm, both time efficiency and segmentation quality have been improved. The improvements in the quality of segmentation were achieved mainly by introducing the concept of saddle points of the discrete function defined over the input point set and by constructing a weighted graph of supernodes that better captures their relations. The time efficiency is improved primarily because the new algorithm is much less affected by the presence of noise in the input, which is typically the case for scanned surface models, reducing the number of sample points required to construct the discrete function. Since the leakage of segments beyond the feature boundaries was greatly reduced, point-wise refinement techniques [41,42] can be applied to obtain desirable geometric properties for segment boundaries. Some additional features of our segmentation process include the following: Surfaces with or without boundaries can be segmented correctly (e.g. the hand model has a boundary at the wrist). The segmentation results are independent of the poses of surface models because our methods are based on geodesic distances and uniform sampling of points. Even though there are several new control parameters, default values can be used for most of them to achieve good segmentation results. For all the surface models used in this paper, only two control parameters, namely the weights in the similarity measures and stopping threshold for the normalized cut, were adjusted. We also discussed an application of the segmentation process to identify ridge-separated features in point-sampled surfaces of CAD models.

Since we operate only on point primitives, all phases of the segmentation process can be applied to higher-dimensional data. Moreover, the embedding space is not restricted to being Euclidean. We merely require the points to be embedded in a metric space. Our method can potentially be used to segment point sets lying on a sub-manifold within a high-dimensional space in which each point is represented by a fixed-length feature vector [6]. Examples of such data sets include protein shapes [53] and hand-written characters [6]. It is therefore possible to extend our method to construct meaningful segmentations of such high-dimensional data sets.

Acknowledgments

The point sets used for our experiments were downloaded from on-line 3D scan repositories [54,55]. We used **qslim** [56] to generate coarse point sets. Yamazaki and Bai were supported in part by the National Science Foundation grants 0313390 and 0611548. Natarajan and Hamann were supported in part by the National Science Foundation grant under contracts ACI 9624034 (CAREER Award) and a large Information Technology Research (ITR) grant. Natarajan was also supported by a faculty startup grant from the Indian Institute of Science. We thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data



hand$_f$ horse$_f$ santa$_f$ bunny$_f$

Fig. 9 Segmentation results from Dey et al's approach. The quality of the segmentation is similar to that in Fig. 7, which was computed by our method. Our method computed the segmentation 2 to 11 times faster.

Analysis and Visualization (IDAV) at the University of California, Davis for helpful discussions.

References

1. M. Pauly, R. Keiser, L. P. Kobbelt, M. Gross, Shape modeling with point-sampled geometry, in: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, ACM Press, New York, NY, USA, 2003, pp. 641–650.
2. M. Zwicker, M. Pauly, O. Knoll, M. Gross, Pointshop 3D: an interactive system for point-based surface editing, in: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, New York, NY, USA, 2002, pp. 322–329.
3. H. Pfister, M. Gross, Point-based computer graphics, IEEE Computer Graphics and Applications 24 (4) (2004) 22–23.
4. M. H. Gross, Getting to the point...?, IEEE Computer Graphics and Applications 26 (5) (2006) 96–99.
5. M. Sainz, R. Pajarola, R. Lario, Points reloaded: Point-based rendering revisited, in: Proceedings Symposium on Point-Based Graphics, Eurographics Association, 2004, pp. 121–128.
6. J. B. Tenebaum, V. de Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 190 (5500) (2000) 2319–2323.
7. I. Yamazaki, V. Natarajan, Z. Bai, B. Hamann, Segmenting point sets, in: SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06), IEEE Computer Society, Washington, DC, USA, 2006, pp. 4–13.
8. T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, D. Dobkin, Modeling by example, in: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, ACM Press, New York, NY, USA, 2004, pp. 652–663.
9. A. Gregory, A. State, M. Lin, D. Manocha, M. Livingston, Interactive surface decomposition for polyhedral morphing, Vis. Comput. 15 (9) (1999) 453–470.
10. M. Zockler, D. Stalling, H.-C. Hege, Fast and intuitive generation of geometric shape transitions, Vis. Comput. 16 (5) (2004) 241–253.
11. Z. Karni, C. Gotsman, Spectral compression of mesh geometry, in: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000, pp. 279–286.
12. D. Cohen-Steiner, P. Alliez, M. Desbrun, Variational shape approximation, in: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, ACM Press, New York, NY, USA, 2004, pp. 905–914.
13. M. Attene, B. Falcidieno, M. Spagnuolo, Hierarchical mesh segmentation based on fitting primitives, Vis. Comput. 22 (3) (2006) 181–193.
14. E. Zuckerberger, A. Tal, S. Shlafman, Polyhedral surface decomposition with applications, Computer and Graphics 25 (5) (2002) 733–743.
15. X. Li, T. Toon, T. Tan, Z. Huang, Decomposing polygon meshes for interactive applications, in: I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, ACM Press, New York, NY, USA, 2001, pp. 35–42.
16. B. Lévy, S. Petitjean, N. Ray, J. Maillot, Least squares conformal maps for automatic texture atlas generation, ACM Trans. Graph. 21 (3) (2002) 362–371.
17. S. Biasotti, S. Marini, M. Mortara, G. Patané, An overview on properties and efficacy of topological skeletons in shape modelling, in: SMI '03: Proceedings of the Shape Modeling International 2003, IEEE Computer Society, Washington, DC, USA, 2003, p. 245.
18. S. Katz, A. Tal, Hierarchical mesh decomposition using fuzzy clustering and cuts, in: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, ACM Press, New York, NY, USA, 2003, pp. 954–961.
19. J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
20. K. Schloegel, G. Karypis, V. Kumar, Graph partitioning for high performance scientific simulations, in: Sourcebook of parallel computing, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003, pp. 491–541.
21. A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323.
22. V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag New York, Inc., New York, NY, USA, 1995.
23. A. Shamir, A formulation of boundary mesh segmentation, in: 3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium, IEEE Computer Society, Washington, DC, USA, 2004, pp. 82–89.
24. M. Garland, A. Willmott, P. S. Heckbert, Hierarchical face clustering on polygonal surfaces, in: I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, ACM Press, New York, NY, USA, 2001, pp. 49–58.
25. P. Sander, J. Snyder, S. Gortler, H. Hoppe, Texture mapping progressive meshes, in: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM Press, New York, NY, USA, 2001, pp. 409–416.
26. K. Zhou, J. Synder, B. Guo, H.-Y. Shum, Iso-charts: Stretch-driven mesh parameterization using spectral analysis, in: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM Press, New York, NY, USA, 2004, pp. 45–54.

27. S. Katz, G. Leifman, A. Tal, Mesh segmentation using feature points and core extraction, Vis. Comput. 21 (8–10) (2005) 649–658.
28. Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, H. P. Seidel, Intelligent mesh scissoring using 3D snakes, in: PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04), IEEE Computer Society, Washington, DC, USA, 2004, pp. 279–287.
29. R. Liu, H. Zhang, Segmentation of 3D meshes through spectral clustering, in: PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference (PG'04), IEEE Computer Society, Washington, DC, USA, 2004, pp. 298–305.
30. A. P. Mangan, R. T. Whitaker, Partitioning 3D surface meshes using watershed segmentation, IEEE Tran. Vis. Comput. Graph. 5 (4) (1999) 308–321.
31. G. Patane, M. Spagnuolo, B. Falcidieno, Para-Graph: Graph-based parameterization of triangle meshes with arbitrary genus, Computer Graphics Forum 23 (4) (2004) 783–797.
32. S. Shalfman, A. Tal, S. Katz, Metamorphosis of polyhedral surfaces using decomposition, Proc. Eurographics 21 (3) (2002) 219–228.
33. E. Zhang, K. Mischaikow, G. Turk, Feature-based surface parameterization and texture mapping, ACM Trans. Graph. 24 (1) (2005) 1–27.
34. Y. Zhou, Z. Huang, Decomposing polygon meshes by means of critical points, in: MMM '04: Proceedings of the 10th International Multimedia Modelling Conference, IEEE Computer Society, Washington, DC, USA, 2004, p. 187.
35. P. Sander, Z. Wood, S. Gortler, J. Snyder, H. Hoppe, Multi-chart geometry images, in: SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003, pp. 146–155.
36. H. Yamauchi, S. Lee, Y. Lee, Y. Ohtake, A. Belyaev, H. P. Seidel, Feature sensitive mesh segmentation with mean shift, in: SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI' 05), IEEE Computer Society, Washington, DC, USA, 2005, pp. 238—245.
37. D. L. Page, A. Koschan, M. A. Abidi, Perception-based 3D triangle mesh segmentation using fast marching watersheds, in: Proc. IEEE Conf. Computer Vision and Pattern Recognition, Vol. 2, 2003, pp. 27–32.
38. T. K. Dey, J. Giesen, S. Goswami, Shape segmentation and matching with flow discretization, in: Proc. Workshop on Algorithms and Data Structure, 2003, pp. 25–36.
39. C. Gotsman, On graph partitioning, spectral analysis, and digital mesh processing, in: SMI '03: Proceedings of the International Conference on Shape Modeling and Applications 2003 (SMI' 03), IEEE Computer Society, Washington, DC, USA, 2003, p. 165.
40. M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, A. Tal, Mesh segmentation - a comparative study, in: SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06), IEEE Computer Society, Washington, DC, USA, 2006, pp. 14–25.
41. C. M. Fiduccia, R. M. Mattheyses, A linear time heuristic for improving network partitions, in: DAC '82: Proceedings of the 19th conference on Design automation, IEEE Press, Piscataway, NJ, USA, 1982, pp. 175–181.
42. B. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell System Technical Journal (1970) 291–307.
43. Y. Matsumoto, An Introduction to Morse Theory, Amer. Math. Soc., 2002, translated from Japanese by K. Hudson and M. Saito.
44. J. Milnor, Morse Theory, Princeton University Press, Princeton, NJ, USA, 1963.
45. P. T. Bremer, H. Edelsbrunner, B. Hamann, V. Pascucci, A topological hierarchy for functions on triangulated surfaces, IEEE Transactions on Visualization and Computer Graphics 10 (4) (2004) 385–396.
46. A. Gyulassy, V. Natarajan, V. Pascucci, P. T. Bremer, B. Hamann, A topological approach to simplification of three-dimensional scalar fields, IEEE Transactions on Visualization and Computer Graphics 12 (4) (2006) 474–484.
47. V. Natarajan, V. Pascucci, Volumetric data analysis using Morse-Smale complexes, in: SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI' 05), IEEE Computer Society, Washington, DC, USA, 2005, pp. 322–327.
48. H. Edelsbrunner, D. Morozov, V. Pascucci, Persistence-sensitive simplification of functions on 2-manifolds, in: SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry, ACM Press, New York, NY, USA, 2006, pp. 127–134.
49. L. C. Freeman, Centrality in social networks: Conceptual classification, Social networks 1 (3) (1979) 215–239.
50. S. Wasserman, K. Faust, Social Network Analysis: Methods and Applications, Cambridge University Press, New York, NY, USA, 1994.
51. M. Hilaga, Y. Shinagawa, T. Komura, T. L. Kunii, Topology matching for fully automatic similarity estimation of 3D shapes, in: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM Press, New York, NY, USA, 2001, pp. 203–212.
52. D. M. Mount, S. Arya, ANN: A library for approximate nearest neighbor searching, http://www.cs.umd.edu/~mount/ANN/.
53. P. Roger, H. Bohr, A new family of global protein shape descriptors, ACM Computing Surveys 182 (2) (2003) 167–181.
54. AIM@SHAPE, http://www.aimatshape.net/.
55. Level of detail for 3D graphics, http://www.lodbook.com/models/.
56. M. Garland, QSlim simplification software, http://www.graphics.cs.uiuc.edu/~garland/software/qslim.html.