Segmenting Point Sets

Ichitaro Yamazaki* yamazaki@cs.ucdavis.edu
Vijay Natarajan† vijayn@ucdavis.edu
Zhaojun Bai* bai@cs.ucdavis.edu
Bernd Hamann†,* hamann@cs.ucdavis.edu

Institute for Data Analysis and Visualization (IDAV)†
and Department of Computer Science*
University of California, Davis, California 95616

Abstract

Extracting features from point sets is becoming increasingly important for purposes like model classification, matching, and exploration. We introduce a technique for segmenting a point-sampled surface into distinct features without explicit construction of a mesh or other surface representation. Our approach achieves computational efficiency through a three-phase segmentation process. The first phase of the process uses a topological approach to define features and coarsens the input, resulting in a set of supernodes, each one representing a collection of input points. A graph cut is employed in the second phase to bisect the set of supernodes. Similarity between supernodes is computed as a weighted combination of geodesic distances and connectivity. Repeated application of the graph cut results in a hierarchical segmentation of the point input. In the last phase, a segmentation of the original point set is constructed by refining the segmentation of the supernodes based on their associated feature sizes. We apply our segmentation algorithm on laser-scanned models to evaluate its ability to capture geometric features in complex data sets.

Keywords: *point sets, sampling, features, geodesic distance, normalized cut, spectral analysis, hierarchical segmentation.*

1. Introduction

Recent developments in scanning technologies have led to a substantial increase in the number of available surface models. *Features* in a surface model are its distinct parts that characterize the surface model. Examples of features are legs of a horse or fingers in a hand. We propose a new definition of features and describe an efficient three-phase segmentation process for partitioning a given point-sampled surface into distinct parts without explicit construction of a triangle mesh. In the first phase, we use the intrinsic dimension of the point sample, which is typically lower than the dimension of the embedding space, to identify sets of points that constitute distinct features. We adopt a topological approach to define features: a discrete function and its associated gradient flow field are constructed for the input point set. The discrete function measures the *centrality* of each point within the surface. A collection of points that flow into a common sink of this flow field constitute a feature. Each sink represents a feature and is called a *supernode*. In the second phase, we use spectral analysis to bisect a graph constructed over the supernodes. Repeated application of this bisection results in a hierarchical segmentation of the surface point set. This segmentation is refined in the third phase, where segments with insignificantly small features are merged into a neighboring segment with a larger feature. Our approach has the following advantages:

- *Efficiency.* Point primitives support simple and flexible modeling of complex shapes and are therefore increasingly being used to represent surfaces [25, 34]. We work directly with the point primitives used to represent the surface without explicit construction of a mesh. This approach leads to efficient usage of storage and computing resources to segment a point set.
- *Simple implementation.* Our program modules for all three phases of segmentation consist of a total of around 350 lines of code.
- *Hierarchy.* A hierarchical segmentation supports multiple views of the partition based on a desired level of detail.
- *Extension to higher dimensions and non-Euclidean spaces.* Since we operate only on point primitives, all three phases of the segmentation can be applied to higher-dimensional data. Also, the embed-

ding space is not restricted to be 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 [31].

2. Related Work

Current approaches to feature-based segmentation typically require that a surface model is provided explicitly via a triangle mesh [14, 18, 24, 32] or that a mesh is procedurally constructed [4].

Segmenting a model into its distinct parts and extracting its features are crucial for several applications like shape recognition, classification, matching, simplification, analysis, morphing, retrieval, and surface reconstruction [7, 9, 12, 16, 19, 28]. Numerous surface segmentation methods have been developed based on techniques from computer vision, load partitioning in finite element methods (FEM), point set clustering in statistics, and machine learning. These methods can be broadly classified into two categories based on their objective, namely patch-type segmentation and part-type segmentation [29]. Patch-type methods obtain segments that are topological disks whereas part-type methods partition a surface into segments that correspond to features.

Existing surface segmentation methods typically assume a surface to be represented by mesh. Various approaches have been used successfully for mesh segmentation: watershed segmentation simulates the accumulation of water into basins [19, 24]; spectral clustering methods analyze an affinity matrix that stores measures of similarity between all pairs of mesh faces in order to partition the mesh [10, 18]; mesh partitioning methods are a crucial ingredient in various load balancing algorithms, where the goal is to partition the input into independent sets that can be processed in parallel [27]; and point clustering methods are used for efficient organization of data with applications in information retrieval [15].

The method described by Dey et. al. [4] and the first phase of our method both initially identify local maxima of a discrete function. Though these local maxima constitute distinct features, they are computed differently. Dey et al. describe the discrete function on an explicitly computed three-dimensional mesh. We, on the other hand, define a discrete function over the point sample. Points are assigned to a feature by Dey et al. based on the flows induced within the mesh whereas we determine features based on gradient flow of the discrete function. Since we work in a lower dimension, namely on the surface, our approach computes a segmentation 5-7 times faster than their approach. Zhang et al. [33] propose a method to identify segments corresponding to local maxima of the average geodesic distance function, which is similar to our method. They are, however, interested in obtaining a patch-type segmentation for surface parameterization. A features is identified by growing a region based on the geodesic distance from a local maximum and searching for a feature boundary, which results in an abrupt increase in the surrounded region. The approach of Katz and Tal [16] also decomposes a mesh using representative faces for distinct features. These representative faces are chosen iteratively using a k-means algorithm. Faces are assigned to a feature based on the geodesic distance between the face and representative.

3. Three-phase Segmentation

Given a point-sampled surface as input, we want to partition the surface into distinct parts by associating each point with the feature that it belongs to. We propose a three-phase process to perform this segmentation:

1. *Feature identification.* In the first phase, we define features using a topological approach. This phase sets the stage for performing hierarchical segmentation in an efficient manner by coarsening the input into supernodes. Each supernode represents a collection of input points that constitute a feature.
2. *Hierarchical segmentation.* In the second phase, we bisect the set of supernodes while ensuring that similar supernodes remain together. Repeated application of this bisection step results in a hierarchical segmentation of the input point set. A near-optimal bisection is computed by using spectral analysis of a graph whose edge weights correspond to the similarity between the end point supernodes. The second phase of segmentation process can be applied directly to the input point set. However, computing a near-optimal bisection is significantly faster when applied on smaller sets of supernodes. Moreover, the first phase ensures that a feature lies within a single segment.
3. *Refinement.* In the third phase, we post-process the segmentation to ensure that all segments contain at least one significant feature. Small-scale features, that are present as individual segments, are merged with a neighboring segment.

In the following sections, we discuss these three phases in detail and provide results of our experiments for various laser scanned models.

4. Feature Identification

We use ideas from *Morse theory* to define features in the input. Morse theory was originally developed to study the relationship between the shape of a space and the critical points of smooth functions defined on the space [20, 21]. Recently, it has been successfully used to construct multi-resolution structures for the visualization of scalar field data [2, 11, 23]. Dey et al. adapted ideas from Morse theory for smooth functions to a discrete domain to segment 3D models [4]. The above applications of Morse theory require an explicit representation of the domain space, i.e., a triangle or tetrahedral mesh. We, on the other hand, work directly with point sets. We construct a discrete function f that measures the centrality of a point within the surface. The notion of centrality has been studied in the context of social networks [6] and more recently in the context of shape matching [13]. In our segmentation process, centrality of a point p is defined as the average geodesic distance from p to other points on the surface, and it measures the importance of the point to capture a feature in the surface. A shortest path in a graph G that connects every point to its k-nearest neighbors approximates the geodesic distance between two points on the surface when a sufficiently dense point-sampled surface is provided [31].

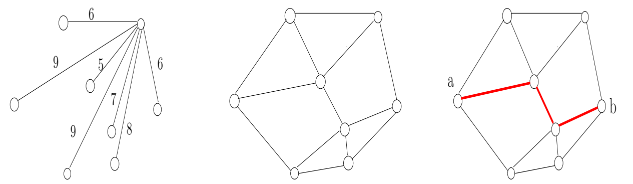

Figure 1. Geodesic distance computation. Distance between two points is given by the Euclidean metric in the embedded space (left). A graph connecting every point to its k-nearest neighbors for $k = 3$ (middle). Shortest path between two points a and b approximating the geodesic distance (right).

A *discrete gradient* of f at an input point p is defined as

$$\max \frac{|f(p) - f(q)|}{\|p - q\|_2}$$

among k-nearest neighbors q of p where $\|\|_2$ denotes the Euclidean distance between p and q. Thus, discrete gradient for an input point is defined by the edge in G that corresponds to the steepest ascent of function f. In our segmentation process, sinks of this discrete gradient flow field define distinct features, and they are called supernodes.

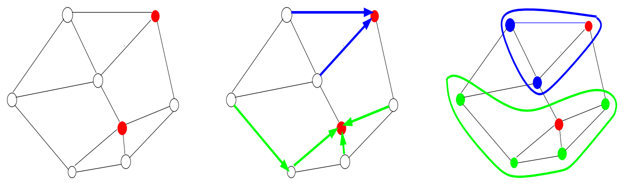

Figure 2. Features and their constituent points. Left: two local maxima of the discrete function; middle: edges with steepest gradient are followed from each point toward a local maximum; right: all points grouped into two sets based on their associated local maxima.

Given a point set P, the first phase of the segmentation approach proceeds as follows:

1. Store P in a kd-tree.
2. For each point $p \in P$, compute the k-nearest neighbors using a kd-tree [22].
3. Construct a weighted graph G over P whose edges are given by the k-nearest neighbors resulting from step 2. Set the weight of an edge to the Euclidean distance between its end points.
4. Compute a discrete function f at each point p as the average shortest distance from p to all points in G. Use Dijkstra's shortest path algorithm [3] to compute the distance between two points in G.
5. For each point $p \in P$, compute a discrete gradient flow by considering the edge incident on P along which f increases maximally. Declare p a sink if none of its neighbors has a higher function value.
6. Follow the gradient flow field from each point p toward a sink, and include p into the feature identified by its associated sink.

In our experiments, we use the value $k = 7$ to construct G. A higher value of k may be required if the points are embedded in a higher-dimensional space. Figure 1 shows the construction of G, for $k = 3$. Figure 3 shows the distribution of f for different surface models. Sinks of the discrete gradient flow field correspond to local maxima of f and are marked in red. Figure 2 illustrates the identification of features corresponding to the sinks, Figure 4 shows features extracted from various surface models. Sinks and their associated points capture distinct features like legs, fingers, ears, etc. Furthermore, the locations of sinks and features are invariant under rigid transformation of the input. Henceforth, each

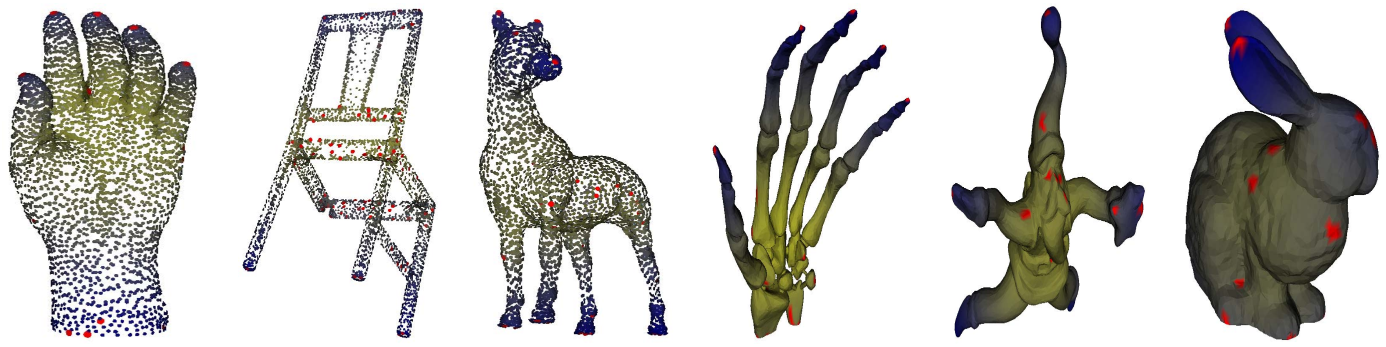

Figure 3. Discrete function measuring centrality of points. PointShop3D [34] was used to generate the three figures on the left. The three figure on the right were generated using a simple mesh viewer. A mesh was created only for the purpose of visualization. Bluer colors correspond to larger function values. The red dots are local maxima of the function. Local maxima are located at the extremal points and represent distinct features of surface models such as fingers, legs, and ears. Points further away from local maxima have smaller function values and contain less distinct features.

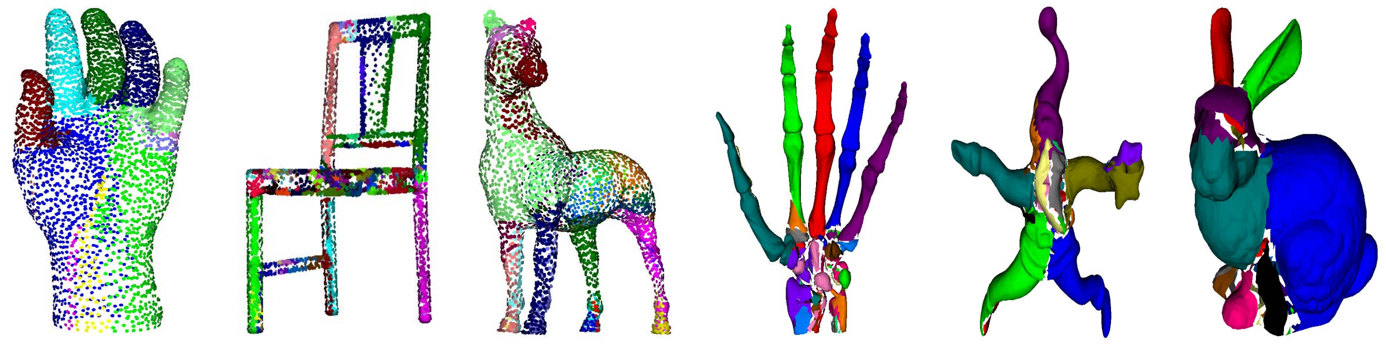

Figure 4. Features in surface models. Points with same color are collapsed to form a supernode while distinct features such as fingers, legs, and ears are captured. For the three models on the right, triangles between two supernodes have been deleted to emphasize the boundary between them.

feature is represented by its sink and called a supernode.

5. Hierarchical segmentation

In the second phase of our segmentation process, we partition the set of supernodes. We compute a partition by repeated application of a graph cut procedure to a weighted graph over the supernodes. All pairs of supernodes have an edge between them, but only those pairs that lie within the same connected component of G are connected by an edge with non-zero weights.

First, we compute the similarity between two supernodes, u and v, using their geodesic distance in the graph G. The weight W of the edge between u and v is defined as

$$W(u,v) = \begin{cases} e^{-d(u,v)} & \text{if } u \text{ and } v \text{ are connected} \\ & \text{ by a path in } G; \\ 0 & \text{otherwise,} \end{cases}$$

where $d(u,v)$ is the geodesic distance between u and v. Small weights are assigned to the edges that connect weakly related supernodes, and large weights are assigned to the edges that connect similar supernodes.

We use the similarity measure W to compute a segmentation such that supernodes within a segment are closely related, and supernodes in different segments are weakly related, i.e., the set of supernodes V is bisected into two disjoint sets V_1 and V_2 such that the normalized cut value

$$NCut(V_1, V_2) = \frac{cut(V_1, V_2)}{assoc(V_1, V)} + \frac{cut(V_1, V_2)}{assoc(V_2, V)},$$

is minimized. Here,

$$cut(V_1, V_2) = \sum_{v_1 \in V_1, v_2 \in V_2} W(v_1, v_2)$$

and

$$assoc(V_1, V) = \sum_{v_1 \in V_1, v \in V} W(v_1, v).$$

Minimizing $NCut$ maximizes the association within a segment while minimizing the cut between segments. Minimizing $NCut$ also avoids bias toward small segments, which are favored if the cut value is minimized without normalization.

Discrete minimization of $NCut$ is NP-complete. Shi and Malik introduced the idea of a normalized cut for image segmentation and showed that an approximate solution to the above minimization problem can be obtained by first solving the generalized eigenvalue system

$$(D - W)y = \lambda y,$$

where D is a diagonal matrix, whose ith diagonal entry d_{ii} is the degree of $v_i \in V$, i.e.,

$$d_{ii} = \sum_{v \in V \setminus v_i} W(v_i, v),$$

and the eigenvector y corresponds to the second smallest eigenvalue [30]. They also proved that $NCut(V_1, V_2)$ is minimized when

$$y_i = \begin{cases} a, & v_i \in V_1 \\ b, & v_i \in V_2. \end{cases}$$

For a near-optimal bisection, all points v_i that lie within a cluster have approximately same value y_i. We determine the split value α that identifies the near-optimal cut from a set of uniformly spaced values between the smallest and largest elements in the eigenvector. Then, we form two clusters of v_i that correspond to the near-optimal solution of the normalized cut:

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

We recursively bisect each subset to obtain a hierarchical segmentation. The recursion terminates when $NCut$ becomes greater than a specified threshold. Figure 5 shows segments of a hand for different levels of the hierarchy.

Solving a generalized eigenvalue system for all eigenvectors requires $\mathcal{O}(m^3)$ operations, where m is the number of supernodes. Since m is small, i.e., in the order of tens for our examples, the computational cost to solve the eigensystem is not significant. Furthermore, $NCut$ approximation requires only an estimate of the eigenvector associated with the second-smallest eigenvalue. High-precision computation of eigenvector elements is not required, and signs of the eigenvector elements are often sufficient to determine a good segmentation. We use the method proposed by Lanczos to compute an approximation of the eigenvector efficiently [17]. Figure 6 shows the eigenvector components of a hand model.

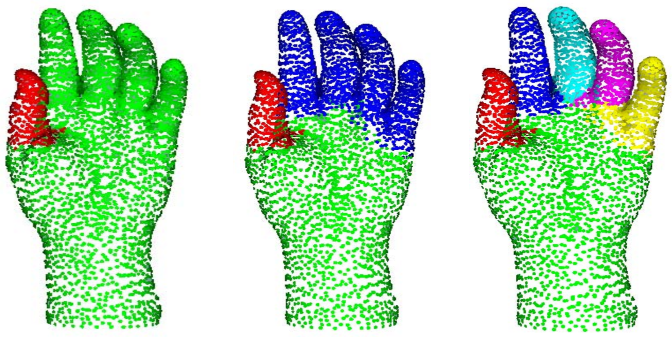

Figure 5. Three levels in the hierarchical segmentation of hand model. The figure on left shows the first bisection of hand that extracts the thumb. The middle figure shows the second bisection where the other fingers have been extracted from the palm. The figure on right shows the sixth and the last bisection where all fingers have been extracted.

6. Refinement

In the second phase of our segmentation process, a segmentation of supernodes is constructed based on the similarity of supernodes. However, significance of the corresponding features is not considered when constructing the segmentation. Some segments may not contain any significant features. In a segment that does not contain a significant feature, the maximum difference in values of f within the segment, called the *feature size*, is relatively small. In the third phase, the segmentation of supernodes is refined based on the feature sizes of segments so that each segment contains at least one significant feature. When the feature size of a segment is smaller than a specified threshold, it is merged with a neighboring segment that has a long boundary with it and has a large feature size.

After refinement, the segmentation of input points is constructed simply by replacing each supernode with

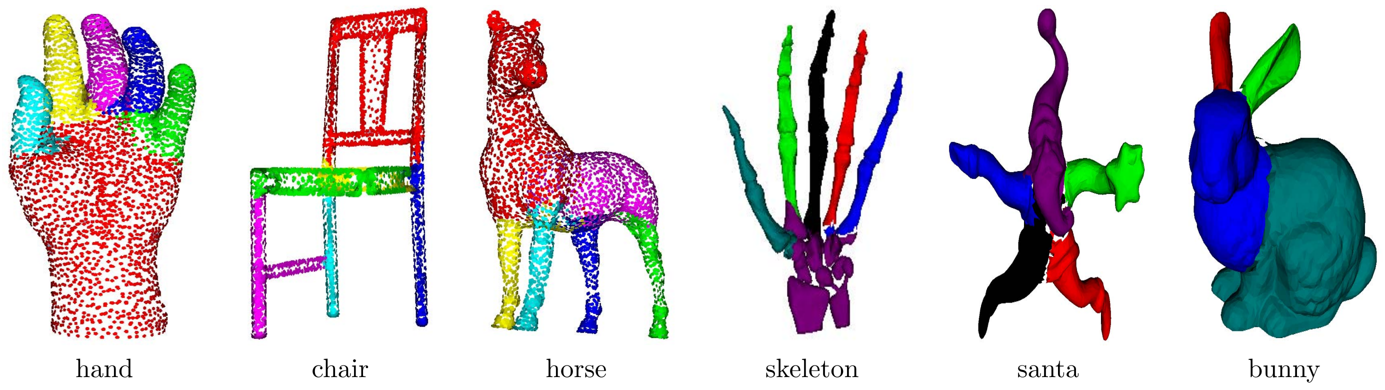

Figure 7. **Segmentation of point-sampled surfaces. Distinct parts of a model such as fingers, legs, and ears are identified without using an explicit surface representation.**

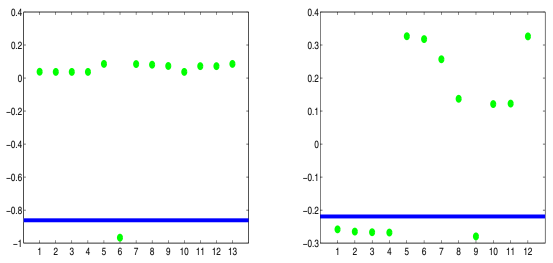

Figure 6. **Eigenvector analysis for hierarchical segmentation of hand model. The green dots represent the values of individual eigenvector components, and the blue line indicates the split value used for bisection. The left graph shows the eigenvector for the first bisection of the hand that segments out the thumb. The right graph shows the eigenvector for the second bisection that segments out the four fingers. All eigenvectors are computed to a required precision using a single Lanczos iteration, and eigenvector components have approximately piecewise constant values.**

points that constitute its associated feature. Figure 7 shows the results of our segmentation process.

7. Analysis and Optimization

In the following, we analyze storage costs and run time complexity of our algorithms and describe an approximation scheme that results in an order-of-magnitude improvement in run time behavior.

Memory requirement is an issue when the input data set is large. Our method uses Euclidean and geodesic distances between points. However, only distances between certain points need to be stored during the construction of segmentation. For example, Euclidean distances between near neighbors are used to compute the geodesic distances. Thus, the space requirement to store Euclidean distances is $\Theta(kn)$, where n is the number of input points, and k is the number of nearest neighbors considered for each point. Geodesic distance between two local maxima is used to compute similarity between the corresponding supernodes. Thus, only the geodesic distances between m local maxima need to be stored, resulting in $\Theta(m^2)$ storage complexity.

Table 1 summarizes our experimental results. Clearly, the geodesic distance computations required to evaluate centrality is a computational bottleneck. The computational complexities of the different steps are:

k-nearest neighbor computation:	$\mathcal{O}(kn\log(n))$
Centrality function computation:	$\mathcal{O}(n^2\log(n))$
Feature identification:	$\mathcal{O}(kn)$
Hierarchical segmentation:	$\mathcal{O}(lm^2)$
Refinement:	$\mathcal{O}(l^2)$.

Here, l is the number of segments resulting from the second phase. The number of Lanczos iterations required to obtain a desired precision for the second eigenvector is assumed to be independent of the input size.

It is possible to modify our segmentation process so that the computational bottleneck in the geodesic distance computation between all pairs of input points can be avoided while maintaining the quality of segmentation. This modification is based on the following steps:

1. Construct a weighted graph G over P whose edges are given by the k-nearest neighbors. Set the

	Data size				Run time (sec)					Optimized (sec)	
Dataset	n	m	l	t	kNN	Cen	Snode	HSeg	Ref	AppCen	Dij
hand (LoRes)	$4,000$	13	7	6	0.00	5.33	0.02	0.14	0.01	4.39	0.02
horse (LoRes)	$4,002$	32	6	6	0.02	5.15	0.01	0.72	0.02	0.01	0.04
bunny (LoRes)	$4,088$	25	4	4	0.00	5.65	0.01	0.27	0.02	0.01	0.04
santa (LoRes)	$5,002$	28	7	5	0.01	8.33	0.02	0.59	0.03	0.02	0.05
chair (LoRes)	$5,015$	99	9	6	0.07	8.13	0.06	2.19	0.06	0.12	0.18
skeleton (LoRes)	$5,992$	35	6	6	0.05	11.71	0.02	1.08	0.06	0.30	0.08
chair	$10,019$	143	7	6	0.08	34.71	0.12	2.01	0.25	0.35	0.43
hand	$30,000$	55	15	7	0.15	383.95	0.17	1.25	1.07	2.29	0.65
bunny	$34,834$	58	4	4	0.19	665.91	0.18	0.99	1.82	0.36	0.75
horse	$40,002$	130	8	7	0.17	709.77	0.32	3.75	2.30	74.21	2.23
skeleton	$49,991$	97	11	8	0.33	$1,161.87$	0.57	3.31	4.32	1.72	2.31
santa	$50,002$	89	5	5	0.25	$1,498.52$	0.61	1.57	3.75	41.13	3.21

Table 1. Performance data for each step of segmentation process. kNN: k-nearest neighbor computation; Cen: centrality function computation; Snode: supernode identification; Hseg: hierarchical segmentation; and Ref: refinement. The two right columns show the run time based on the optimization technique. App: approximate centrality computation including computation of approximate centrality and identification of local maxima in the graph; Dij: shortest-path computation between local maxima using Dijkstra's algorithm. These two steps in the optimization replace centrality function computation step in the segmentation process. The other steps in the segmentation process, i.e., computation of k-nearest neighbors, constructing the graph of supernodes, the hierarchical segmentation of supernodes, and refinement, take about the same amount of time, with or without optimization. Here, n is the number of input points, m is the number of supernodes, l is the number of segments after hierarchical segmentation, and t is the number of segments after refinement. k-nearest neighbors are computed using a kd-tree [22], and the first two phases, Cen and Snode, were implemented in C. All other algorithms were implemented in MATLAB. We used a laptop PC with a 1.7GHz processor and 1GB RAM for our experiments.

weight of an edge to the Euclidean distance between its end points.

2. Compute a discrete function $\widetilde{f}(p)$ at each point p that approximates the centrality $f(p)$.
3. Declare local maxima of $\widetilde{f}$ as supernodes.
4. Construct a graph of supernodes.
5. Compute shortest paths distance between all pairs of supernodes.
6. Compute a hierarchical segmentation of supernodes.
7. Refine the segmentation of supernodes and construct a segmentation of the input point set.

Since the number of supernodes is much less than the number of input points, the all-pair shortest path computation between supernodes is significantly faster than that between input points.

Similar to the approach proposed by Eppstein and Wang [5], we approximate the centrality of each point from a small sample of input points. However, instead of approximating centrality from a random sample, we sample points uniformly over the point set. We choose a sample point furthest from all points that have been sampled previously. Given a weighted graph G over P, the relevant steps to approximate the centrality are:

i. Compute shortest path distance $d(p,q)$ from a given point p to all points q in P.

ii. Create a set S of sample points and set $S = \{p\}$.

iii. Create two arrays g and $\widetilde{f}$, and set $g(q) = \widetilde{f}(q) = d(p,q)$ for all q in P.

iv. Pick a point r with largest $g(r)$

v. For all points q in P

   (a) compute shortest path distance $d(r,q)$ from r to q,

   (b) update $\widetilde{f}(q) = \widetilde{f}(q) + d(r,q)$, and

   (c) replace $g(q) = \min\{g(q), d(r,q)\}$.

vi. Update $S = S \cup \{r\}$

vii. Repeat Step 4 - 6 until the size of S becomes greater than a given threshold.

viii. $\widetilde{f}(q) = \widetilde{f}(q)/|S|$ for all q in P where $|S|$ is the size of S.

After the completion of the above steps, $\widetilde{f}(p)$ contains a value that is approximately equal to the centrality of p. We sample $\Theta(\sqrt{n})$ points, and therefore, the complexity of computing approximate centrality of all points becomes $\mathcal{O}(n\sqrt{n}\log(n))$.

Figure 8 shows that the samples are uniformly distributed over the input point sets and that the location of the local maxima of $\widetilde{f}$ are close to the location of local maxima of f. If centralities are normalized to have values between zero and one, the root mean squared errors are less than 0.08 for all point sets used in this paper. Figure 9 shows that there is no loss in the quality of segmentation due to the approximation. The right columns in Table 1 document the benefit of the optimization. The run time is reduced by an order-of-magnitude.

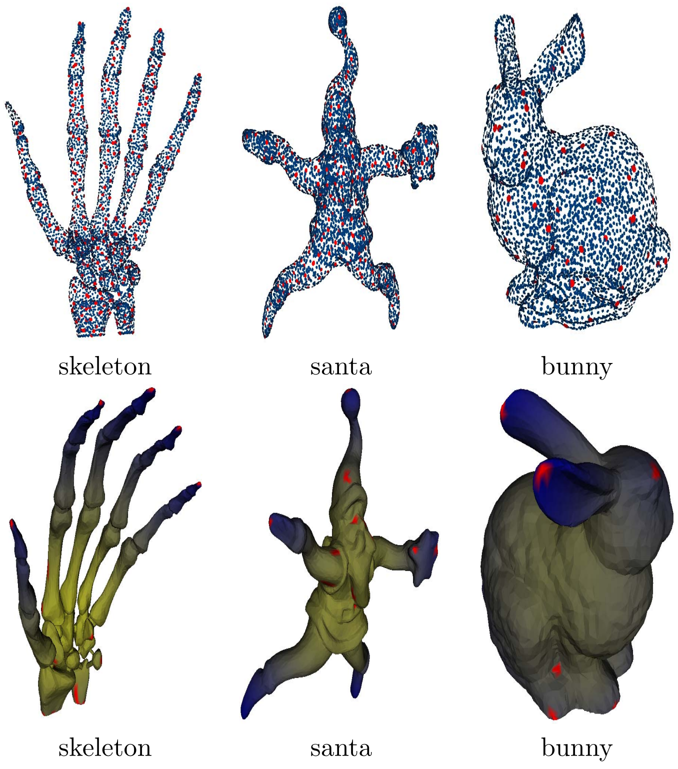

Figure 8. Top: uniform distribution of sample points (red) in input point sets (blue). Bottom: discrete function measuring approximate centrality. Blue regions have larger function values. The uniform distribution of sample points results in a good approximation of centrality. Maxima of the function (marked in red) represent distinct features similar to maxima of the function shown in Figure 3.

8. Future Work

We have introduced a technique for partitioning point-sampled surfaces into distinct features without explicit construction of a mesh. Since our method works directly with a point set, it can be extended to segment point sets in high-dimensional and non-Euclidean spaces, where each point represents a fixed-length feature vector. Examples are collection of protein shape analysis [26] and hand-written character recognition [31] represented as sets of points in high-dimensional space. Each dimension would correspond to a characteristic attribute of a protein or a hand-written character. We plan to extend our method to construct meaningful segmentations of such data sets.

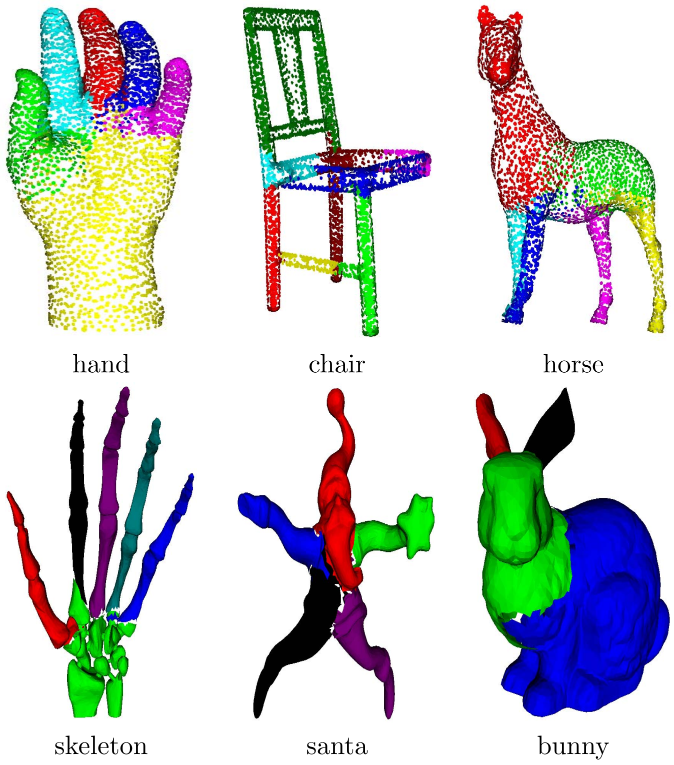

Figure 9. Segmentation of point-sampled surfaces using approximate centrality. Distinct parts are identified similarly to the results shown in Figure 7.

We are also considering an alternate approach to feature identification, using a new function $f_2(p)$ that measure the geodesic distance from p to its supernode which is the local maximum of f. This approach allows us to identify ridge-separated features in a point sampled surface such as a laser scan of a cube shown in Figure 10.

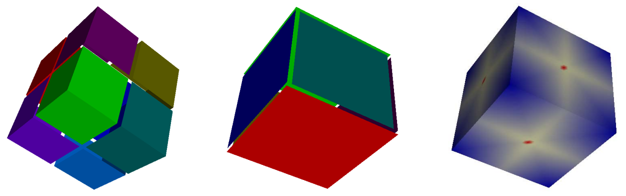

Figure 10. Results of two-step discrete function computation. Left figure shows features identified using the centrality measure f. The local maxima of f are located on the ridges of the input surface. Middle figure shows the features identified using a distance function f_2 computed as the geodesic distance from a point to its corresponding supernode. With $f_2(p)$, the ridge-separated features are identified. The right figure shows the distribution of values of f_2.

One phenomenon, that we have observed in the first phase of our segmentation process, is that the gradient field of f flows from a few local minima to several local maxima. This phenomenon can result in some artifacts. For example, some parts of torso are assigned to legs and arms of santa in Figure 7. Furthermore, when many segments merge at a local minimum, a large number of sampling points is required for the approximate centrality computation in order to obtain a reasonably good segmentation. We are considering an alternative approach where both local maxima as well as local minima of the gradient field create supernodes. We believe that this alternative approach will improve the quality of segmentation and also speedup the segmentation process.

Acknowledgements

The point sets used for our experiments were downloaded from on-line 3D scan repositories [1]. We used **qslim** [8] to generate low-resolution models.

This work was supported by the National Science Foundation under contracts ACI 9624034 (CAREER Award) and ACI 0222909, through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and a large Information Technology Research (ITR) grant, and Lawrence Livermore National Laboratory (B347878, B503159, B523294). We thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis.

References

[1] Level of detail for 3d graphics. http://lodbook.com/models/.

[2] Bremer, P.-T., Edelsbrunner, H., Hamann, B., and Pascucci, V. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics 10*, 4 (2004), 385–396.

[3] Cormen, T. H., Leiserson, C., Rivest, R. L., and Stein, C. *Introduction to Algorithm*, 2 ed. MIT Press, Cambridge, Massachusets, 2001.

[4] Dey, T. K., Giesen, J., and Goswami, S. Shape segmentation and matching with flow discretization. In *Proc. Workshop on Algorithms and Data Structure* (2003), vol. 2748 of *LNCS*, pp. 25–36.

[5] Eppstein, D., and Wang, J. Fast approximation of centrality. *Journal of Graph Algorithm and Applications 8*, 1 (2004), 39–45.

[6] Freeman, L. C. Centrality in social networks: Conceptual classification. *Social networks 1* (1979), 215–239.

[7] Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., and Dobkin, D. Modeling by example. *ACM Trans. Graphics 23*, 3 (2004), 652–663.

[8] Garland, M. Qslim simplification software. http://graphics.cs.uiuc.edu/∼garland/software/qslim.html.

[9] Garland, M., Willmott, A., and Heckbert, P. S. Hierarchical face clustering on polygonal surfaces. In *Proc. Symposium on Interactive 3D graphics* (2001), pp. 49–58.

[10] Gotsman, C. On graph partitioning, spectral analysis, and digital mesh processing. In *Proc. Intl. Conf. Shape Modeling and Applications* (2003), pp. 165–174.

[11] Gyulassy, A., Natarajan, V., Pascucci, V., Bremer, P.-T., and Hamann, B. Topology-based simplification for feature extraction from 3d scalar fields. In *Proc. IEEE Conf. Visualization* (2005), pp. 535–542.

[12] Heckel., B., Uva, A. E., and Hamann, B. Cluster-based generation of hierarchical surface models. In *Proc. Scientific Visualization* (1997), pp. 113–122.

[13] Hilaga, M., Shinagawa, Y., Komura, T., and Kunii, T. L. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proc. SIGGRAPH* (2001), pp. 203–212.

[14] Hitoshi, Y., Lee, S., Lee, Y., Ohtake, Y., Belyaev, A., and Seidel, H.-P. Feature sensitive mesh segmentation with mean shift. In *Proc. Intl. Conf. Shape Modeling and Applications* (2005), pp. 236–243.

[15] Jain, A. K., Murty, M. N., and Flynn, P. J. Data clustering: a review. *ACM Computing Surveys 31*, 3 (1999), 264–323.

[16] Katz, S., and Tal, A. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graphics 22*, 3 (2003), 954–961.

[17] Lanczos., C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Stand. 45* (1950), 255–281.

[18] Liu, R., and Zhang, H. Segmentation of 3d meshes through spectral clustering. In *Proc. Pacific Graphics* (2004), pp. 298–305.

[19] Mangan, A. P., and Whitaker, R. T. Partitioning 3d surface meshes using watershed segmentation. *IEEE Trans. Visualization and Computer Graphics 5*, 4 (1999), 308–321.

[20] Matsumoto, Y. *An Introduction to Morse Theory.* Amer. Math. Soc., 2002. Translated from Japanese by K. Hudson and M. Saito.

[21] Milnor., J. *Morse Theory.* Princeton Univ. Press, New Jersey, 1963.

[22] Mount, D. M., and Arya., S. Ann: A library for approximate nearest neighbor searching. http://www.cs.umd.edu/∼mount/ANN/.

[23] Natarajan, V., and Pascucci, V. Volumetric data analysis using Morse-Smale complexes. In *Proc. Intl. Conf. Shape Modeling and Applications* (2005), pp. 320–325.

[24] Page, D. L., Koschan, A., and Abidi, M. A. Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (2003), vol. 2, pp. 27–32.

[25] Pauly, M., Keiser, R., Kobbelt, L. P., and Gross, M. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics* (2003), 641–650.

[26] Roger, P., and Bohr., H. A new family of global protein shape descriptors. *ACM Computing Surveys 182* (2003), 167–181.

[27] Schloegel, K., Karypis, G., and Kumar, V. *CRPC Parallel Computing Handbook*. Morgan Kaufmann, 2000, ch. Graph partitioning for High performance scientific simulations.

[28] Shalfman, S., Tal, A., and Katz, S. Metamorphosis of polyhedral surfaces using decomposition. In *Proc. Eurographics* (2002), pp. 219–228.

[29] Shamir, A. A formulation of boundary mesh segmentation. In *Proc. Second International Symposium on 3DPVT* (2004), pp. 82–89.

[30] Shi, J., and Malik., J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22* (2000), 888–905.

[31] Tenebaum, J. B., de Silva, V., and Langford., J. C. A global geometric framework for nonlinear dimensionality reduction. *Science 190*, 5500 (2000), 2319–2323.

[32] Wu, K., and Levine, M. D. 3d part segmentation using simulated electrical charge distributions. *IEEE Trans. Pattern Analysis and Machine Intelligence 19*, 11 (1997), 1223–1235.

[33] Zhang, E., Mischaikow, K., and Turk, G. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics 24*, 1 (2005), 1–27.

[34] Zwicker, M., Pauly, M., Knoll, O., and Gross, M. Pointshop 3d: An interactive system for point-based surface editing. In *SIGGRAPH* (2002), pp. 322–329.