社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－CG－119（6）
2005／5／20

A Multiscale Decomposition of Point-sampled 3D Objects

Hamid Laga and Hiroki Takahashi and Suguru Saito and Masayuki Nakajima
Graduate School of Information Science and Engineering
Tokyo Institute of Technology, Japan
email:{hamid,suguru,rocky,nakajima}@img.cs.titech.ac.jp

Abstract

In this paper we propose a simple method for the decomposition of point-sampled 3D objects into its basic components. Our approach is based on recent methods for fuzzy clustering and hierarchical decomposition of 3D meshes, where we use instead a k-nearest neighbor graph of the point cloud. Our approach proceeds in two steps: We first encode the geometric properties of the 3D shape into an inter-surfel distance matrix. The distance between two surfels takes into account the geodesic distance, the angular distance to measure the shape convexity and the surface variation. Then, we apply a clustering algorithm on the distance matrix to extract the different components of the input surfels. We demonstrate the efficiency of the proposed approach on a collection of point sampled 3D objects.

1 Introduction

Decomposition is a technique used to simplify models of complex shape and topology into smaller sub-models that are easier to handle. 3D model decomposition into meaningful components benefits many applications. In content-based retrieval of 3D models [20], a decomposition graph serves as a non-rigid invariant signature. For part-based object recognition and classification [13], decomposition is used to detect meaningful components. In modification and modeling by example [11], segmentation is used to select parts to edit, search in a database for similar parts and composite them to create new models. Other applications include collision detection, metamorphosis, compression and simplification.

Within the computer graphics community, meshes are the most commonly used representation. Consequently many algorithms exist for the decomposition of 3D mesh surfaces.The algorithms differ in the technique, the type of the targeted components and the definition of the boundaries, i.e sharp lines or deep concavities.

Recently, point based representations have been proposed as an alternative to triangles for 3D surfaces. Decomposing point-based surfaces into meaningful components should add more modeling power to the flexibility of the representation. Unfortunately, mesh decomposition techniques are not applicable to point cloud unless one reconstructs a surface out of it, which undermines the fundamental philosophy and advantages of point-based models.

In this paper we make use of recent results on hierarchical decomposition using fuzzy clustering of 3D meshes to decompose point-sampled objects into meaningful components. This generalizes the methods proposed by [14, 16] for meshes. The advantage of this method is its simplicity, efficiency in boundary handling and its hierarchical nature. Moreover, generalizing 3D object processing methods will motivate the use of hybrid representations.

Point-sampled 3D object decomposition benefits many applications including 3D reconstruction, shape matching, recognition and animation.

1.1 Related work

The minima rule theory [12] defines a framework for how human perception might decompose an object into its constituent parts. It states that human perception partitions objects into convex components [19], i.e the boundaries are lines of negative curvature.

Based on this theory, many algorithms have been introduced. Page et al.[17] extend the watershed algorithm to decompose 3D mesh surfaces. Bespalov et al.[3] uses hierarchical decomposition of a 3D model into features based on its spectral properties. The resulting representation is then used for topological matching of 3D objects. Katz et al.[14] combines hierarchical mesh decomposition and fuzzy clustering and cuts to decompose a 3D shape into convex components. The decomposition is then used for control-skeleton extraction. Later, Liu et al.[16] extended this approach using spectral clustering.

Lien et al.[15] captures the key structural features of a 3D mesh model using an approximate convex decomposition. They use the similarity of the object to its convex hull as a measure of the importance of non-convex features. They demonstrated the efficiency on 2D polygons, genus-0 and genus-1 3D objects but not on models with complex topology.

Another approach is to decompose objects into components bounded by line features. Clarenz et al.[7] proposed a local surface classifier based on the zero and first order moments. This classifier has been used for surface fairing and decomposition on mesh models [7, 4] and later extended to point-sampled models [5, 6] using a strictly local Delaunay meshing. This approach is bottom-up and aims to detect first the line features and then extract components bounded by these features. One problem of this algorithm is that, unlike CAD/CAM models, natural objects contain parts separated by incomplete and fuzzy lines, or even not separated by lines. Furthermore, the approach is local since it identifies features from local variations of the surface.

Dey et al.[8] presented a shape segmentation algorithm from noisy point cloud. Although they bypass the explicit surface reconstruction step, they require a global Delaunay triangulation of the point set to determine the tetrahedra that have to be clustered together to form segments of the shape interior.

1.2 Our approach

In this paper, we generalize and apply for the first time, the methods proposed by [14, 16] for hierarchical mesh decomposition by adapting specific parts to point-based surfaces. For completeness, we briefly overview this algorithm in Figure 1, but refer the reader to the related papers for more details.

Function ConvexMeshDecompose(Mesh (V, F))

1. **Dual graph:** construct a dual graph G by connecting each face to its adjacent faces.
2. **Distance matrix:** assigning distances to all pairs of nodes in the dual graph.
3. **Probability assignment:** after computing an initial decomposition, assigning each face a probability of belonging to each patch.
4. **Fuzzy decomposition:** using an iterative clustering scheme, the probability values are refined.
5. **Boundary refinement:** constructing the exact boundaries between the components, thus transforming the fuzzy decomposition into the final one.

Figure 1. Hierarchical mesh decomposition using fuzzy clustering and cuts [14]

In our approach, we replace the dual graph by the local neighborhood graph, where each node is associated to a surfel. We investigate the k-nearest neighborhood graph, , and the local 2D Delaunay graph. The distance matrix encodes the distances between all pairs of *surfels*. To handle large models we use progressive point set simplification [18, 9].

The remaining parts of this paper are organized as follow; section 2 details the decomposition algorithm adapted to point set surfaces. Issues relevant to different neighborhood graphs are discussed in section 2.2. In section 2.3 we explain how do we handle large models using progressive point set simplification. The clustering algorithm is presented in section 2.4. In section 2.5 we summarize the algorithm parameters and their settings. Results and discussion are shown in section 3. We end with conclusions and issues for future work in section 4.

2 Point-based surface decomposition

We propose to adapt the hierarchical decomposition and fuzzy clustering of 3D meshes to decompose point-based surfaces. This involves replacing the dual graph, in step 1 of Katz algorithm (Figure 1), by a local neighborhood graph, and adapting the distance matrix to satisfy the metric conditions.

The key steps are the first and second ones: adjacency graph and distance matrix construction. We will now describe them in detail. Let us denote by $P = \{p_i | i = 1, \ldots, n\}$ a set of surfel centers approximating a surface S, and $\aleph_i$ the index set of the neighbors of surfel i.

2.1 Distance matrix

In order to partition a point-sample surface along edges of deep concavities, we group *surfels* instead of faces, i.e., we construct the distance matrix with respect to the connectivity of the local neighborhood graph of the surfels. The pairwise face distances used by [14] model the minima rule and we use them to build the inter-surfel distance matrix.

The distance a_{ij} between two adjacent surfels p_i and p_j is defined as:

$$a_{ij} = \delta \cdot \frac{geod\_dist(p_i, p_j)}{avg\_geod} + (1-\delta) \cdot \frac{ang\_dist(p_i, p_j)}{avg\_ang} \quad (1)$$

avg_geod and avg_ang are, respectively, the average geodesic and average angular distances. The first term of equation 1 accounts for the normalized geodesic distance, while the second term measures the angular distance. δ is a weighting parameter set to 0.02 in all our experiments to penalize concave regions. Now the distance between any pair of surfels p_k and p_l is the length of the shortest path between the two surfels.

On point-sampled surfaces and under the sampling density requirements, i.e., if the underlying surface is sampled sufficiently dense [2], the geodesic distance between two adjacent surfels is simply approximated by the Euclidian distance between their centers:

$$geod\_dist(p_i, p_j) = ||p_i - p_j|| \quad (2)$$

Similarly, the angular distance is measured using the angle between the normals of adjacent surfels:

$$ang\_dist(p_i, p_j) = \eta(1 - \cos(angle(N_i, N_j))) \quad (3)$$

where η is a real parameter set to 1 for concave angles and 0.1 for convex angles.

2.2 Neighborhood graph

For point-sampled surfaces, all computations are based purely on neighborhoods induced by the Euclidean vicinity instead of combinatorial proximity in the mesh settings. A number of approaches have been studied in [10]. In this section we discuss two concrete choices for the neighborhood graph which we have tested.

2.2.1 The k-nearest neighborhood

The neighborhood of each surfel p_i is the set of $k-$nearest surfels. This choice is adequate when the point-set is dense and the distribution of the points is reasonably uniform. If one of these conditions is not satisfied, the path between two surfels may not exist, forcing the algorithm to classify them into two different components.

2.2.2 The 2D local Delaunay neighborhood

Similar to [10] and [6], we collect k points p_j around the point p_i, fit a least square plane and project the points onto that plane. Then we triangulate the projected points using a 2D Delaunay triangulation. The neighborhood $\aleph_i$ of the point p_i is then the one-ring around p_i.

Notice that, for the purpose of inter-surfel distance computation, we do not need to eliminate from the triangulation triangles with too small angles, a well known problem for discretizing PDEs [6].

This triangulation method is appropriate when the points are unevenly distributed locally which arises after the progressive point-set simplification (section 2.3).

A direct consequence of using local neighborhood graphs is that, unlike mesh models, the distance matrix A is not symmetric. In fact, the shortest pathes from p_i to p_j and from p_j to p_i might be different. In order to obtain a symmetric distance matrix, which would possess more desirable properties, we use

$$a_{ij} = min\{dist(p_i, p_j), dist(p_j, p_i)\} \quad (4)$$

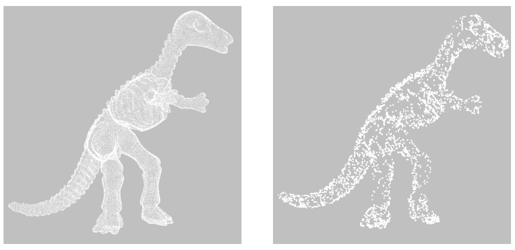
original 56194 surfels simplified $3K$ surfels

Figure 2. Progressive simplification of the input point sets.

With this transformation, the distance matrix A satisfies the metric requirements.

2.3 Handling large models

Similarly to mesh decomposition [14], the distance matrix is large and dense. To handle large data sets we make use of progressive point set simplification [18, 9].

First, we reduce the original point set $P = \{p_1, \dots, p_n\}$ into a base point set $\tilde{P} = \{\tilde{p}_1, \dots, \tilde{p}_m\}$, with $m << n$ using the neighborhood collapse operation. Then we build a surface patch list by grouping into a single patch all points collapsed into the same point $\tilde{p}_i$. That is, we view $\tilde{P}$ as a set of patch representative centers. We run the decomposition on $\tilde{P}$. Then, we assign each point in the patch i to the surface component to which the patch center $\tilde{p}_i$ has been affected. Note that:

- There is a lower limit for m, the base-set size. Otherwise, the local neighborhood graph is not consistent (not planar any more). In our experiments we set $m = 3K$ points.
- Progressive simplification induces a smoothing effect on the surface. In our implementation, the normal vector to each surfel is computed at the original set and kept along the simplification. Otherwise the angular distances will be affected.

Figure 2 shows the Dinosaur (56194 surfels)to $3K$ surfels. We can see that the points are unevenly distributed on the surface, thus, using local 2D Delaunay neighborhood is more appropriate.

2.4 Hierarchical clustering

Given the reduced point set $\tilde{P}$, the decomposition algorithm performs as follows:

1. **Distance matrix:** compute the distance matrix A using the local neighborhood graph.

2. **Initialize the number of components and their centers.**.

3. **Clustering:** perform fuzzy C-Means clustering on A.

4. **Weight assignment** as an estimate of the convexity of each component and insert them into a priority queue.

5. **Repeat from 2** using the component of high priority.

Our algorithm differs from previous ones [14] in step 4. In [14], every component is further decomposed in the order they are detected, but this can easily run into asymmetric decomposition even for symmetric objects.

In our case, we assign a weight w_i to each detected component C_i as follows:

$$w_i = \frac{1}{n_i} \sum_{p_j \in C_i} a_{ij} \tag{5}$$

provided that $p_i \in P$ is the representative point of the component C_i, and n_i is its size. w_i gives a measure of the concavity of the detected component. This way we achieve quite symmetric decompositions.

2.5 Algorithm parameters

The inter-surfel distance function involves two parameters: δ and η (equations 1 and 3 respectively). δ balances between the geodesic distance and the angular distance. In all our experiments we set $\delta = 0.02$. That is the angular distance contributes more to the overall pairwise distance. η, in the other hand, is used to penalize concave regions. Similar to mesh models [14, 16], we set $\eta = 0.1$ for convex angles and $\eta = 1$ for concave angles.

For the neighborhood graph, we use the local 2D Delaunay triangulation. It requires setting the maximum neighborhood size. Through our experiments, we found that values between 16 and 30 offer a good balance between the computation time and decomposition quality. This parameter is set to 22 in all the examples given in section 3.

Finally, the progressive point set simplification requires to specify the size of the base point set. In all our experiments we set it to $3K$ points. As a result, all the decompositions can be performed in a constant time.

3 Experimental results

We implemented the decomposition algorithm described in this paper as a plugin to Pointshop3D system [1]. The progressive point set simplification, however, is implemented separately as an external module using JAVA.

We have run our algorithm on a variety of point-based models. Figure 3 shows the hierarchical binary decomposition of the dinosaur and Santa models $(56,194$ and $75,781$

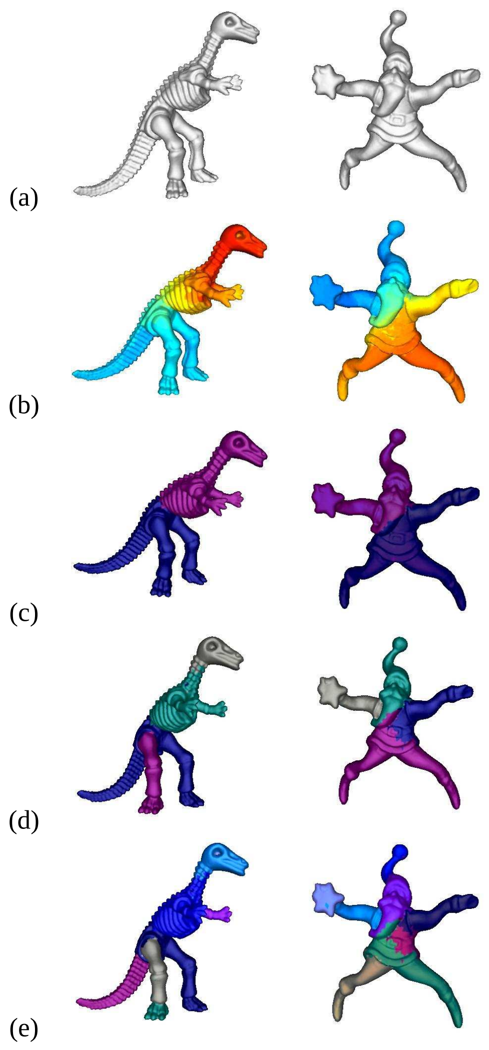

Figure 3. Hierarchical binary decomposition of the dinosaur and Santa models. (a) Input model. (b) plot of the probabilities. (c), (d) and (e) First, second and third level decompositions.

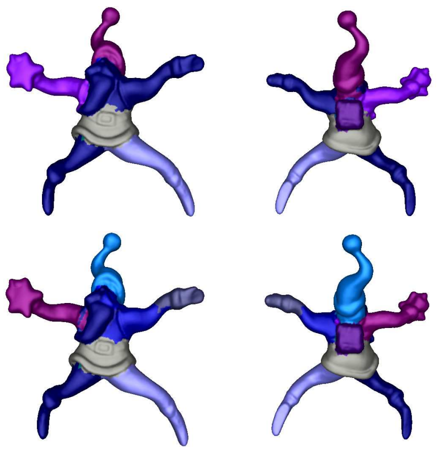

Figure 4. Hierarchical k-ways decomposition of the Santa model with front and back views using 2D local Delaunay neighborhood.

surfels respectively). We used the local $2D$ Delaunay triangulation to compute the distance matrix. Figure 3-(b) shows for each surfel the probability of belonging to the upper component (after one level decomposition).

Figures 4 and 5 show the k-ways decomposition of the same models with an automatic estimation of the number of components at each level. In these two examples, we used the local 2D Delaunay neighborhood. We can see on the Santa model $Figure 4$ that the main components of the model have been efficiently extracted. Moreover, because the most concave component is decomposed first, we achieve plausible symmetric decomposition.

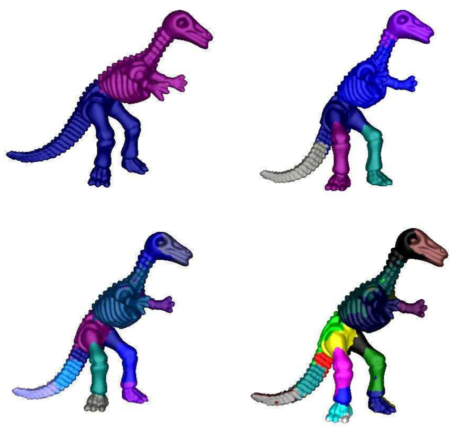

Figure 5. Hierarchical k-ways decomposition of the dinosaur model.

Other decomposition results are illustrated in Figure 6. Finally, Figures 4 and 7 show results for different choices

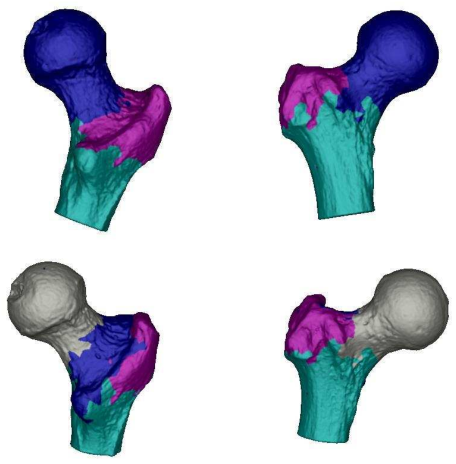

Figure 6. Decomposition of the balljoint model ($137,062$ points) into 3 (top) and 4 (bottom) components.

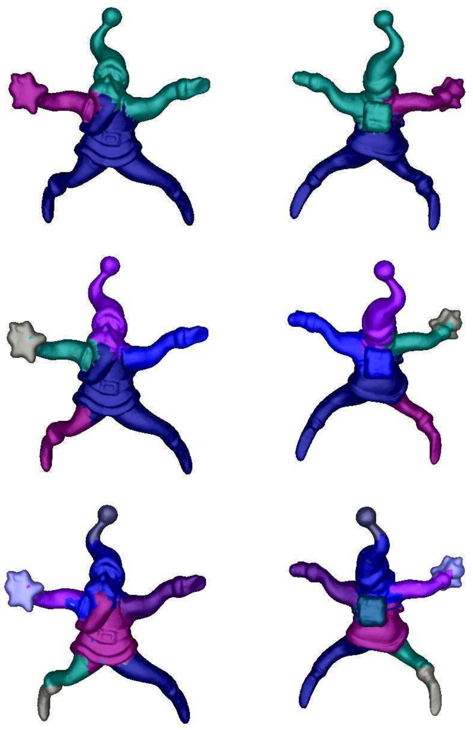

Figure 7. Hierarchical k-ways decomposition of the Santa model with front and back views using k-nearest neighborhood graph.

of the local neighborhood graph. In figure 4 where we used 2D Delaunay neighborhood with maximum neighborhood size equal to 22, the main components of the model have been efficiently extracted. While in 7, where we used the k-nearest neighborhood graph with $k = 22$, features such as the Santa's beard couldn't be extracted. Clearly local 2D Delaunay neighborhood leads to better decomposition.

4 Conclusion and discussion

In this paper we reported the results of applying hierarchical decomposition using fuzzy clustering on point-sampled 3D objects. We have shown that, with a careful choice of the adjacency graph, we are able to obtain very good results on clean point sets. We believe that, using a unified processing framework for different representations, in our case mesh and points, will add more flexibility to hybrid representations.

In term of improving efficiency, there are a number of issues to investigate. This include the use of other distance functions taking into account other surface properties such as: surface variation for line boundaries, and color and texture information. Also to achieve a fully automatic decomposition, the parameters need to be estimated automatically, which requires more formal investigation. Finally, we also plan to experiment with hybrid representations.

References

[1] *http://www.pointshop3d.com.*

[2] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. pages 415–421, 1998.

[3] D. Bespalov, A. Shokoufandeh, W. C. Regli, and W. Sun. Scale-space representation of 3d models and topological matching. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 208–215. ACM Press, 2003.

[4] U. Clarenz, M. Griebel, M. Rumpf, M. A. Schweitzer, and A. Telea. Feature sensitive multiscale editing on surfaces. *Visual Computer*, 20(5):329–343, 2004.

[5] U. Clarenz, M. Rumpf, and A. Telea. Fairing of point based surfaces. In *Computer Graphics International*, pages 600–603, 2004.

[6] U. Clarenz, M. Rumpf, and A. Telea. Finite elements on point based surfaces. In *Eurographics Symposium on Point-Based Graphics*, 2004.

[7] U. Clarenz, M. Rumpf, and A. Telea. Robust feature detection and local classification for surfaces based on moment analysis. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):516–524, 2004.

[8] T. K. Dey, J. Giesen, and S. Goswami. Shape segmentation and matching from noisy point clouds. In *Eurographics Symposium on Point-Based Graphics*, pages 193–199, 2004.

[9] S. Fleishman, D. Cohen-Or, M. Alexa, and C. T. Silva. Progressive point set surfaces. *ACM Transactions on Graphics*, 22(4):997–1011, 2003.

[10] M. S. Floater and M. Reimers. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design*, 18(2):77–92, 2001.

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

[12] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.

[13] D. Huber, A. Kapuria, R. Donamukkala, and M. Hebert. Parts-based 3d object classification. pages 82–89, 2004.

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

[15] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polygons. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 17–26. ACM Press, 2004.

[16] R. Liu and H. Zhang. Segmentation of 3d meshes through spectral clustering. In *Pacific Conference on Computer Graphics and Applications*, pages 298–305. IEEE Computer Society, 2004.

[17] D. L. Page, A. Koschan, and M. A. Abidi. Perception-based 3d triangle mesh segmentation using fast marching watersheds. pages 27–32, 2003.

[18] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization '02*, pages 163–170. IEEE Computer Society, 2002.

[19] L. Walker and J. Malik. Can convexity explain how humans segment objects into parts? *Journal of Vision*, 3(9):503a, 2003.

[20] E. Zuckerberger, A. Tal, and S. Shlafman. Polyhedral surface decomposition with applications. *Computers & Graphics*, 26(5):733–743, 2002.