Segmentation of 3D Articulated Components by Slice-Based Vertex-Weighted Reeb Graph

Nilanjana Karmakar¹, Partha Bhowmick², and Arindam Biswas¹

¹ Department of Information Technology,
Indian Institute of Engineering Science and Technology, Shibpur, India
{nilanjana.nk,barindam}@gmail.com

² Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur, India
bhowmick@gmail.com

Abstract. A fast and efficient algorithm for segmentation of the articulated components of 3D objects is proposed. The algorithm is marked by several novel features, such as DCEL-based fast *orthogonal slicing*, *weighted Reeb graph* with slice areas as vertex weights, and graph cut by *exponential averaging*. Each of the three sets of orthogonal slices obtained from the object is represented by a vertex-weighted Reeb graph of low complexity, as the slicing is done with an appropriate grid resolution. Each linear subgraph in a Reeb graph is traversed from its leaf node up to an articulation node or up to a node whose weight exceeds a dynamically-set threshold, based on exponential averaging of the predecessor weights in the concerned subgraph. The nodes visited in each linear subgraph are marked by a unique component number, thereby helping the inverse mapping for marking the articulated regions during final segmentation. Theoretical analysis shows that the algorithm runs faster for objects with smaller surface area and for larger grid resolutions. The algorithm is stable, invariant to rotation, and leads to natural segmentation, as evidenced by experimentation with a varied dataset.

Keywords: 3D segmentation, DCEL, orthogonal slicing, orthogonal polyhedron, Reeb graph.

1 Introduction

Segmentation of 3D triangulated meshes has an authoritative impact on shape-analytic applications. Hence, 3D segmentation has been attempted over and over again in a wide range of ways. For example, segmentation by multi-dimensional scaling and feature points is proposed in [13]. The techniques in [6,10] are based on shape diameter function, skeletons, and randomized cuts. Diffusion distance metric and a variety of medial structures are used in [9]. The works in [13,10] provide segmentation along the natural seams of an object, whereas the work in [6] concentrates on object volume, object posture, and topology. The notion of topological maps and region adjacency graphs has been proposed in [7]. Other

Table 1. Some of the existing algorithms versus the proposed algorithm

Algorithm	Features
Nurre, 1997 [15]	Input: 3D point cloud, organized into slices for segmentation, works only on the object human body
Ju <i>et al.</i> , 2000 [11]	Input: 3D point cloud, curvature analysis of profiles, works only on the object human body
Y. Xiao <i>et al.</i> , 2003 [17]	Input: 3D point cloud, Reeb graph, works only on the object human body
Katz <i>et al.</i> , 2005 [13]	Hierarchical, pose-invariant, feature-based, core extraction by spherical mirroring, natural seam segmentation
Golovinskiy-Funkhouser, 2008 [10]	Hierarchical, pose-invariant, randomized cuts, partition function, natural seam segmentation
Shapira <i>et al.</i> , 2008 [6]	Hierarchical, pose-invariant, skeleton extraction by shape diameter function, focuses on invariance of object volume
Goes <i>et al.</i> , 2008 [9]	Hierarchical, pose-invariant, diffusion distance metric, medial structure
<i>Proposed algorithm</i>	Orthogonal slicing, rotation-invariant, based on Reeb graph analysis, speedy execution, natural segmentation.

related techniques can be seen in [2,5,8,19]. Table 1 provides a brief comparison of the existing algorithms.

Irrespective of the adopted technique, there always lies a chance of over-segmentation [4,14], which, at times, affects the accuracy of segmentation, and sometimes may also assist in deriving important information about the object [18]. With an objective of obtaining natural segmentation, which is free of skeletonization, and hence fast, robust, and rotationally invariant, we propose here a novel algorithm, which is based on the idea of orthogonal slicing [12], followed by Reeb graph analysis of the slice sets. The only parameter of orthogonal slicing is the grid resolution, g , which is shown to have insignificant impact on the output quality. The proposed segmentation algorithm may be useful for analysis of geometric structures and peripheral topology of objects, which, in turn, may be quite effective for shape matching, collision detection, texture mapping, etc.

Graph-theoretic analysis of 3D objects leads to a stable and dependable performance for 3D shape analysis. Reeb graph, in particular, provides a natural representation that is suitable for object surface representation due to its one-dimensional graph structure and invariance to both global and local transformations [16,1,3]. Given an unorganized cloud of 3D points, identification of human body parts has been attempted in several works. In [15], the data set is organized into a stack of slices so that a specific set of data points belong to a specific part of the object according to its topology. This approach is further improved by curvature analysis [11]. A further modification using Reeb graph is proposed in [17]. However, there is a difference between the work in [17] and the one proposed by us. In [17], a 3D point cloud is used as input, which has the disadvantage of high space complexity; our algorithm, on the contrary, works with triangulated mesh as input. Secondly, the algorithm in [17] is effective only for objects like

human body, whereas our algorithm is applicable to a wide range of objects including human body.

2 Preprocessing and Orthogonal Slicing

In our work, a discrete 3D object A is considered as a triangulated mesh in which the triangles truly capture the peripheral topology of the object, i.e., on each edge of the triangulated surface, exactly two triangles are incident. As a preprocessing step, a 3D grid $\mathbb{G}$ of size (resolution) g is imposed on A , and three sets of orthogonal slices (henceforth mentioned as *slices* or *slice polygons*) are obtained by slicing the object along the three coordinate planes. These three slice sets are computed using a combinatorial technique similar to the one proposed in [12]. Each set of slices is processed to identify the subset of slices belonging to each articulated portion and central region of the object. By articulated portion, we mean the meaningful components of the object [1,9,3]. Finally, for each articulated portion, the three subsets of slices are combined to identify the object segment.

The grid $\mathbb{G}$ is represented as a set of *unit grid cubes* (UGCs), each of size g . If a triangle $T_{abc}(v_a, v_b, v_c)$ intersects or lies inside a UGC U_k , then U_k is considered as *object-occupied*. U_k is intersected by T_{abc} if at least one UGC-face $f_k \in U_k$ is intersected by T_{abc} . The projections of f_k and T_{abc} are considered along yz -, zx -, and xy -planes to find the intersection based on the nature of projection of f_k (line segment or square) and that of T_{abc} (line segment or triangle).

Once the object-occupied UGCs are identified, the set of orthogonal slices parallel to each of the yz -, zx -, and xy -planes are determined. The condition of *object occupancy*, as stated in [12], has been modified for this. In [12], a UGC-face is considered as object-occupied if one or both of its adjacent UGCs contains object *voxels*. But in the current work, a UGC-face is considered as object-occupied if exactly one of its adjacent UGCs is intersected by a triangle. The slice polygons are stored in an adjacency list containing n lists, one for each slicing plane, n being the number of slicing planes. Each list contains a sequence of vertices for all the slice polygons on that slicing plane. Apart from this, each slice polygon is stored in two lexicographically sorted lists, L_{xy} and L_{yx} (or L_{yz} and L_{zy} , or L_{zx} and L_{xz}), sorted according to x - and y -(or y - and z -, or z - and x -)coordinates, respectively.

3 Reeb Graph Construction

W.l.o.g., consider the set of slice polygons S_y formed by a set of slicing planes $\Pi = \{\Pi_1, \Pi_2, \dots, \Pi_n\}$, parallel to the zx -plane. Note that one or more slice polygons may lie on each slicing plane Π_i . If two slice polygons P' and P'' lie on two consecutive planes, Π_i and Π_{i+1} , so that they have a non-empty intersection when one is projected on the other, then P' and P'' are called *consecutive slices*. If P' lies on Π_i such that exactly one consecutive slice polygon exists either on Π_{i+1} or on Π_{i-1} , then P' is said to be a *leaf slice*.

Fig. 1. Segmentation of slices in S_y . In R_y and G_y , node weights are shown. G_y is segmented with difference in area threshold—shown beside each edge—computed at every slice as exponential average of the previous slice areas.

The set S_y is represented by a (weighted) *slice graph* G_y . Each node of G_y corresponds to a slice polygon of S_y ; two nodes have an edge if their corresponding slices are consecutive. The area of a slice is assigned as the weight of its corresponding node. This graph results to *weighted Reeb graph*. As a Reeb graph provides a topological signature of an object, we use it for identification of object articulations based on the peripheral topology captured through three orthogonal sets of slices. A Reeb graph, R_y , corresponding to S_y , is shown in Fig. 1(b). It consists of only leaf nodes and *non-linear articulation nodes* (NLA, of degree ≥ 2). Observe that a critical point of the object, at which the surface topology changes, corresponds to an NLA. The slice graph G_y , on the contrary, contains linear substructures in addition to leaf nodes and NLAs, as illustrated in Fig. 1(c).

Fig. 2. Different positions of triangles on the object surface w.r.t. slicing planes

During the construction of G_y from S_y , a list of leaf nodes, L_y , is maintained. A leaf node ν_0 is enqueued in a queue Q and BFS starts from ν_0 in G_y . While dequeuing ν_0 , the difference of its weight with a dynamically-set threshold τ is checked. If the difference lies within λ times the weight of the last-dequeued node, where λ varies between 0 and 1, then ν_0 is marked by a *component id*, and all the unvisited neighboring nodes of ν_0 are enqueued in Q . Otherwise, ν_0 is considered as a leaf node and appended to L_y . The nodes visited by a single BFS traversal are marked by a unique component id. Next, BFS traversal is started from another leaf node, and the process is continued until all the leaf nodes are visited. The BFS forest finally results to identification of all components.

The threshold τ is decided by *exponential averaging* with the weights of the previous nodes having the same component id. As the traversal starts from a leaf node ν_0 , the area of ν_0 is used to initialize the threshold value τ_0 . Based on the area of ν_i , at each subsequent node ν_{i+1} , its area is compared with τ_{i+1} , which is computed as $\tau_{i+1} = \rho w_i + (1 - \rho)\tau_i$, where, τ_i denotes the exponential average of the weights of the last i nodes, and w_i denotes the weight of ν_i . The value of ρ can range between 0 and 1; however, we conventionally take it as $\frac{1}{2}$. The node ν_{i+1} belongs to the same component as ν_i if $|w_{i+1} - \tau_{i+1}| < \lambda w_i$, where λ ranges between 0 and 1. The use of exponential averaging for dynamically setting the threshold value ensures natural articulation for widely varying component dimensions. Figure 1 shows a demonstration. The threshold values are computed in the direction the algorithm proceeds; the leaf slices are considered in the order 1, 2, 5, 18, 19, and finally node 8 for the central region.

4 Segmentation by Weighted Reeb Graph

The set of triangles is represented by a doubly connected edge list (DCEL) [12]. For each vertex of a triangle, the neighboring vertices are obtained in constant time, using DCEL. Depending on the positions of its three vertices, a triangle T_{abc} may be intercepted by or may lie on Π_{i-1} (Fig. 2). Hence, a vertex v_a lies on Π_i or between Π_i and Π_{i+1} . Since the set of slices parallel to a coordinate plane are stored as lexicographically sorted lists L_{xy} and L_{yx} (Sec. 2), the coordinate values for the vertex v_a are looked up in L_{xy} and L_{yx} to find the slice polygon

Algorithm 3DSEGMENT($A, \mathbb{G}$) 01. for each coordinate plane t 02. $G_t \leftarrow$ graph representing S_t 03. $L_t \leftarrow$ leaf nodes 04. GRAPH-SEGMENT(G_t, L_t) 05. $V_t \leftarrow$ triangle vertices with component ids 06. VERTEX-SEGMENT(V_t)	Procedure VERTEX-SEGMENT(V) 01. count $\leftarrow 1$ 02. for each vertex $v \in V$ 03. if $visited[v] = 0$ 04. $segid[v] \leftarrow -1$ 05. for each neighbor u of v 06. if $visited[u] = 1$ 07. if (COMPARE(u, v)) 08. $segid[v] \leftarrow segid[u]$ 09. if $segid[v] = -1$ 10. $segid[v] \leftarrow$ count 11. count $\leftarrow$ count + 1 12. $visited[v] \leftarrow 1$
Procedure GRAPH-SEGMENT(G_t, L_t) 01. count $\leftarrow 1$ 02. for each leaf node $\nu \in L_t$ 03. if $visited[\nu] = 0$ 04. ENQUEUE(Q, ν) 05. $\nu_1 \leftarrow \nu$ 06. while Q is not empty 07. $\nu \leftarrow$ DEQUEUE(Q) 08. if $ w[\nu] - \tau_w < \lambda \cdot w[\nu_1]$ 09. $compid[\nu] \leftarrow$ count 10. for each neighbor ν' of ν 11. if $visited[\nu'] = 0$ 12. ENQUEUE(Q, ν') 13. $visited[\nu] \leftarrow 1$ 14. $\nu_1 \leftarrow \nu$ 15. else 16. $L_t \leftarrow L_t \cup \{\nu'\}$ 17. count $\leftarrow$ count + 1	Procedure COMPARE(u, v) 01. if ((($c_x^v = c_x^u$) $\wedge$ ($c_y^v = c_y^u$) $\wedge$ ($c_z^v = c_z^u$)) $\vee$ ((($c_x^v = c_x^u$) $\wedge$ ($c_y^v = c_y^u$) $\wedge$ ($c_z^v \neq c_z^u$)) $\wedge$ (($c_x^v \neq -1$) $\vee$ ($c_y^v \neq -1$))) $\vee$ ((($c_x^v = c_x^u$) $\wedge$ ($c_y^v \neq c_y^u$) $\wedge$ ($c_z^v \neq c_z^u$)) $\wedge$ (($c_x^v \neq -1$) $\wedge$ ($c_y^v = c_z^v = -1$)))) 02. return 1 03. else if ((($c_x^v \neq c_x^u$) $\wedge$ ($c_y^v \neq c_y^u$) $\wedge$ ($c_z^v \neq c_z^u$)) $\vee$ ((($c_x^v = c_x^u$) $\wedge$ ($c_y^v = c_y^u$) $\wedge$ ($c_z^v \neq c_z^u$)) $\wedge$ ($c_x^v = c_y^v = -1$)) $\vee$ ((($c_x^v = c_x^u$) $\wedge$ ($c_y^v \neq c_y^u$) $\wedge$ ($c_z^v \neq c_z^u$)) $\wedge$ (($c_x^v = -1$) $\wedge$ (!($c_y^v = c_z^v = -1$)))))) 04. return 0 05. return -1

Fig. 3. The algorithm for segmentation and its related procedures

S to which v_a belongs. Consequently, v_a is assigned the same component id as S . This process is repeated for the vertices of all the triangles, and thus, each such vertex obtains a 3-tuple of component ids corresponding to S_x, S_y, S_z .

Let v be a vertex belonging to triangle T_{abc} . 3-tuples of component ids are assigned to v and all its adjacent vertices in the process outlined above. In order to assign segment id to v , its 3-tuple of component ids, namely (c_x^v, c_y^v, c_z^v) , is compared with those of all its neighboring vertices, using DCEL. Then, based on certain combinatorial rules (R1–R4), segment ids are assigned. A segment id signifies the combined id obtained from the 3-tuple of component ids. The procedure COMPARE in Fig. 3 enumerates the combinatorial possibilities. Note that, when the algorithm 3DSEGMENT terminates, we have as many segments as the number of distinct segment ids. It accepts the 3D discrete object A in the form of a triangulated mesh as input and uses the following procedures:

GRAPH-SEGMENT segments the Reeb graph, thereby identifying the slices that belong to each articulated portion or central region along each coordinate plane.

VERTEX-SEGMENT combines the slices belonging to different components along the three coordinate planes, in order to find the final segments of the object.

PROCEDURE COMPARE uses the rules for comparing the component ids of two vertices to decide whether they belong to the same segment or not.

Let (c_x^v, c_y^v, c_z^v) and (c_x^u, c_y^u, c_z^u) denote the 3-tuples of component ids for vertices v and u , and their final segment ids be s_u and s_v , respectively.

R1 If $c_x^v = c_x^u$, $c_y^v = c_y^u$, and $c_z^v = c_z^u$, then s_u is assigned to s_v .

R2A If $c_x^v = c_x^u$, $c_y^v = c_y^u$, and $c_z^v \neq c_z^u$ where $\exists c \in \{c_x^v, c_y^v\}$ such that $c \neq -1$, then s_u is assigned to s_v .

R2B If $c_x^v = c_x^u$, $c_y^v = c_y^u$, and $c_z^v \neq c_z^u$ where $\forall c \in \{c_x^v, c_y^v\}$ such that $c = -1$, then a new value is assigned to s_v .

R3A If $c_x^v = c_x^u$, $c_y^v \neq c_y^u$, and $c_z^v \neq c_z^u$ such that $c_x^v \neq -1$ and $c_y^v = c_z^v = -1$, then s_u is assigned to s_v .

R3B If $c_x^v = c_x^u$, $c_y^v \neq c_y^u$, and $c_z^v \neq c_z^u$ such that $c_x^v = -1$ and not both of c_y^v and c_z^v are equal to -1 , then a new value is assigned to s_v .

R4 If $c_x^v \neq c_x^u$, $c_y^v \neq c_y^u$, and $c_z^v \neq c_z^u$, then a new value is assigned to s_v .

Note that c_x^v , c_y^v , and c_z^v may be used interchangeably in the rules R2A, R2B, R3A, and R3B, where their comparisons with c_x^u , c_y^u , and c_z^u are considered. Vertices belonging to same (or different) components along all the three coordinate planes, belong to the same (or different) segments (R1 and R4). Along each coordinate plane, the central region (indicated by -1) is adjacent to all other components but neither of the components are adjacent to each other. Hence, the rules R2 and R3 are primarily based on whether v or u lie in the central region along any coordinate plane.

4.1 Time Complexity

Let n be the number of UGCs intersected by the surface of the object A , α be the surface area of A , β its volume, and g be the grid resolution. Then, number of UGC-faces on the object surface is $n_f = O(n)$. If A has a sufficiently large area-to-volume ratio, then n UGCs would cover its volume, i.e., $\beta < ng^3$. For a sufficiently small area-to-volume ratio, $\alpha > ng^2$. So, in general, $\beta/g^3 < n < \alpha/g^2$, or, $n_f = O(\alpha/g^2)$.

The number of slices in S_z is $|S_z| = O(n_f)$. Construction of R_z requires identification of consecutive slices from the sorted lists L_{xy} and L_{yx} . Each slice polygon P is traversed exactly once, which needs $O(\log |S_z|) = O(\log n_f)$ time for searching its first vertex in L_{xy} and L_{yx} . Traversal time of P is linear in its perimeter, as DCEL is used. So, Reeb graph construction time is $O(n_f \log n_f)$, since total traversal time is upper-bounded by $O(n_f)$.

The procedure GRAPH-SEGMENT is called thrice, hence requiring $O(n_f)$ time. The procedure VERTEX-SEGMENT executes on the UGCs covering the object surface, hence requiring $O(n)$ time, as Procedure COMPARE executes in constant time. So, the total time complexity is $O(n_f \log n_f) = O(\alpha/g^2 \log(\alpha/g^2))$.

Fig. 4. Results on Dog ($g = 2$). From top-left to bottom-right: Slices parallel to yz -, xz -, xy -planes (with segments marked), and final segmentation

Fig. 5. Results on Airplane ($g = 2$)

5 Results and Conclusion

The proposed algorithm has been implemented in C in Linux Fedora Release 7, Dual Intel Xeon Processor 2.8 GHz, 800 MHz FSB. As found from experimental results (Figs. 4–8), the algorithm can successfully separate out the limb-like articulated portions of the objects from its central region up to a high degree of accuracy. It may be noticed from these results that the final segmentation is a correctly-synthesized output from the three sets of orthogonal slices. The

Fig. 6. Results on Octopus ($g = 2$)

dynamically set thresholding for graph cut adds to the robustness in the behavior of the algorithm.

For a given object, the algorithm gives almost equally accurate results for different grid sizes within a reasonable range, and even when the object is tilted at an arbitrary angle (Fig. 8), which demonstrates its stability. An articulated part of a 3D object may have an arbitrary orientation w.r.t. the coordinate system, and hence such a part is identified by correlating its three sets of slices taken along the three coordinate planes. In particular, when the concerned part goes on arbitrarily changing in orientation, no less than three sets of slices can fully capture its shape. This is confirmed by the segmentation results presented in Figs. 4 to 6, and particularly through the results in Figs. 7 and 8. Some more results and statistical data related to our experimentation are given in Appendix. The standard deviation of the segment areas for ‘Chair’ rotated from 10° through 90° demonstrates the robustness of the algorithm w.r.t. rotation. The accuracy of segmentation with varying threshold values is also shown there for ‘Dog’ and ‘Airplane’.

The CPU time of segmentation increases with increase in the surface area of the object and also with the total number of slices along the three coordinate planes; and with grid size increase, the CPU time falls more than quadratically,

Fig. 7. Results on Chair (top two rows: $g = 2$, bottom two rows: $g = 4$)

Table 2. Statistical results and CPU times for segmentation of some digital objects

Object	Object size		# Slices			CPU time (in secs.)		
	# Vertices	# Faces	<i>yz</i> -plane	<i>zx</i> -plane	<i>xy</i> -plane	Slicing	Segmentation	Total
<i>g</i> = 2								
Dog	14862	301435	30	105	82	0.036	0.126	0.162
Octopus	16253	346467	95	98	156	0.062	0.405	0.467
Human	7093	170547	75	93	18	0.020	0.100	0.120
Chair	12787	228913	59	135	46	0.045	0.132	0.177
<i>g</i> = 4								
Dog	14862	301435	16	53	39	0.007	0.026	0.033
Octopus	16253	346467	48	46	70	0.011	0.055	0.066
Human	7093	170547	37	45	8	0.005	0.019	0.024
Chair	12787	228913	29	66	22	0.008	0.024	0.032
<i>g</i> = 6								
Dog	14862	301435	10	34	26	0.004	0.014	0.018
Octopus	16253	346467	28	27	45	0.005	0.018	0.023
Human	7093	170547	25	30	5	0.003	0.011	0.014
Chair	12787	228913	21	44	15	0.004	0.010	0.014

Fig. 8. Results on Chair, tilted (*g* = 4), coloring as in Fig. 4

as evident from Table 2. This conforms to our theoretical analysis on time complexity (Sec. 4.1). For instance, the surface areas for ‘Human’, ‘Dog’, and ‘Chair’ are 193, 337, and 390 units. The area ratios are, therefore, Human:Dog = 0.6, Human:Chair = 0.5, Chair:Dog = 1.18; the corresponding CPU time ratios for *g* = 2 are 0.75, 0.67, 1.09.

Acknowledgement. A part of this research is funded by CSIR, Govt. of India.

References

1. Agathos, A., et al.: 3D articulated object retrieval using a graph-based representation. *Vis. Comput.* 26(10), 1301–1319 (2010)
2. Attene, M., et al.: Mesh segmentation—A comparative study. In: *Proc. SMI 2006*, pp. 7–18 (2006)
3. Berretti, S., et al.: 3D Mesh Decomposition using Reeb Graphs. *Image Vision Comput.* 27(10), 1540–1554 (2009)
4. Chazelle, B., et al.: Strategies for Polyhedral Surface Decomposition: An Experimental Study. *Comput. Geom. Theory Appl.* 7, 327–342 (1997)
5. Chen, X., et al.: A Benchmark for 3D Mesh Segmentation. *ACM Trans. Graph.* 28(3), 73:1–73:12 (2009)
6. Cohen-Or, D., et al.: Consistent Mesh Partitioning and Skeletonization using the Shape Diameter Function. *The Visual Computer* 24, 249–259 (2008)
7. Dupas, A., Damiand, G.: First Results for 3D Image Segmentation with Topological Map. In: Coeurjolly, D., Sivignon, I., Tougne, L., Dupont, F. (eds.) *DGCI 2008*. LNCS, vol. 4992, pp. 507–518. Springer, Heidelberg (2008)
8. Dupas, A., Damiand, G., Lachaud, J.-O.: Multi-Label Simple Points Definition for 3D Images Digital Deformable Model. In: Brlek, S., Reutenauer, C., Provençal, X. (eds.) *DGCI 2009*. LNCS, vol. 5810, pp. 156–167. Springer, Heidelberg (2009)
9. de Goes, F., et al.: A Hierarchical Segmentation of Articulated Bodies. In: *Proc. SGP 2008*, pp. 1349–1356 (2008)
10. Golovinskiy, A., et al.: Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)* 27(145) (2008)
11. Ju, X., et al.: Automatic Segmentation of 3D Human Body Scans. In: *Proc. Int. Conf. Comp. Graphics & Img.*, pp. 239–244 (2000)
12. Karmakar, N., Biswas, A., Bhowmick, P.: Fast slicing of orthogonal covers using DCEL. In: Barneva, R.P., Brimkov, V.E., Aggarwal, J.K. (eds.) *IWCIA 2012*. LNCS, vol. 7655, pp. 16–30. Springer, Heidelberg (2012)
13. Katz, S., et al.: Mesh segmentation using feature point and core extraction. *The Visual Computer* 21, 649–658 (2005)
14. Mangan, A., et al.: Partitioning 3D surface meshes using watershed segmentation. *IEEE TVCG* 5, 308–321 (1999)
15. Nurre, J.H.: Locating Landmarks on Human Body Scan Data. In: *Proc. 3DIM*, pp. 289–295 (1997)
16. Werghi, N.: A robust approach for constructing a graph representation of articulated and tubular-like objects from 3D scattered data. *PRL* 27(6), 643–651 (2006)
17. Xiao, Y., et al.: A Discrete Reeb Graph approach for the Segmentation of Human Body Scans. In: *Proc. 3DIM*, pp. 378–385 (2003)
18. Zhang, X., et al.: Salient Object Detection through Over-Segmentation. In: *Proc. ICME 2012*, pp. 1033–1038 (2012)
19. Zhu, N., et al.: Graph-Based Optimization with Tubularity Markov Tree for 3D Vessel Segmentation. In: *Proc. CVPR 2013*, pp. 2219–2226 (2013)

Appendix

Fig. 9. Segmentation results on Chair rotated from 10°, 20°, ..., 90° demonstrating rotational invariance. Segments are colored in their order of detection. Segment 1: Left fore leg, 2: Right fore leg, 3: Left hind leg, 4: Right hind leg, 5: Back rest, and 6: Seat.

Fig. 10. Segmentation results on Airplane ($g = 2$) for threshold varying with $\rho = 0.4, 0.5, 0.6, 0.7$ (from top-left to bottom-right)

Fig. 11. Segmented articulated portions and central region of Bird, Dolphin, Teddy, Dinosaur, Ant, and Human, at $g = 2$

Fig. 12. Segmentation results on Dog ($g = 2$) for threshold varying with $\rho = 0.3, 0.4, 0.5, 0.6$ (from top-left to bottom-right)