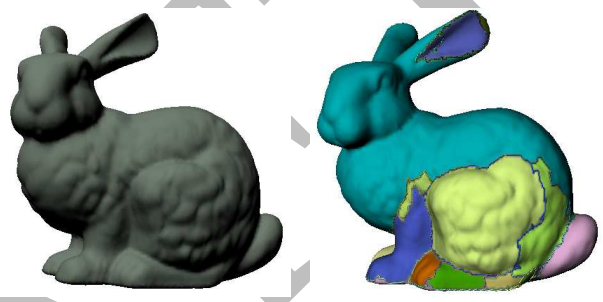
Ridge Walking for 3D Surface Segmentation

Andrew Willis, Beibei Zhou
Department of Electrical and Computer Engineering
University of North Carolina at Charlotte
Charlotte, NC 28223
arwillis@uncc.edu

Abstract

This paper describes a new 3D surface segmentation algorithm that separates a closed surfaces into regions by computing surface contours that traverse ridges of high-curvature surface regions. We refer to the approach as ridge-walking *since these contours tend to follow convex ridge-like structures and concave valley-like substructures present within the geometry of the model. Segmentation is achieved by solving for closed ridge contours on the surface, each of which serves to divide the surface into two disjoint regions. A collection of such curves then provides a segmentation of the surface into surface patches which is the segmentation result. The proposed approach also introduces a new parametrization for 3D models in term of the mesh vertices and edges that is a significant departure from standard methods that parametrize the surface based on mesh polygons. Solving directly for the boundary allows for direct control of the segmentation solution in ways that cannot be easily controlled with polygon-based parametrization. This can include control of the boundary length, shape, and, most importantly, the overall salience of the region boundary as defined by an optimization function such as the minima rule, which seeks to place boundaries along concave surface regions. We detail the segmentation algorithm, the surface parametrization use to develop the parametrization and provide analysis of several segmentation results for standard test surfaces from the literature in this area. Segmentation is an important surface analysis step needed when processing both synthetic (CAD-generated) and real-world (via 3D scanning) models which are commonplace in many contexts today. Applications of segmentation include surface compression, object recognition, texture mapping, surface re-parametrization, animation, collision detection and reverse engineering.*

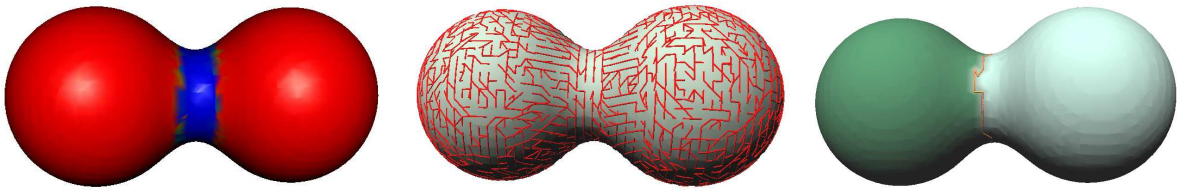


(a) (b)

Figure 1: We propose a new method to parametrize surfaces based on their ridges and apply this parametrization for the purposes of 3D model segmentation which takes a 3D model as input (a) and provides a decomposition of the model surface into meaningful parts shown in (b) as surface patches having distinct colors.

1. Introduction

The prevalence of 3D surface modeling and 3D capture technologies using laser scanners has made it much easier to generate complex models either manually via surface modeling packages or by capturing the shape of real-world objects using technologies such as 3D laser scanning. In order to work with these models, there is often a need to decompose a single complex model into separate pieces where each piece has simplistic geometry (see Fig. 1). Most methods make use of the *minima rule* which states that human perception usually divides a surface into parts along concave portions of the surface, i.e., we perceive concavities more sensitively as appropriate regions for separating a surface into parts. Scale also plays an important role, i.e., small-scale surface fluctuations are typically ignored for preference of large scale protrusions on object which we tend to separate along a concave contour close to the base of the protrusion (see Fig. 2(a)). Segmentation is an important surface analysis step needed when process-



(a) (b) (c)

Figure 2: (a) a "dumbbell" surface colorized by its curvature. (b) edges of the ridge-tree are shown as a collection of edges superimposed on the surface. The ridge-tree is computed by solving for the spanning tree of a graph defined where nodes are mesh vertices and graph edges are the edges of the mesh and the edge weights measure how "ridge-like" the surface is in the direction of the mesh edge. A segmentation of the surface obtained by by forming cycles in the ridge tree by adding mesh edges referred to as "loopy edges", that are any edges not included in the spanning tree. Addition of a loopy edge creates a cycle in the tree, or equivalently, a simple closed contour on the surface that divides the surface into two parts. (c) shows a simple surface segmentation obtained by inserting a single loopy edge the resulting contour is shown in pink and the two parts are shown in green (left) and cyan (right).

ing both synthetic (CAD-generated) and real-world (via 3D scanning) models which are commonplace in many contexts today. Applications of segmentation include surface compression, object recognition, texture mapping, surface re-parametrization, animation, collision detection and reverse engineering. While many existing algorithms today focus on segmentation of CAD models which has implications for man-made surface models which often have almost no surface noise, our approach is intended for use on surfaces generated from 3D sensors such as a 3D LIDAR or striped-light scanner, 3D structured light system, or surfaces generated from images via computer vision techniques such as photometric (shape-from-shading) reconstruction or multi-view stereo.

2. Related Work

The breadth of application for this topic has made it a consistent topic of interest for the research community for over 2 decades and a recent survey in [1] and [2] provide comparative analysis for leading contemporary methods. These methods are often analogous to well-established methods for image segmentation which is not surprising given the fact that an image may be considered to be a function graph having intensities defined over spatial (x, y) coordinates; $i = f(x, y)$. This is particularly true for range images that may be represented in the same way ($z = f(x, y)$) and many early works on this topic are targeted for segmenting range images [3, 4]. One popular approach that has enjoyed long-lasting popularity is segmentation via region-growing [5, 6] which proceeds by selecting (at random) polygons on the surface as a seed to a region and subsequently merges neighboring polygons into the seeded region according to a compatibility criterion, e.g., the dihedral angle is less than a threshold. Region growth stops when all neighboring polygons do not satisfy the compatibility criterion. The success of such methods continues to grow with recent variants [7] which includes enhancements such as multi-scale surface-smoothing, user-interactivity, and a stochastic filter to propagate growth more rapidly in flat areas than in regions of complex structure. [8] applies the well-known watershed segmentation approach for images to 3D surface models. Graph cut methods based on [9] have also been used to segment 3D surfaces automatically [10, 11] and with user assistance [12]. Other methods for segmentation include clustering surface regions based on common feature values computed over local surface regions which are typically geometric [13] but others have used spectral components [14]. Landmark-based segmentation approaches divide surfaces using paths that segment protrusions of objects away from the remainder of the model using landmarks called critical points which are deemed to be visually salient on the object [15]. A common tie to all of these methods is that they seek to cluster together polygons, i.e., surface elements, of the mesh. Contiguous groups of polygons assigned to a common cluster then correspond to a segmented patch from the surface. To our knowledge, the "mesh scissoring" approach from [16] represents the only contemporary approach that considers solving for *surface contours* that divide the surface into distinct regions. In this work, the authors generalize the snake model of [17] such that the boundary curves that divide the surface into parts propagate into concave surface regions. As known for snake models in general and explicitly noted in [1], such methods can sometimes converge to local minima of the curve-evolution performance functional which can result an undesirable surface segmentation.

Our algorithm is related to [16] since we are directly solving for contours on the surface that satisfy a salience criteria. Solving for the boundary curve directly has two

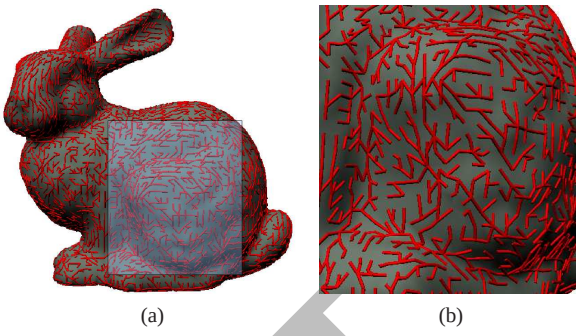
key benefits: (1) one can guarantee that the boundary of segmented surface regions will satisfy specific salience criterion; a constraint that is very difficult to enforce for surface element clustering methods and (2) one can directly impose constraints on the form of the segmented surface patch boundaries. Important constraints typically include differential properties of the boundary such as total length, smoothness, and curvature. Control of these properties of the boundary can suppress segmentation results that include small regions, improve on the regularity of the extracted surface patch boundaries, and prevent boundaries from meandering through regions of low curvature.

Yet, our model is also related to the graph-cut methods of [10, 11] since the contours which we compute come from a graph defined over the points and edges of the polygonal model. Typical graph cut methods define a graph based on the dual-mesh where graph nodes correspond to mesh polygons and graph edges exist between those polygons that share a common edge in the 3D model. In contrast, our graph is defined over the points and edges of the mesh with graph nodes being the points of the mesh and edges of the graph correspond to actual edges of the 3D mesh model (see Fig. 2(b)). By solving directly for the boundary of the surface region new capabilities are provided that allow one to control the form of the segmentation solution in ways that cannot be easily controlled with polygon-based parametrization (see Fig. 2(c)). This can include control of the boundary length, shape, and, most importantly, the overall salience of the region boundary as defined by an optimization function such as the minima rule, which seeks to place boundaries along concave surface regions.

3. Methodology

Our approach for segmentation seeks to find a parametrization of the surface in terms of the mesh edges where there is a unique path along the parametrization between any two points on the surface. As with other graph based methods, [10, 11], we define a graph over the polygonal mesh where graph edges are the mesh edges and graph nodes are the mesh vertices (note this is a significant departure from standard approaches see § 2 for details). Let $\mathbf{e}_{ij}$ denote the graph edge connecting nodes $\mathbf{p}_i$ and $\mathbf{p}_j$ and define $E = \{\cup_{ij}\mathbf{e}_{ij}\}$ as the set of all graph edges and $P = \{\cup_i \mathbf{p}_i\}$ as the set of all graph nodes. The graph $G(E, P)$ is then an equivalent representation for the mesh and is the object of computation for our segmentation approach.

As with other graph cut methods, a weight is associated with each edge of the graph. We denote this weight as a *salience measure* $w(\mathbf{e}_{ij})$ which provides a measure of how salient the edge is for some segmentation goal. We propose a sequence of potential salience measures based upon the principal directions of the surface and the minimum and maximum curvatures observed along these directions. For each surface point, we estimate the directions of principal curvature as two vectors, $\mathbf{v}_i$ and $\mathbf{u}_i$, in the local surface tangent plane and we denote the curvature of the surface in these directions as $(\kappa_{max}, \kappa_{min})_i$ respectively. *Salience functions* specify the weight of an edge and incorporate two criterion: (1) the curvature of the surface in the direction perpendicular to the edge, and (2) how well the edge aligns with the direction of minimum curvature. This is accomplished by approximating the direction of principal curvature at the midpoint of the edge. We utilize three different salience functions as provided in equations (1, 2, and 3)



(a) (b)

Figure 3: (a) shows a view of the spanning tree defined over the vertices and edges of the surface mesh. (b) shows a zoomed view of the surface region from (a) highlighted in blue. Here one can see that branches of the computed tree tend to follow ridge sub-structures on the surface and serve to parametrize the surface in terms of these ridge structures which are known to be perceptually important cues for meaningful surface segmentation.

$$w_{ridge}(\mathbf{e}_{ij}) = \frac{\mathbf{u}_i + \mathbf{u}_j}{\|\mathbf{u}_i + \mathbf{u}_j\|} \cdot \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|} \overline{\kappa}_{max} \tag{1}$$

$$w_{curv}(\mathbf{e}_{ij}) = \frac{\mathbf{u}_i + \mathbf{u}_j}{\|\mathbf{u}_i + \mathbf{u}_j\|} \cdot \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|} \max\left(|\overline{\kappa}_{max}|, |\overline{\kappa}_{min}|\right) \tag{2}$$

$$w_{valley}(\mathbf{e}_{ij}) = \frac{\mathbf{u}_i + \mathbf{u}_j}{\|\mathbf{u}_i + \mathbf{u}_j\|} \cdot \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|} (-\overline{\kappa}_{min}) \tag{3}$$

where the quantity $\frac{\mathbf{u}_i+\mathbf{u}_j}{\|\mathbf{u}_i+\mathbf{u}_j\|} \cdot \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|}$ represents how well the direction of the edge agrees with the estimated direction of the ridge-line on the surface (this is the direction of minimum curvature for convex regions and the direction of maximum curvature for concave regions). The values $\overline{\kappa}_{max} = \frac{(\kappa_{max})_i + (\kappa_{max})_j}{2}$ and $\overline{\kappa}_{min} = \frac{(\kappa_{min})_i + (\kappa_{min})_j}{2}$ are approximations for the principal curvatures averaged over the extent of the edge $\mathbf{e}_{ij}$.

The complete algorithm is provided below:

1. Compute a weight for each edge using a prescribed salience function such as (1,2 or 3).

2. Compute the (maximum) spanning tree of the graph (see Fig. 3). Denote $S = \{\cup_{ij} \mathbf{s}_{ij}\}$ as the set of edges that jointly define the spanning tree of the surface and $L = \{\cup_{ij} \mathbf{l}_{ij}\}$ as the set of all remaining edges which we refer to as "loopy edges."

3. As part of (2) we compute weights for the loopy edges and we store these loopy edges in a stack sorted in decreasing order based on the values of the chosen salience function. A large proportion of these edges are discarded and a small number of loopy edges, those having the largest weights, are kept to form closed contours on the surface.

4. Loopy edges are added into the mesh, each edge creates a new closed contour on the surface. Each time an edge is added, a uniqueness criterion and a curve-length criterion are used to suppress contours that are nearly equivalent or have very short arc-length.

5. The surface segmentation is computed by finding all contiguous surface regions within the contours created in step 4.

Step (2) makes use of standard textbook algorithms to compute the spanning tree of the surface, e.g., Prim's Algorithm or Kruskal's Algorithm. The computational challenge to step (3) is to find the closed surface contour created by adding a cycle to the spanning tree of the graph. This can be efficiently computed using the structure inherent to trees, e.g., a data structure having parent-child relationships. Loopy edges necessarily connect two leaves of the spanning tree and by tracing the ancestry (parents) of the two leaves we eventually find a common ancestor. The tree nodes traversed then constitute the edges and vertices of the surface contour (including the common ancestor). Step (4) is used to eliminate contours that are undesirable for our segmentation goal. In theory, this aspect of the algorithm could enforce arbitrary constraints on the boundary such as its global shape or smoothness. In our case, we implement two constraints that segmentation boundary contours must satisfy: (1) each contour must be distinct from other extracted surface contours and (2) each contour must be of sufficient length (at least 5% of the longest extracted curve loop in the segmentation). Details for this step are provided in § 3.1. Step (5) is implemented by merging all contours into a single curve which may include many self-intersections. All simple closed loops of the resulting curve are then extracted. Loop extraction is done by choosing a random point and direction on the merged contour and then traversing the contour such that, at each intersection point one follows either a clockwise or anti-clockwise direction

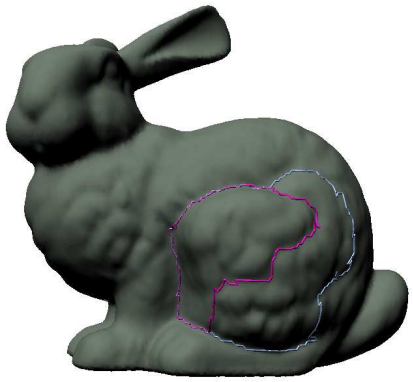


Figure 4: Two partially overlapping contours on the surface are shown in pink and in cyan. Boundaries such as this may be merged as part of the post-processing of the segmentation. In this case, the pink that divided the thigh into two pieces was discarded in preference for the larger cyan contour. This decision is based on the relative value of edge weights found in the non-overlapping regions of the competing contours.

until you arrive at the point originally chosen. This process continues until all edges of the merged contour have been traversed in both the clockwise and anti-clockwise directions. Note that this can be viewed as an application of the "left-turn algorithm" used commonly to automatically solve 2-dimensional mazes. Contiguous surface regions attached to the extracted simple closed contours is trivially computed given the graph and the boundary values.

3.1. Enforcing Contour Constraints

Many segmentation approaches incorporate post-processing steps that filter the initial segmentation to prevent over-segmentation which can occur when many small surface regions are detected as distinct and to merge similar regions that share a boundary yet were classified as distinct. Our segmentation algorithm also includes a post-processing step whose purpose is to enforce constraints on the extracted surface contours. For the results of this article, two key filters were implemented: (1) a filter to eliminate short contours and (2) a filter to eliminate similar contours.

The first constraint is trivial to enforce from knowledge of the points and edges of the contour. We use a threshold that requires all contours have be at least 5% of the length of the longest extracted contour. Since all contours are constrained to follow the tree (except for the loop edge), similar contours are often created by adding loopy edges that have close proximity on the surface. Our contour similarity filter evaluates the salience difference between two contours and merges regions delimited by contours that are do not satisfy

a salience criterion. Since both contours are computed from the spanning tree, at some point they must share a common ancestor or ancestor(s) which denote geometric regions where these contours overlap. The average salience (as determined by the salience function) is found for the contour pair, $\overline{w}$, and the salience of their non-overlapping subsets. If the salience of the non-overlapping part of a contour is less than the average salience of the entire contour, the contour segment it is removed which is equivalent to merging the two regions delimited by the removed contour part.

Specifically, the criteria for merging edges is based on comparing the average edge weight for non-overlapping parts of the two intersecting loops. Let us assume the two contours are generated by adding loopy edges l_{ab} and l_{mn} and we wish to compute the relative salience of their non-overlapping parts for potential merging of the regions enclosed by these contours. The following steps detail our method for merging contours: (*for the sake of simplicity, we refer to each contour by the loopy edge used to generate the contour*):

1. Compute the average salience, $\overline{w}_{l_{ab}}$ and $\overline{w}_{l_{mn}}$, for each contour:
$$\overline{w_{l_{ab}}} = \frac{1}{N} \sum_{\mathbf{e}_{jk} \in l_{ab}} w(\mathbf{e}_{jk})$$
where, N denotes the number of edges in loop created by inserting loopy edge l_{ab}.

2. Compute the global average edge weight, $\overline{w}$, for the contour pair:
$$\overline{w} = \frac{1}{2}(\overline{w}_{l_{ab}} + \overline{w}_{l_{mn}})$$

3. For each pair of intersecting loops (l_{ab}, l_{mn}), compute the average edge weight for non-overlapping part of each loop
$$\overline{w'_{l_{ab}}} = \frac{1}{K_{ab}} \sum_{e_{jk} \in l_{ab}, e_{jk} \notin l_{mn}} w(\mathbf{e}_{jk})$$
$$\overline{w'_{l_{mn}}} = \frac{1}{K_{mn}} \sum_{\mathbf{e}_{jk} \notin l_{mn}, \mathbf{e}_{jk} \in l_{mn}} w(\mathbf{e}_{jk})$$
where, K_{ab} is the number of edges in loop l_{ab} are not part of loop l_{mn} andK_{mn} is the number of edges in loop l_{mn} that do not belong to loop l_{ab}.

4. Merge the loops by comparing the average edge weight for non-overlapping part of two loops (l_{ab}, l_{mn}); if $\overline{w'_{l_{ab}}} > \overline{w}$ and $\overline{w'_{l_{mn}}} > \overline{w}$, keep both loops otherwise, keep the loop with larger average edge weight and discard the other contour.

Step (2) from the overall algorithm uses loopy edges having high weights for constructing contours. Many of these contours can (and often do) pass through almost identical surface regions which makes the filtering steps discussed in this section a necessity to prevent over-segmentation.

4. Results

Fig. 5 shows segmentation results for three different surfaces obtained via laser scanning. Each row of the figure shows segmentation results for a different surface model and each column shows segmentation using a different salience criteria. The left column shows segmentation results for ridge-based segmentation as defined in equation (1), the middle column shows segmentation results for concave regions as defined in equation (3), and the right column shows segmentation results for curved regions, i.e., either ridges or valleys, as defined in equation (2). The first two surfaces are standard surfaces commonly used for presenting surface processing results and the third surface is a laser scan of a bone fragment from a replica of a human tibia. For the bunny model, the minima rule seems to work well for segmentation, i.e., the segmentation in both (b) and (c) seem perceptually satisfactory whereas (a) is not. For the horse model, segmentation results (d) and (f) seem perceptually satisfactory whereas (e) is not. For the bone model, segmentation results (g) and (i) seem satisfactory whereas (h) is not. Interestingly, the bone fragment represents a somewhat non-standard surface for analysis such as this since its surface is convex nearly everywhere. Hence, satisfactory segmentation of that surface relies on extraction of convex ridges where the opposite is true for the bunny model which is highly convex, i.e., sphere-like, and details are almost exclusively offered by concave surface regions. One can see that the solutions offered by taking the absolute value of the curvature shows less overall variability for these surfaces which may be desirable, especially in cases where one is attempting to recognize and object parts of the segmented model surface.

5. Conclusion

We have proposed a new parametrization of a surface in term of a ridge-tree which is a spanning tree computed from curvature data estimated at the model vertices. Tree nodes and edges correspond to the nodes and edges of the 3D mesh under analysis. Different salience functions can be used to make the segmentation approach sensitive to specific kinds of surface sub-structures which can help effectively segment difficult or unusual surfaces such as the bone fragment shown in Fig. 5(g,h,i). As with other segmentation algorithms, the output does not always produce geometrically meaningful parts as separate components. Surprisingly, segmentation using the absolute value of the cur-

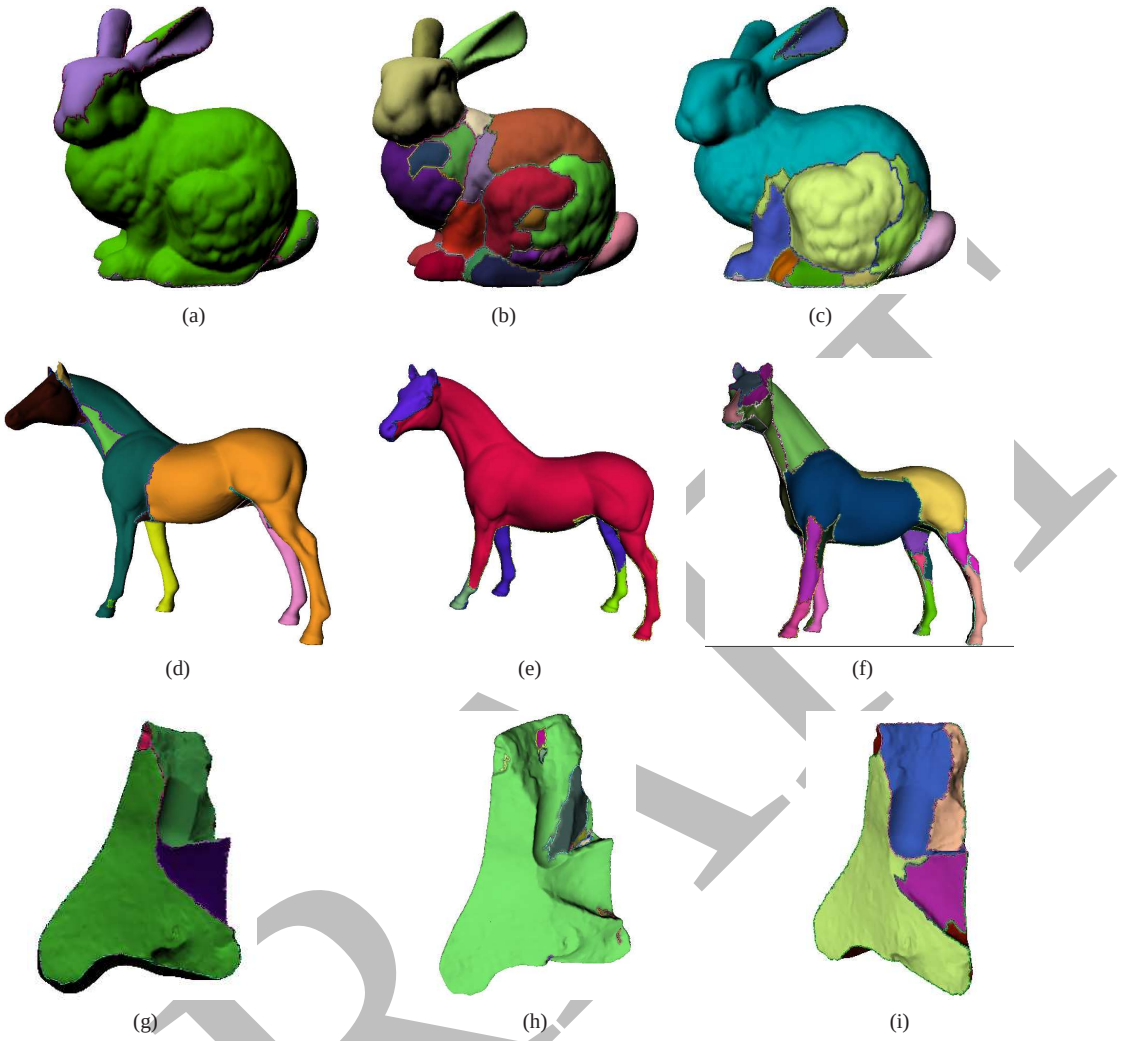


Figure 5: Results are shown that segment three different surfaces using three discussed salience criteria. (left column) segmentation by convex ridges, (middle column) segmentation by concave valleys, (right column) segmentation by ridges and valleys. Different salience functions as defined by equations (1) (2) and (3) generate optimization criterion for the segmentation that can be made are sensitive to specific surface sub-structures which can generate acceptable results as in (a,c,d,f,g,i) and also poor results as in (b,e,h). A mixed segmentation model based on absolute curvature, i.e., equation (2), seems to be most reliable across the surfaces shown here.

vature seems to provide stable results for surfaces that may be difficult by straight-forward application of the minima rule. For detailed results comparison of our segmentation results with other leading algorithms similar to that in [18] is needed. We would also like to attempt to apply some of the newly defined benchmarking techniques as mentioned in [19] that may provide methods to objectively quantify the quality of the generated 3D segmentation results which remains a challenge in the 3D graphics community.

References

[1] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis, "3d mesh segmentation methodologies for cad applications," *Computer Aided Design & Applications*, vol. 4, no. 6, pp. 827–841, 2007. 2

[2] A. Shamir, "Segmentation and shape extraction of 3d boundary meshes," in *In State-of-the-Art Report (STAR), Proceedings Eurographics*, 2006, pp. 137–149. 2

[3] P. J. Besl and R. C. Jain, "Segmentation through

variable-order surface fitting," *IEEE Transaction on pattern analysis and machine intelligence*, vol. 10, no. 2, pp. 167–192, 1988. 2

[4] R. Hoffman and A. K. Jain, "Segmentation and classification of range images," *IEEE Transaction on pattern analysis and machine intelligence*, no. 5, pp. 608–620, 1987. 2

[5] K. DongHwan, Y. TiDong, and L. SangUK, "Triangular mesh segmentation based on surface normal," in *Proceedings of ACCV2002*, 2002, pp. 23–25. 2

[6] M. Vieira and K. Shimada, "Surface mesh segmentation and smooth surface extraction through region growing," *Comput. Aided Geom. Des.*, vol. 22, no. 8, pp. 771–792, 2005. 2

[7] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin, "Rapid and effective segmentation of 3d models using random walks," *Comput. Aided Geom. Des.*, vol. 26, no. 6, pp. 665–679, 2009. 2

[8] A. P. Mangan and R. T. Whitaker, "Partitioning 3d surface meshes using watershed segmentation," *IEEE Transaction on visualization and computer graphics*, vol. 5, no. 4, pp. 308–321, 1999. 2

[9] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, 2000. 2

[10] M. Pauly, N. J. Mitra, J. Giesen, M. Gross, and L. J. Guibas, "Example-based 3d scan completion," in *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2005, p. 23. 2, 3

[11] A. Golovinskiy and T. Funkhouser, "Randomized cuts for 3d mesh analysis," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–12, 2008. 2, 3

[12] A. Sharf, M. Blumenkrants, A. Shamir, and D. Cohen-Or, "Snappaste: an interactive technique for easy mesh composition," *Vis. Comput.*, vol. 22, no. 9, pp. 835–844, 2006. 2

[13] N. Gelfand and L. J. Guibas, "Shape segmentation using local slippage analysis," in *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. New York, NY, USA: ACM, 2004, pp. 214–223. 2

[14] R. Liu and H. Zhang, "Segmentation of 3d meshes through spectral clustering," in *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 298–305. 2

[15] H.-Y. S. Lin, H.-Y. Liao, and J.-C. Lin, "Visual salience-guided mesh decomposition," in *Multimedia Signal Processing, 2004 IEEE 6th Workshop on*, Sept.-1 Oct. 2004, pp. 331–334. 2

[16] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel, "Mesh scissoring with minima rule and part salience," *Comput. Aided Geom. Des.*, vol. 22, no. 5, pp. 444–465, 2005. 2

[17] C. Xu and J. Prince, "Snakes, shapes, and gradient vector flow," *Image Processing, IEEE Transactions on*, vol. 7, no. 3, pp. 359–369, Mar 1998. 2

[18] M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, and A. Tal, "Mesh segmentation - a comparative study," in *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*. Washington, DC, USA: IEEE Computer Society, 2006, p. 7. 6

[19] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3d mesh segmentation," in *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*. New York, NY, USA: ACM, 2009, pp. 1–12. 6