Surface Mesh Segmentation using Local Geometry

Chansophea Chuon
Computer Science and Information Management
Asian Institute of Technology
Klong Luang, Pathumthani 12120, THAILAND
chansophea.chuon@ait.asia

Sumanta Guha
Computer Science and Information Management
Asian Institute of Technology
Klong Luang, Pathumthani 12120, THAILAND
guha@ait.asia

***Abstract*—We present a novel algorithm to segment a 3D surface mesh into visually meaningful regions. Our approach is based on an analysis of the local geometry of vertices. In particular, we begin with a novel characterization of vertices as convex, concave or hyperbolic based upon their discrete local geometry. Hyperbolic and concave vertices are considered potential feature region boundaries. We propose a new region growing technique starting from these boundary vertices leading to a segmentation of the surface that is subsequently simplified by a region-merging method. Experiments indicate that our algorithm segments a broad range of 3D models at a quality comparable to existing algorithms. Its use of methods that belong naturally to discretized surfaces and ease of implementation make it an appealing alternative in various applications.**

***Keywords*-Feature extraction; hyperbolic vertex; local geometry; mesh segmentation; region growing; region merging.**

I. Introduction

Polygonal meshes are the dominant mode of representing surfaces in computer graphics. They are geometrically simple and intuitive, and lend themselves to efficient algorithms and data structures. However, other than vertex-edge-face adjacency data, a polygonal mesh does not intrinsically possess any high-level semantic structure. For example, there is no information as such in the mesh data of the hand in Figure 1 that distinguishes the five fingers. However, such segmentation of an object into perceptually meaningful regions is important in applications such as shape recognition [19], morphing [7], texturing [9] and collision detection [10], amongst others.

The goal of surface mesh segmentation then is to decompose the input mesh into smaller regions that are perceptually significant. Nevertheless, as Attene et al [1] point out it is not possible to define exactly what constitutes perceptual significance. Not only does this lie “in the eyes of the beholder” it may vary from application to application. The viewer’s world knowledge is critical to segmentation as well – in a model that a layman would segment only into heart and lung and tissue, an oncologist may distinguish cancerous tumors and benign growths as well.

Our contribution is a novel approach to mesh segmentation that first separates vertices based on their local geometry. In particular, we distinguish between convex, concave and hyperbolic vertices by examining their geometric neighborhoods as a discrete structure. This characterization, which seems to be new in CG applications, is not derived from the well-known method of differentiating between hyperbolic, parabolic and elliptic points on a smooth surface using Gaussian curvature, though the two are not unrelated. Following the characterization of vertices, we estimate the potential (meaningful) region boundaries to be along the concave and hyperbolic vertices. Regions are subsequently grown from the boundary vertices by a novel modification of the watershed segmentation scheme. A final region merging step bounds the number of segments.

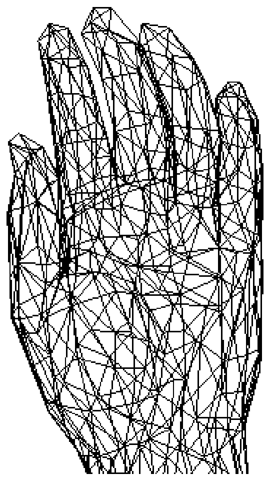
Figure 1. Hand model.

The rest of the paper is organized as follows. Section II briefly reviews existing algorithms that are relevant to ours. In Section III we discuss the background theory and our segmentation method. Section IV shows experimental results on various 3D models and Section V concludes the paper.

II. Related Work

An approach to segmentation that splits the surface into coherent nearly-flat patches based, typically, on a curvature analysis is called patch-type segmentation [6], [4], [16], [19]. See Figure 2. Patch-type segmentation is used in applications that are sensitive to geometric properties such as planarity and convexity. These include texture mapping, re-meshing and simplification [15].

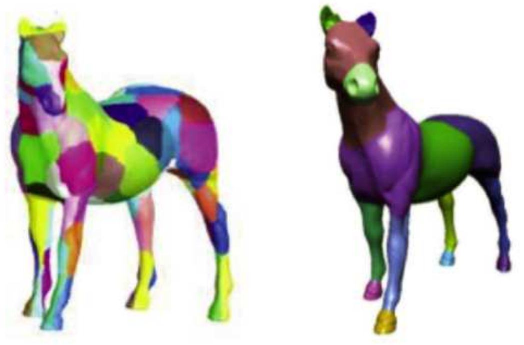

Figure 2. Patch-type (left) and part-type (right) segmentation taken from [15].

Part-type segmentation, on the other hand, seeks to subdivide the input mesh into sub-meshes that match human perception. Algorithms of this type can assist applications such as shape retrieval, shape recognition, collision detection and object manipulation.

Rom and Medioni [13] in 1994 proposed a framework to decompose 3D objects into parts using the curvature properties of parabolic curves. Though they initiate the use of curvature in segmentation their approach cannot directly process triangle meshes.

Sacchi et al [14] and Mangan and Whitaker [11] propose approaches to segmentation using curvature that can apply to a mesh object. They use total curvature as the key to segmenting parts from an input mesh. However, their segmentation of large models often fails to match viewer perception.

Page et al [12] in 2003 introduced a method called *Fast Marching Watersheds* that splits a triangle mesh into segments that fit the definition of visual parts according to the *minima rule* proposed by Hoffman and Richards [8]. The minima rule is an elegant theory from computer vision which propounds that human perception decomposes a 3D object into visual constituents at contours of negative Gaussian curvature – bands of hyperbolic points in other words.

Zhang et al [18] proposed a segmentation algorithm that estimates the Gaussian curvature at a vertex, identifies the hyperbolic vertices as potential region boundaries, applies the watershed technique to grow regions, and completes finally with a curvature-driven region merging phase. Recently, Chen and Georganas [3] improved the algorithm of Zhang et al. [18] by extending the boundary detection step to additionally identify concave vertices and prevent them from breaking segment integrity.

Our segmentation algorithm is inspired by that of Chen and Georganas. However, we do not invoke Gaussian curvature. Instead, we apply our new classification scheme to distinguish concave and hyperbolic vertices, which subsequently serve as region boundaries. Our region growing algorithm is a modification of the watershed technique that seems to better allow growth into intuitively separate parts.

III. Theory and Algorithm

A. Vertex classification

To simplify the theoretical development we assume that the input mesh is a topologically closed surface, implying that it has a meaningful interior. This assumption is not essential, however, and our code works for non-closed meshes as well. We assume that mesh faces are all triangular. We begin with a characterization of vertices as convex, concave or hyperbolic.

Definition 1. *A vertex V of a mesh M is hyperbolic if it is contained in the interior (as a subspace of $\mathbb{R}^3$) of the convex hull C of its neighbors. A vertex V that is not hyperbolic is convex if there exists a (flat) disc D centered at V which does not intersect the interior of M; otherwise, it is concave.*

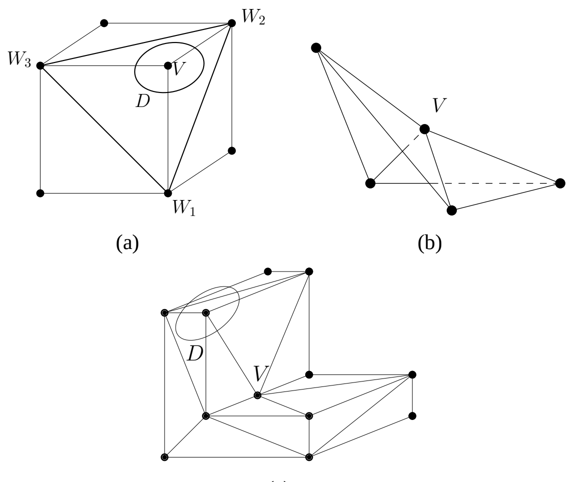

Figure 3. Illustrations of convex, hyperbolic and concave vertices.

For example, the convex hull of the neighbors of V in Figure 3(a) is the triangle $W_1W_2W_3$, which has empty interior in $\mathbb{R}^3$. Therefore, V is trivially non-hyperbolic; moreover, the disc D proves that V is convex. The vertex V on the saddle-shaped surface in Figure 3(b) is hyperbolic. The reader may check in Figure 3(c) that all the vertices on the L-shaped solid are non-hyperbolic, and that only vertex V is concave, while all the rest are convex. The disc D at a corner of the L-shaped solid indicates the reason why we cannot replace the disc in the definition of convexity with its containing plane – a plane may intersect the mesh at a distant point.

Our characterization of a mesh vertex is not unrelated to that of a point of a smooth surface as hyperbolic, parabolic or elliptic by means of Gaussian curvature, but we'll not explore the connection here. It should be observed though that our use of the discrete local geometry to classify vertices seems more natural for a meshed surface than computing pseudo-curvature values, as some authors do, on a smooth approximation.

Whether a vertex V is hyperbolic or not is a local decision depending on its disposition with respect to its neighbors. However, distinguishing a non-hyperbolic vertex V as either convex or concave necessitates one to determine the side of the surface near V that the interior of M lies, which requires global knowledge of M.

B. Region Growing

Our strategy to find features is to first presume hyperbolic and concave vertices as potential feature region boundaries. The motivation is simple. A surface whose vertices are all convex, e.g., the box of Figure 3(a), is likely featureless. On the other hand, the two legs of the L-surface of Figure 3(c) appear to be features that are separated by the edge at the crook between them – all vertices in the interior of this edge being concave. Vertices along the circular region where the stem meets the cap of the mushroom of Figure 4 appear to be hyperbolic.

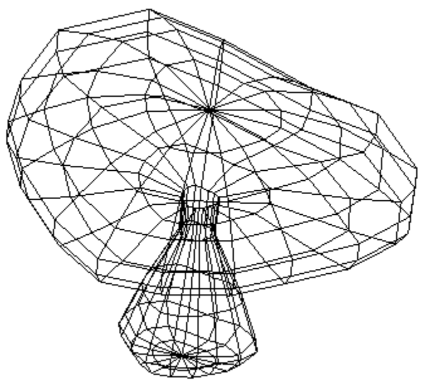

Figure 4. Mushroom model.

The next stage in our strategy is, therefore, to treat the hyperbolic and concave vertices as ”seeds”, from which to grow the feature regions. In particular, we shall grow the regions in a scattershot manner as the local geometry at a seed does not in general suggest a best direction. The plan is as follows. Seeds are initially all labeled differently. They are next processed iteratively to label their neighbors. Vertices adjacent to each seed, which have not already been labeled, are given new and different labels. In the following rounds vertices already labeled are processed iteratively to propagate their labels in a breadth-first manner.

For example, say the labels after the first round of the vertices $w_1, w_2, \ldots, w_k$, adjacent to a seed v are $l_1, l_2, \ldots, l_k$, respectively. Then, vertices adjacent to w_1, that have not already been labeled, are labeled l_1; then, those adjacent to w_2, that have not already been labeled, are labeled l_2; and so on. Figure 5 illustrates the idea. The process of growing feature regions in this manner from each seed is repeated iteratively until all the vertices of the mesh have been labeled.

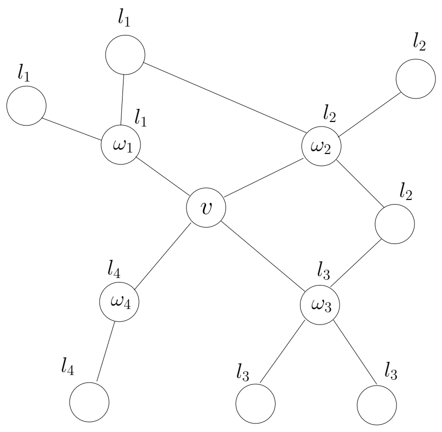

Figure 5. Region growing.

Our region growing method, though inspired by the watershed technique, is different in that the latter labels regions with the names of the seeds themselves, while our method adds one layer of growth in every possible direction. This idea suggested itself from experimentation with various models and seems to actually work better than the straight watershed scheme in practice.

C. Region Merging

Vertices having like labels form a feature region, but there is likely great over-segmentation at this point, so we begin a third stage of region merging. In this final stage, the feature region with the smallest number of vertices is iteratively merged with a neighboring region. We choose the neighboring region to merge with as the one – borrowing a heuristic from Chen and Georganas [3] – that shares the longest border with the candidate for merging. Figure 6 shows the idea. Region merging continues until a preset threshold is reached in the number of segments.

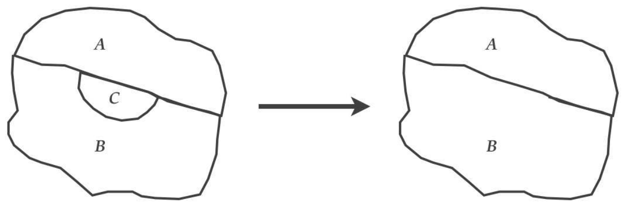

Figure 6. Region merging.

D. Complexity

Leaving out details in this version, it is still fairly straightforward to see that the complexity of our segmentation procedure is linear in the size of the mesh, because: (a) the classification of vertices as convex/concave/hyperbolic requires examination only of each vertex's link, and, moreover, each edge of a surface mesh can appear in the link of at most two vertices, for a total cost linear in the number of edges; (b) region growing is a breadth-first procedure requiring time linear in the number of vertices; (c) region merging costs $O(1)$ at most per vertex for a total linear cost as well.

IV. Experimental Results

We have implemented our segmentation algorithm, coding it in C++ with the STL on an Intel platform with 2Ghz CPU and 1GB RAM. We used several pre-packaged routines from the CGAL library [2], the Qt toolkit [17] to build the GUI, and the libQGLViewer [5] as a rendering engine.

Figures 7-11 show experimental results on fairly large meshes. The quality of the output compares well with that of existing methods.

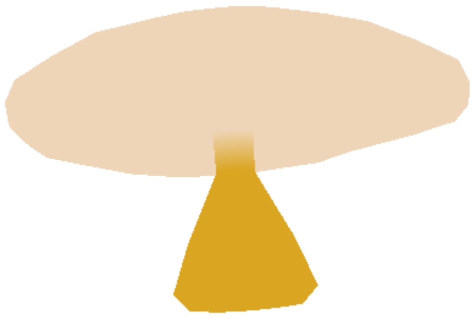
Figure 7. Mushroom model (240 triangles) segmented to 2 regions.

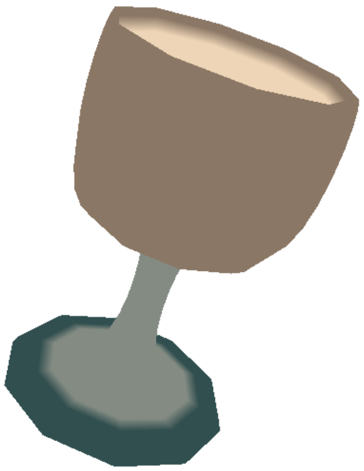
Figure 8. Goblet model (510 triangles) segmented to 4 regions.

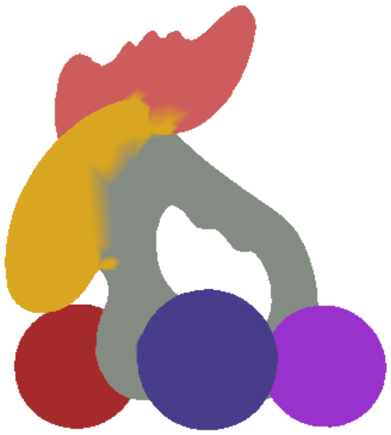
Figure 9. Elk model (10K triangles) segmented to 7 regions.

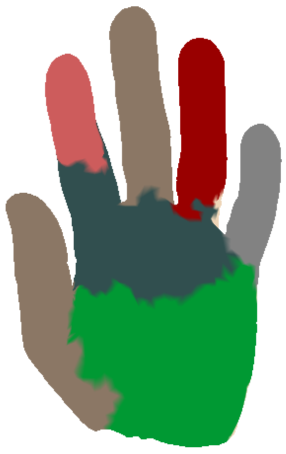
Figure 10. Hand model (50K triangles) segmented to 8 regions.

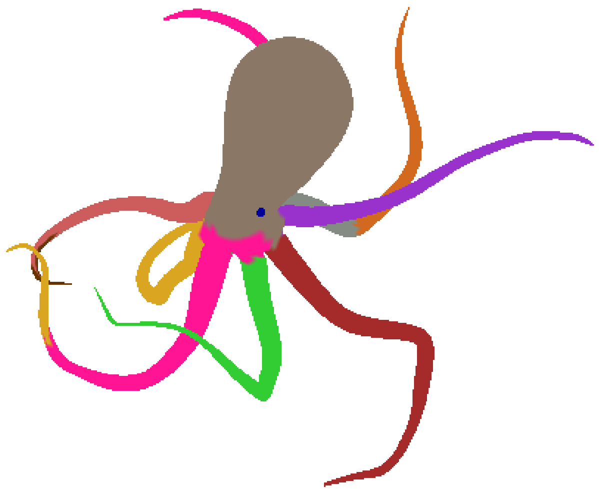
Figure 11. Octopus model (34K triangles) segmented to 15 regions.

V. Conclusions and Future work

We have proposed a novel algorithm for segmenting meshed surfaces into smaller visually meaningful parts. We apply a discrete and local characterization, itself new, of vertices into hyperbolic, convex and concave, to find region boundaries. This method seems a more natural fit for a mesh structure than the popular Gaussian curvature estimation method, the latter being a transplant from smooth surfaces. Our method of region growing is a novel modification as well of the watershed method.

There are specific improvements that can be made. For example, the region growing method is not robust and is sensitive to the initial labeling of the boundary vertices. We hope to address this problem by "inverting" the labeling procedure so that it starts from the non-boundary vertices, each such finding a nearest (according to an appropriate distance metric) boundary vertex and adopting the latter's label. We would also like to replace the user-set threshold for the number of regions at which the merging process stops, with automatic termination based upon some measure of the quality of the current segmentation. After these modifications are done, and experimental results satisfactory, we plan to make the software available freely.

We have as yet to make a systematic comparison of our algorithm with existing ones. We would like to do this following the schema of Attene et al [1]. Further, we intend to make at least an empirical observation of the processing time on large meshes.

Another important direction for future work that we wish to pursue is to extend the algorithm to process point cloud data, where there is no connectivity information.

Acknowledgements

We are grateful to the AIM@SHAPE Project for the mesh models that we used. We thank Nguyen Tan Khoa for initial contributions.

References

[1] 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 (SMI'06)*, page 7, Washington, DC, USA, 2006. IEEE Computer Society.

[2] http://www.cgal.org/.

[3] L. Chen and N. D. Georganas. An efficient and robust algorithm for 3d mesh segmentation. *Springer Science + Business Media, LLC*, 2006.

[4] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 905–914, New York, NY, USA, 2004. ACM Press.

[5] G. Debunnes. http://artis.imag.fr/˜Gilles.Debunne/ QGLViewer/.

[6] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 173–182, New York, NY, USA, 1995. ACM Press.

[7] A. D. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston. Interactive surface decomposition for polyhedral morphing. *The Visual Computer*, 15(9):453–470, 1999.

[8] D. D. Hoffman and W. Richards. Parts of recognition. Technical Report AIM-732, Massachusetts Institute of Technology - Artificial Intelligence Laboratory, 1983.

[9] B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371, 2002.

[10] X. Li, T. W. Toon, and Z. Huang. Decomposing polygon meshes for interactive applications. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 35–42, New York, NY, USA, 2001. ACM Press.

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

[12] D. L. Page, A. F. Koschan, and M. A. Abidi. Perception-based 3d triangle mesh segmentation using fast marching perception-based 3d triangle mesh segmentation using fast marching watersheds. *Proceedings of International Conference on Computer Vision and Pattern Recognition*, II:27–32, June 2003.

[13] H. Rom and G. Medioni. Part decomposition and description of 3d shapes. In *Pattern Recognition*, pages 629–632, 1994.

[14] R. Sacchi, J. F. Poliakoff, P. D. Thomas, and K. H. Hafele. Curvature estimation for segmentation of triangulated surfaces. In *Digital Imageing and Modeling*, pages 536–543, 1999.

[15] A. Shamir. A formulation of boundary mesh segmentation. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*, pages 82–89, Washington, DC, USA, 2004. IEEE Computer Society.

[16] A. Sheffer. Model simplification for meshing using face clustering. In *Computer Aided Desgin*, pages 925–934, 2001.

[17] TrollTech. Qt 4.2.2 community version http://www.trolltech.com.

[18] Y. Zhang, J. Paik, A. Koschan, M. A. Abidi, and D. Gorsich. A simple and efficient algorithm for part decomposition of 3-d triangulated models based on curvature analysis. *IEEE ICIP*, 2002.

[19] E. Zuckerberger, A. Tal, and S. Shlafman. Polyhedral surface decomposition with applications. In *Computer & Graphics*, pages 733 – 743, 2002.