# Surface Segmentation Using Morphological Watersheds

Alan P. Mangan Ross T. Whitaker*

Department of Electrical Engineering
University of Tennessee, Knoxville

## Abstract

This paper describes a 3D surface segmentation method that uses morphological watersheds. The use of watersheds is commonly used to segment images, and in this paper we describe a generalization of that concept to the task of segmenting three-dimensional surface meshes. While the image-segmentation algorithms operate on the gradient magnitude or some related edge measure, the surface-mesh algorithm uses surface curvature. The algorithm segments the surface into patches, where each patch has a relatively consistent curvature throughout and is bounded by areas of higher, or significantly different, curvature.

**Keywords:** Surface segmentation, watershed algorithm, surfaces, curvature-based methods.

## 1 Introduction

By 3D surface segmentation, we mean the process of dividing the set of nodes that comprise the surface mesh (or edges and faces) into a number of subsets, where each subset is also connected. The goal is to be able to take as input a *bucket of polygons* and to produce as output a decomposition of the surface into meaningful pieces. Such a mesh segmentation is useful for a variety of applications.

One application is mesh reduction. The goal of mesh reduction [1, 2] is to reduce the amount of data in a 3D mesh while maintaining, to a certain degree, the fidelity of the surface. Most mesh reduction algorithms operate on the data as a whole. However, as the reduction rates get very high, reduction algorithms tend to distort the shapes of objects by removing vertices that correspond to important features. We propose that using our segmentation method as a pre-processing step and operating on each region separately offers an advantage. With this technique, reduction is confined to regions where the curvature is relatively consistent throughout. This allows reduction within regions but prevents the reduction of vertices associated with edges and similar discontinuities indicated by the presence of high curvature.

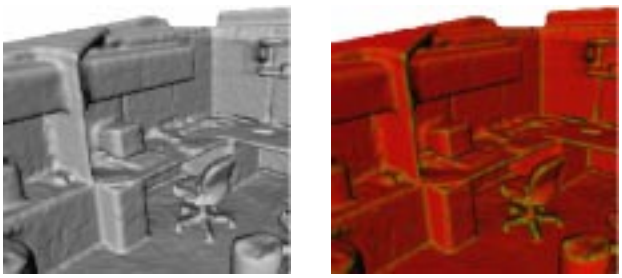
Because the proposed segmentation method breaks the surface into regions comprised of consistent curvature and bounded by high curvature, the geometry of the regions tends to be relatively homogeneous. Thus, as an alternative to conventional mesh reduction, one could decrease the complexity of the surface by fitting more powerful geometric primitives (e.g. splines or superquadrics) to the distinct patches produced by the segmentation.

*Texture mapping* is another technique that could profit from the application of such a 3D surface segmentation. Traditionally texture mapping for volumes utilizes a single texture map for the entire surface [3]. While this might work well for relatively simple volumes, such as solids of rotation, it does not address the issue of self-occlusion present in more complicated scenes. Creating an individual texture map for each segment of the surface rather than the surface as a whole promises to present a partial solution to this problem. Because each segment is bounded by high curvature regions do not extend around edges, thus avoiding self-occlusions for the most part. In the case of cylinders or similar consistently curved objects a single texture map can still be created using an approach along the lines of [3]. This allows for the creation of a texture-mapped scene based on an arbitrary 3D surface, whether self-occlusions are present or not.

The literature does present some approaches to surface segmentation [4]. However, these approaches are for the most part strictly geometric. For instance, many mesh partitioning methods strive to break the surface into convex patches. However, such methods do not attempt to apply any semantic meaning to the patches (e.g. hyperbolic regions are broken down into their constituent polygons). Also, such methods ignore the effects of noise or inaccuracies in the model the itself. For instance, planar regions, which occur regularly in geometric models, are rarely perfect planes.

The proposed method makes no assumptions about the underlying geometry or convexity of the surface in question, rather, the surface is segmented into *patches* bounded by regions where the total curvature is relatively high. The motivation is that regions of high curvature represent boundaries between distinct parts or faces. The surface patches generated by the watershed algorithm are of arbitrary convexity (or, topology for that matter); what they share is a common degree of curvature throughout.

The surface segmentation method presented in this paper is based on morphological watersheds, which are used in image processing to achieve sensible, reliable partitioning of images. For images the algorithm commonly operates on the gradient magnitude of the input image. For 3D surface meshes we use the total curvature of each vertex on the surface. For this results in this paper, the surface



(a) Surface model (b) Total Curvature

Figure 1: *The isosurface of a volume (a) and the total curvature at each vertex (b) (Red indicates low curvature, increasing through green and blue.) serve as inputs to the segmentation algorithm.*

*rtw@utk.edu

meshes are computed from volume data sets using marching cubes [5], and the curvature of the surface at the vertices is calculated directly from the volume data. The volumes were in turn created using the method described in [6]. Figure 1(a) shows one such surface mesh, and Figure 1(b) the total curvature of this surface. The watershed segmentation algorithm, described in Section 2, consists of two steps. The first step is the identification of distinct catchment basins. The second step is a region merging process, which combines insignificant regions (identified by the depth of the catchment basin), thereby making the results less sensitive to noise. We present results in Section 3, where we examine the performance of the algorithm in the presence of noise and present segmentations for several data sets.

# 2 The Watershed Algorithm

This section describes the strategy used in the watershed algorithm and the specifics of the implementation. The input to the system is a surface mesh where the total curvature at each vertex is known. Where the image processing version of the algorithm operates on a rectilinear 2D grid of points, we have extended the algorithm to a manifold in 3D, consisting of a mesh of connected vertices where each vertex also has associated with it a set of neighboring vertices. For this work, the mesh is extracted from a 3D volume and the mean, $H$, and Gaussian curvature, $K$, are found from [7]. The total curvature, $D$, is calculated from these using: $D^2 = 4H^2 - 2K^2$. Linear interpolation of volume derivatives are used to compute the curvatures at the vertex locations (which lie along grid lines). A lower threshold is then applied to the curvature as shown in Figure 2. This step is performed

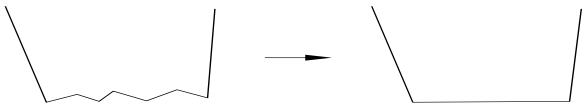


Figure 2: *Thresholding the total curvature.*

so that relatively flat areas where the curvature is extremely low throughout will be classified as exactly flat, easing the computational burden during later processing. Once this threshold has been applied the watershed algorithm proper can begin.

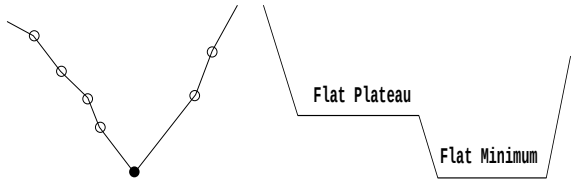
There are two different strategies for implementing the watershed algorithm. One strategy is to start at the bottom of a catchment basin and fill upward. The second strategy, which we use, is to track points downward to their associated minima. Thus, a brief overview of the algorithm is:

1. Locate and label all local minima, each of which forms the bottom of a catchment basin.

2. Flat regions with uniform curvature are then found, labeled, and classified depending on whether they have any neighboring vertices adjacent to their boundaries lower than their curvature or not.

3. Move a token (downward, steepest descent) starting from each remaining unlabeled vertex of the mesh until it encounters a labeled region.

4. Merge shallow regions with their neighbors until the depths of all remaining regions are above a preset threshold.




(a) Minima labeling. (b) Flat regions.

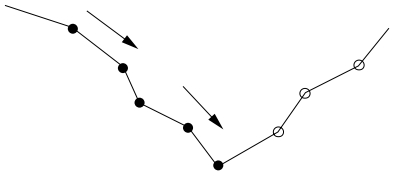
Figure 3: *Labeling of local minima and classification of flat regions.*

## 2.1 Initial Labeling

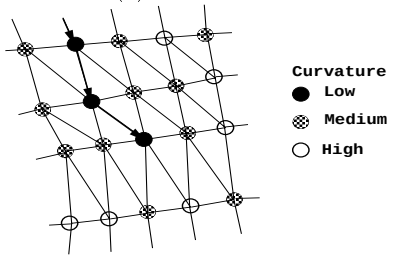
The watershed method first finds and labels all local minima, i.e. those vertices with curvature lower than that of all of their neighbors, as in Figure 3(a). Each minimum also serves as the initial seed for a surface region, i.e. a distinct region on the surface formed during the descent of vertices along their paths of steepest descent. Next, flat regions are found. These are one of two types as shown in Figure 3(b); either flat plateaus or flat minima; depending on whether they have any neighboring vertices on their borders lower than their curvature or not. Flat minima are labeled and treated in the same manner as local minima. After all flat regions are found a descent is made from each of the flat plateaus until a labeled region is encountered.

## 2.2 Descent

For flat plateaus the descent starts from the boundary vertex of the plateau whose neighbor, not of the region, has the lowest curvature. Imagine a drop of water placed at the starting vertex, flowing towards the point of lowest curvature. Each vertex it encounters on its path "downwards" is labeled with the same identifying label as the first labeled vertex it encounters, as shown in Figure 4. This drop will either flow all the way to a local (labeled) minimum, or hit a labeled vertex already associated with a minimum. Hence the name watershed. The plateau and its path of descent are than labeled and joined to the region finally hit during the descent. The final step in the initial labeling stage is then to allow all other unlabeled vertices to similarly descend until they hit a labeled region, then join them to that region.



(a) 1D descent.




(b) Descent along 3D mesh.

Figure 4: *Descent until labeled region encountered.*

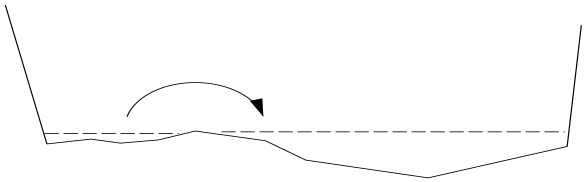


Figure 5: *Merging adjacent regions with shallow depths.*

### 2.3 Region Merging

After the final step above all of the surface vertices are labeled and assigned to a region. However, leaving the surface in this state can, due to small fluctuations in surface curvature, yield an over-segmented result. To alleviate this we merge the regions together in order to obtain reasonable results. This is done by combining with their neighbors those regions deemed insignificant (as in Figure 5). The saliency measure is watershed depth, which is the maximum height of the water that the region could contain without spilling over into an adjacent region. In order to proceed with this step several additional pieces of information are needed for each region. The vertex with the lowest curvature for the region is found, used to later estimate the depth of the region. The boundary of the region is found, and from this all neighboring regions are determined. For each neighbor $B$ the boundary vertex of the current region $A$ adjacent to $B$ with the lowest curvature is also found. From these boundary vertices the lowest (in terms of curvature) boundary vertex of the region is then calculated. With this information available region merging can then proceed.

The region merging is done based on the importance of a given region. The depth of the region is found and used as a measure of the region's significance. The depth is calculated as the difference between the lowest vertex in the region and the lowest vertex on the region's boundary. If this depth falls below some preset threshold then the region is merged with one of its neighbors. The neighbor adjacent to the lowest boundary vertex is selected as that one with which to merge. If this lowest boundary vertex happens to have two neighbors adjacent to it, then the neighbor with the lowest curvature at that point is chosen. For implementation purposes rather than re-label all of the region's vertices with the label of the neighboring region it has merged with, a pointer is set for the region indicating with which region it is merged. It also proves necessary to check all other regions merged with this one, and change their pointers to where they all indicate that they are all now merged with the same region.

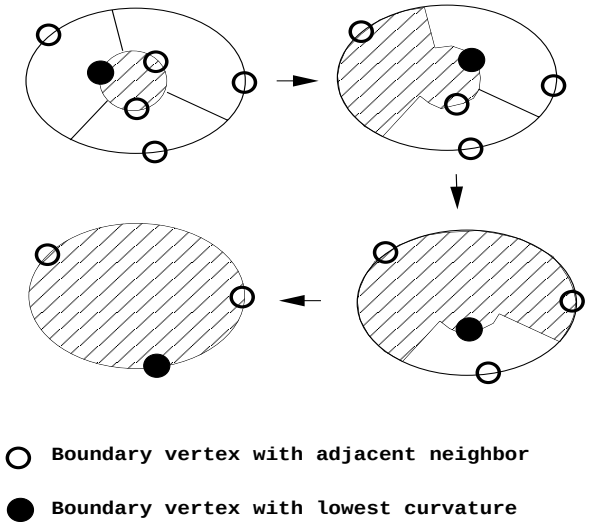



Figure 6: *Region merging. As each new region is added to the current region, indicated by the shaded area, its list of neighbors and associated boundary vertices needs to be updated accordingly.*

Once a region is joined with another it is necessary to update the information of this new region. This requires comparing and changing the lowest curvature vertex and adjusting the neighbor information of this new region. All of the current region's neighbors need to be added as neighbors of the new region, unless they themselves are merged with the new region. The current region, and all other regions which had been in turn merged with it, need to be then removed from the new region's list of neighbors. The boundary vertices associated with these neighbors also have to be removed from their respective arrays. The regions are merged in this manner according to Figure 6. Once this information has been updated, the new lowest boundary vertex has to be calculated and set. This merging procedure is iteratively applied to all regions on the surface until no further regions are within the preset depth threshold. At this point the watershed algorithm is complete.

## 3 Results

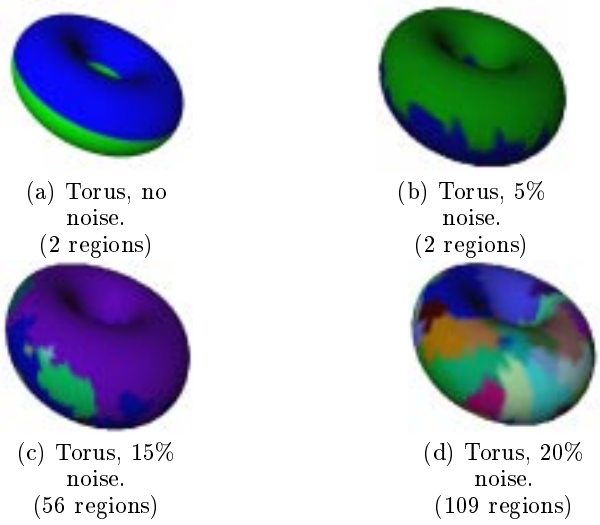
We have implemented the proposed watershed algorithm and used it to segment the surfaces of several volumetric data sets; simple shapes such as spheres and torii, surfaces with added noise, and a reconstruction of a real scene.

Results of using the watershed algorithm to segment the surface of a sphere can be seen in Figure 7. Figure 7(a) shows the results of using the method without any region merging. This serves to underscore again the necessity of region merging, as even a relatively homogeneous shape such as a uniform sphere gives rise to several hundred regions (348) when segmented without any merging. However, with the use of a small threshold when merging the majority of these regions are combined with their neighbors. Figure 7(b) shows the use of a threshold of 0.0002, yielding 6 regions. Using a threshold of 0.001 manages to segment the sphere into one region, the desired output.

(a) No region merging (348 regions). (b) Threshold = 0.0002 (6 regions). (c) Threshold = 0.001 (1 region).

Figure 7: *Segmentation of a sphere using the watershed algorithm.*

The performance of the algorithm in the presence of additive, uniform noise is displayed in Figure 8. Figure 8(a) is the segmentation of a torus without noise, and Figures 8(b)-(d) shows the torus with varying levels of additive noise applied to the curvature of each vertex. With 5% added noise the



(a) Torus, no noise. (2 regions)

(b) Torus, 5% noise. (2 regions)

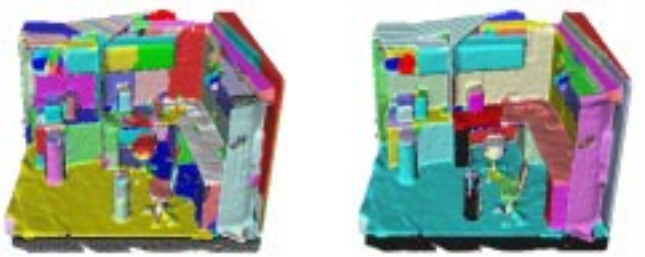
(c) Torus, 15% noise. (56 regions)

(d) Torus, 20% noise. (109 regions)

Figure 8: *Segmentation of a torus with different levels of additive noise applied.*

torus is still segmented into 2 regions, but the boundary between the regions is no longer as well defined. The torus has a relatively high degree of curvature throughout its surface, meaning that the addition of any amount of noise has detrimental effects upon the segmentation. 15% added noise severely affects the segmentation, the 2 main regions are still somewhat preserved, but many small additional regions are created around the former boundary. At 20% noise the segmentation has degraded completely and no longer corresponds to the underlying geometry.

Initial results of using the system on the sequence of data shown in Figure 1 with different threshold values are shown in Figure 9. The first segmentation in Figure 9(a) has the lowest threshold value, 0.15, and as expected the greatest number of regions, 626. Increasing the threshold to where fewer regions merge together, i.e. the criterion for the relative significance of each region is lowered, naturally leads to less regions. The method can be sensitive to the threshold level. Changing the threshold by as little as 0.05 leads to significantly fewer regions, and visibly alters the appearance of the final scene. The more the threshold is increased the more dramatic the final changes perceived in the scene. Segmentation results for additional volumetric data sets are presented in Figure 10.



(a) Threshold = 0.15 (626 regions).

(b) Threshold = 0.25 (306 regions).

Figure 9: *Segmentation of a scene using varying threshold levels.* 9640 *regions before merging.*

(a) Dart Segmentation.

(b) Jet Segmentation.

Figure 10: *Dart and jet Segmentations.*

## 4 Conclusions

We have presented here a method of surface segmentation which uses a generalization of morphological watersheds to 3D meshes. As demonstrated by the results, the method can segment isosurfaces of a 3D volume into regions that are bounded by dramatically differing curvature. Future work involves generalizing the technique to accept surface data in the form of a polygonal mesh and finding the curvature directly from the model. The algorithm also displays some sensitivity to the user-specified threshold. Varying the threshold allows for different levels of segmentation, the exact level depending upon the final application.

## Acknowledgments

Thanks go to the Computer Graphics Group at the California Institute of Technology for providing the jet and dart volume data. Thanks also to Samuel Burgiss Jr. for his assistance in preparing this paper. This work is supported by the Office of Naval Research grant N00014-97-0227.

## References

[1] Hugues Hoppe, "Progressive Meshes," *Computer Graphics (SIGGRAPH 96 Proceedings)*, pp. 99–108, July 1996.

[2] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen, "Decimation of Triangle Meshes," *Computer Graphics (SIGGRAPH 92 Proceedings)*, vol. 26, no. 2, pp. 65–70, July 1992.

[3] Makoto Maruya, "Transforming Object-Surface Texture Data into a Texture Map," *NEC Research and Development*, vol. 36, no. 2, pp. 335–341, Apr. 1995.

[4] Bernard Chazelle, David P. Dobkin, Nadia Shouraboura, and Ayellet Tal, "Strategies for Polyhedral Surface Decomposition: An Experimental Study," in *Proceedings of the 11th Annual ACM Symposium on Computational Geometry*, June 1995, pp. 297–305.

[5] William E. Lorensen and Harvey E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics (SIGGRAPH 87 Proceedings)*, vol. 21, no. 4, pp. 163–169, July 1987.

[6] Ross T. Whitaker, "A Level-Set Approach to 3D Reconstruction from Range Data," *To Appear: International Journal of Computer Vision*, 1998.

[7] James Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Science*, Cambridge University Press, New York, 1996.