Isosurface Topology Simplification

Zoë Wood (Caltech)
Hugues Hoppe (Microsoft Research)
Mathieu Desbrun (U. of So. Cal.)
Peter Schröder (Caltech)

January 2002

Technical Report
MSR-TR-2002-28

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

This page intentionally left blank.

Isosurface Topology Simplification

Zoë Wood
Caltech

Hugues Hoppe
Microsoft Research

Mathieu Desbrun
U. of So. Cal.

Peter Schröder
Caltech

Abstract

Many high-resolution surfaces are created through isosurface extraction from volumetric representations, obtained by 3D photography, CT, or MRI. Noise inherent in the acquisition process can lead to geometrical *and* topological errors. Reducing geometrical errors during reconstruction is well studied. However, isosurfaces often contain many topological errors, in the form of tiny topological handles. These nearly invisible artifacts hinder subsequent operations like mesh simplification, compression, and parameterization. In this paper we present an efficient scheme for removing topological handles in an isosurface. Our scheme makes an axis-aligned sweep through the volume to locate handles, compute their sizes, and selectively remove them. Additionally, the algorithm is designed for out-of-core execution. It finds the handles by incrementally constructing and analyzing a surface Reeb graph. The size of a handle is measured as the shortest surface loop that breaks it. Handles are removed robustly by modifying the volume rather than attempting "mesh surgery." Finally, the volumetric modifications are spatially localized to preserve geometrical detail. We demonstrate topology simplification on several complex models, and show its benefit for subsequent surface processing.

Additional Keywords: topological artifacts, genus reduction, surface reconstruction, marching cubes.

1 Introduction

Highly accurate geometric models of physical objects are often acquired through discrete scanning techniques. For example, models are regularly obtained using laser range scanners, computed tomography (CT) or magnetic resonance imaging (MRI). Laser range scanners achieve full coverage of complex objects by acquiring and merging multiple scans. Many surface reconstruction algorithms perform the merging of scanned data using a volumetric grid representation, in which the model is represented as the zero-contour of its sampled distance function, *i.e.*, as an *isosurface* [4, 12, 14, 19]. Similarly, CT or MRI produce data volumes from which isosurfaces are extracted [20].

For surface reconstruction, one key advantage of an isosurface representation is that it naturally supports models of arbitrary genus, *i.e.*, with any number of "handles". For instance, the Buddha object used in Figure 1 has genus 6. Unfortunately, reconstructed isosurfaces may have higher genus than expected, due to the presence of extraneous topological handles. In fact, the scanned Buddha surface has genus 104 because of nearly invisible artifacts like the one revealed in Figure 2. Similar artifacts also arise in models acquired from CT and MRI scans, and can result in incorrect connectivity of biological structures, such as a brain surface with non-zero genus. In general, topological defects are caused by a number of factors, including sampling density, sampling noise, misalignment of scans, and grid discretization.

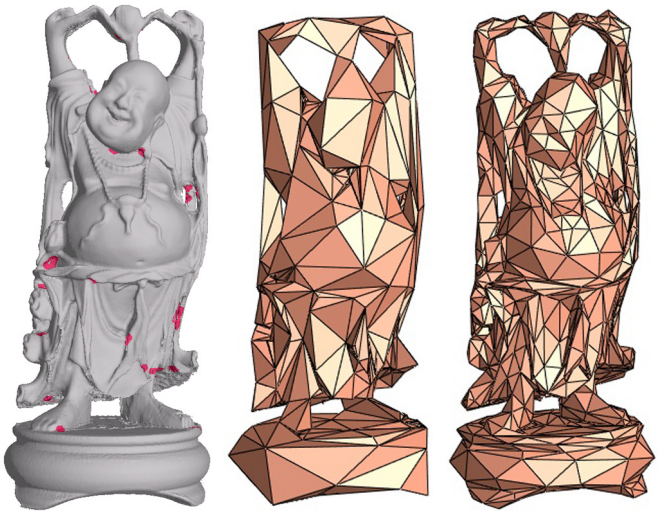


Figure 1: *This scanned Buddha mesh has genus 104 instead of the expected 6. Regions with extraneous handles are highlighted in red. The two images on the right compare mesh simplification results before and after topology simplification. The high-genus mesh requires many triangles to needlessly represent topological artifacts, resulting in loss of overall geometric quality.*

While often invisible, extraneous topological handles create significant problems for subsequent geometry processing like model simplification, smoothing, compression, and parameterization. As seen in Figure 1, traditional mesh simplification preserves all handles, resulting in inferior overall quality at coarse resolutions. Also, topological artifacts hinder any processing that must parameterize the surface, such as texture mapping and remeshing (see Section 3). Finally, correct topology can be essential for applications such as the fitting of organ templates to medical MRI data [26].

We present a method for removing topological defects in an isosurface. Rather than attempting to repair the defects on a mesh already extracted from the volume [9], our approach operates on the volume representation directly, as this offers advantages of efficiency and robustness. Our method performs a single sweep through the volume grid to locate topological handles, compute their sizes, and selectively remove them. The method offers the following contributions:

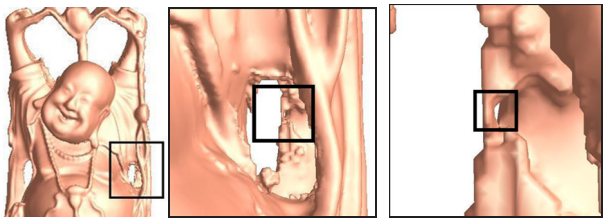
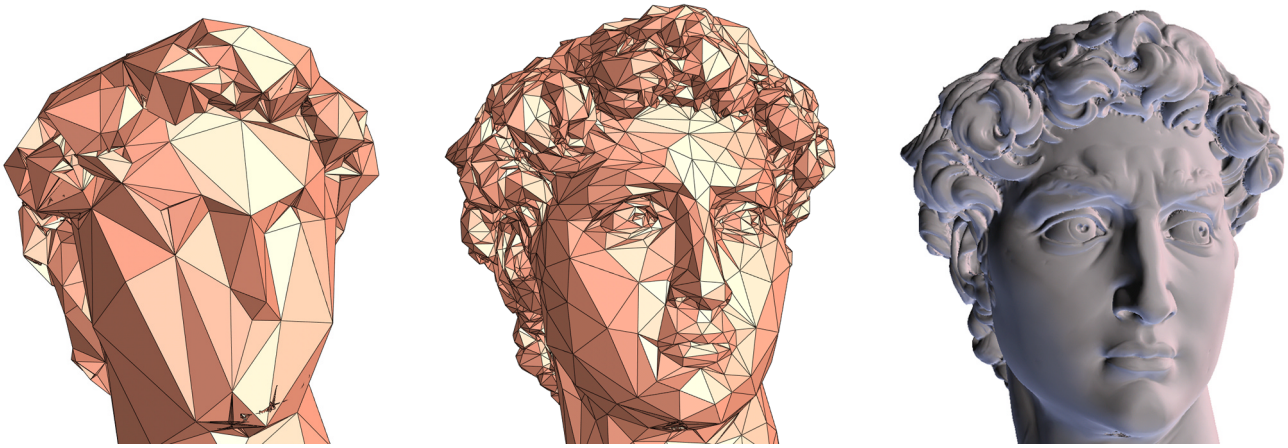


Figure 2: *Sequence of progressively closer views revealing an extraneous topological handle in the Buddha mesh.*



15,000 triangles
(a) Original model (genus 957)

15,000 triangles

1 million triangles
(b) Topologically simplified model (genus 0)

Figure 3: *Comparison of progressive meshes of the David model before and after topology simplification. On the far left, many triangles are wasted representing invisible topological artifacts. The right image demonstrates that topology simplification only requires minute changes that do not alter the visible appearance of the model.*

Out-of-core execution Complex 3D models are represented by large volumes that may not fit entirely in memory. The model in Figure 3 is from a $885\times709\times736$ grid, and much larger models now exist [19]. Our sweep method reads the volume in planar slices, so its data access pattern is highly regular. Moreover, we encode surface topology as the sweep progresses, using a Reeb graph so that few slices need be in memory at any time.

Fast identification of topological handles Handles are efficiently identified during the sweep, either as cycles in the Reeb graph as it is incrementally constructed, or as handles contained within a slice itself. We are guaranteed to detect all topological handles during the sweep.

Handle size estimation Some models have genus that should be preserved, such as the handles formed by the Buddha's arms. We introduce a novel measure of handle size as the length of its minimal surface loop, and remove all topological handles with a size smaller than a threshold.

Volumetric modification To remove a topological handle, we alter the scalar values of the volume, thus indirectly modifying the isosurface. Since isosurfaces are always manifold, operating on the volume is robust. In contrast, traditional "mesh surgery" must deal with issues of surface self-intersection and non-manifoldness.

Local repair To retain as much as possible the fine geometric detail of the model, our handle removal scheme aims to minimally perturb the original volume data. This is achieved by removing the shortest surface loop that simplifies the topology.

1.1 Related Work

Reeb graphs Given a scalar function defined on the surface, a *Reeb graph* tracks the connected components of the pre-image of the function. For instance, if the scalar function returns the z coordinate of the volume, its pre-image is the intersection of the surface with z planes, and the connected components consist of closed planar contours. The Reeb graph tracks how these contours split and merge as z varies. It is often used to analyze surface topology, since cycles in the graph correspond to topological handles. Shinagawa *et al.* [27] use this framework for the reconstruction of surfaces from contours. Axen and Edelsbrunner [2], Hilaga *et al.* [11], and Wood *et al.* [28] analyze Reeb graphs induced by a geodesic distance function with respect to a seed point. Because these geodesic-based schemes require a breadth-first traversal of the surface, the irregular accesses to the volume make out-of-core processing difficult. Like Shinagawa *et al.*, we construct a Reeb graph based on a scalar height function, and thus only require an axis-aligned sweep. We consider a discrete set of z grid intervals, rather than the continuous z function, and modify the graph construction accordingly.

Mesh-based topology simplification Guskov and Wood [9] remove topological noise from already extracted meshes. They repeatedly grow ϵ-balls over the surface, and remove any topological handle enclosed within such a ball using mesh surgery. Their approach has several drawbacks. For large ϵ, locating the topological handles is slow. Additionally, their definition of topological feature size fails to detect long thin handles, since they do not fit in a small ball. Finally, topological repair using mesh surgery can give rise to surface self-intersections.

Using the concept of alpha hulls, El-Sana and Varshney [6] reduce surface genus by re-tessellating small handles in a model. Their algorithm creates candidate tessellation regions by heuristically detecting crease edges in mechanical CAD models. The approach has not yet been generalized to work on more general surfaces. Edelsbrunner *et al.* [5] also use alpha hulls to characterize the sizes of topological features, by tracking the evolution of complexes. This allows for a combinatorial definition of topological feature size. While theoretically useful, the resulting structure is too heavy and rich for our purposes.

Volume-based topology simplification Nooruddin and Turk [22] convert a polygonal model into a volumetric representation in order to repair its topology. They apply morphological operations (dilation and erosion) to the volume data, causing topological handles to close. However, the operators affect the entire volume, resulting in the smoothing of geometry and thus loss of fine detail. We prefer a more targeted approach that exactly preserves geometric detail in regions away from topological artifacts.

Shattuck and Leahy [26] address the specific problem of constructing a genus-zero model of the human cortex from MRI scans, for use in cortical flattening and mapping. In contrast to the previous methods, they build Reeb graphs over the volume rather than the mesh. Specifically, they construct two Reeb graphs, encoding the connectivity of foreground and background voxels respectively. Their scheme removes all handles without regard to size, and always breaks handles along axis-aligned planes (Figure 9 shows an example where their strategy fails). In contrast, our approach performs one main sweep, constructs a single graph, has a more accurate measure of handle sizes, and repairs the volume with a more general and minimal operation.

Model simplification Several schemes simplify topology as a byproduct of model simplification [7, 10, 23]. Since these schemes simultaneously simplify geometry and topology, removing topological artifacts invariably involves loss of geometrical detail. In contrast, our focus is on simplifying topology while preserving geometrical detail.

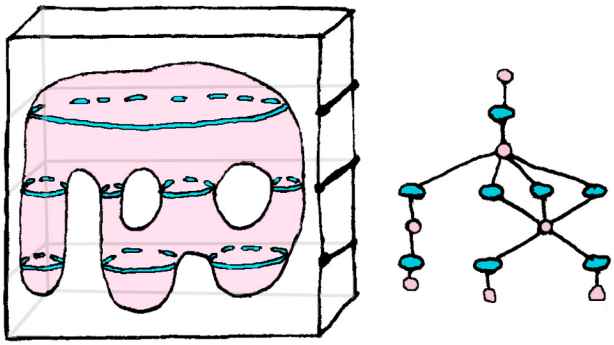


Figure 4: *An isosurface and its corresponding Reeb graph. In the graph, contour nodes are shown in blue, and ribbon nodes in pink.*

2 Our Approach

For clarity, we first introduce some definitions and terminology. Our input consists of a regularly sampled 3D grid of scalar values. A grid *cube* is bounded by 8 grid data points. Within each cube, an isosurface generation algorithm (such as [18] or [20]) defines a set of *surfels* (for surface elements) [28]. Each cube may have up to 4 surfels. The surfels from all cubes together form a polygonal mesh, which is a discrete representation of the isosurface. For our algorithm, the important element is connectivity of the surfels, as this connectivity defines the topology of the surface.

An axis-aligned *sweep* through the volume visits the grid data along parallel *planes*. The isosurface intersects each such plane along a set of *contours* (oriented closed polylines) as depicted on Figure 4 and 5. A *slice* of the volume is the set of grid cubes between two adjacent data planes. Within each slice, the surface may have several connected components; each such component is called a *ribbon*. The boundaries of a ribbon consist of one or more contours in the two adjacent planes.

A *topological handle* corresponds to a surface region with genus 1. The genus of a region with boundaries is computed by closing each boundary component with an end-cap. Note that a topological handle is unchanged if all data values in the volume are negated, *i.e.*, the model is turned inside out. Thus, we avoid the terms "tunnel" and "hole," as these have connotations of orientation. We define a *surface loop* as a closed path on the surface that *spans* a handle, *i.e.*, the surface remains connected when cut along the loop.

Problem statement The topology of a surface is characterized by its genus, its orientability, the number of its connected components, and the number of its boundary components [21]. Isosurfaces have the property that they are always orientable, and never have boundaries (if one pads all sides of the volume with "outside" scalar values). Thus, our problem of topology simplification corresponds to reducing surface genus, *i.e.*, removing topological handles.

Our algorithm deals with multiple disconnected components by concurrently simplifying them independently. Typically, for the final output, one discards all but the largest component, since the others are usually spurious artifacts, particularly for range data. However, for completeness we simplify the topology of all the components in the volume.

Our goal is to locate topological handles in the isosurface and selectively remove them. Removing a handle involves modifying the data values of nodes in the grid, from positive to negative or vice-versa. The ideal choice of which handles to remove is likely

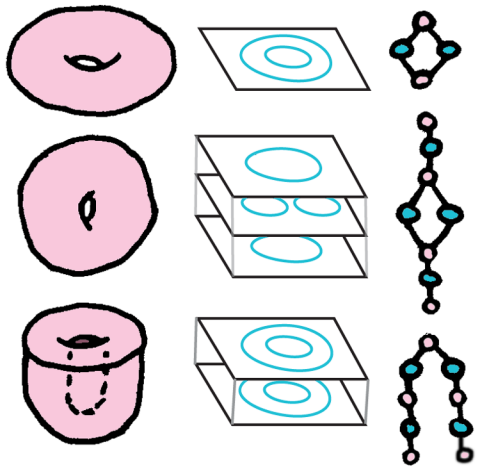


Figure 5: *Example surfaces and their associated contours and Reeb graphs. The examples are: a torus on its side, an upright torus, and a bowl-like surface.*

subjective, since some topology may be "inherent" to the model. While our system could be designed to locate handles and repeatedly ask the user for guidance, we sought an automatic solution. To make this problem computationally tractable, we introduce a definition for handle size, and let our scheme remove all handles whose measured sizes are smaller than a user-provided threshold ℓ. Specifically, we define the size of a handle as the length of the minimal loop spanning the handle. See Figure 6 and 9 for an illustration of such loops. The issue of setting the handle size threshold ℓ is discussed in Section 3.2.

Approach overview Our approach can be summarized as:

- Sweep through the volume to *locate* all topological handles.
- For each handle found, *measure* its size.
- If the size is sufficiently small, *remove* the handle.

We now present each of these steps in more detail.

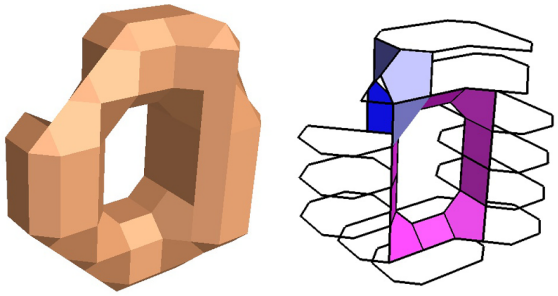


Figure 6: *On this irregularly shaped torus, the Reeb loop is shown in magenta, and the cross loop is shown in blue. Note that the cross loop, which corresponds to the shortest loop around the handle, is not limited to a single slice of the volume in this example.*

2.1 Locating Topological Handles

Determining the genus of an isosurface is a relatively simple task. One can sweep through the volume and count the number of vertices, edges, and faces which would be generated during isosurface mesh extraction [18]. The *Euler characteristic* is then $\chi = |V| - |E| + |F|$, and the surface *genus* is $g = (2 - \chi)/2$. However, this genus analysis fails to provide any information as to the location or size of topological handles.

To locate handles, we perform a sweep through the volume along the z axis, and construct a Reeb graph to track the connected components of the surface as the sweep advances. More precisely, we analyze the isosurface one slice at a time, which corresponds to an

interval in z. Within a slice, the surface is made up of ribbons, whose boundaries are contours in the two adjacent z planes. Both the ribbons and contours are identified using breadth-first search to find connected sets of surfels (in the slice) and edges (in the planes) respectively. We create nodes in the Reeb graph corresponding to *both* ribbons and contours, and record their adjacency as graph edges, as illustrated in Figures 4 and 5. Cycles in the Reeb graph correspond to topological handles on the surface.

Because our Reeb graph considers surface connectivity over discrete z intervals, rather than continuously, some topological handles may not be detected as cycles in the graph because they are entirely contained within a slice. We detect these by computing the Euler characteristic of each surface ribbon. If a ribbon has non-zero genus, it obviously contains handles. One such *intra-ribbon handle* is shown in Figure 7. In practice, these intra-ribbon handles compose 10-20% of the total topology, depending on the input and the sweep direction (see discussion in Section 3). Note that an isosurface cannot have a handle within a single cube, nor in a single row of cubes. Within a slice, the smallest configuration appears to be a 6×6 cube configuration which brings about the surface in Figure 7.

To summarize, the genus of the isosurface S is partitioned as:

$$g(S) \;=\; \#\text{cycles(Reeb Graph)} \;+\; \sum_{r \in \text{ribbons}} g(r)\;.$$

We next discuss how to locate the handles in both cases.

Finding cycles in the Reeb graph Cycles in the Reeb graph are detected incrementally as the sweep advances through the volume. This progressive detection allows for handle removal to occur concurrently during the sweep. Our approach is as follows.

For each Reeb graph node, *i.e.*, both ribbons and contours, we associate a label that identifies the connected component to which it belongs. The only way that a cycle can form is when an edge is added in the Reeb graph from a ribbon node to a contour node in the previously visited plane. When adding such an edge, we test whether the two nodes have the same label. If so, they belong to the same connected component and a cycle is formed. In any case, after the edge is added, we relabel the graph nodes to reflect the merging of connected components. This process is implemented efficiently using a Union-Find algorithm on a disjoint-set data structure [3], taking negligible time.

When a cycle is detected, we perform a breadth-first search through the graph to find the shortest cycle. Note that there may be multiple paths, due to nested cycles associated with handles we chose to preserve earlier. The cycle path consists of alternating ribbon and contour nodes and defines a *topological handle*.

The following pseudocode summarizes the key parts of the detection algorithm:

function Add_ribbon_to_Reeb_graph(ribbon r, ReebGraph G)
  Add ribbon r as node in Reeb graph G.
  label(r) := unique_label().
  Identify previous contours C adjacent to r on surface.
  Foreach (pair contours $c1, c2 \in C$)
    if label($c1$) = label($c2$) **then**
      path P := shortest path from $c1$ to $c2$ in G.
      Report cycle as $(c2, r) + (r, c1) + P$.
  Foreach (contour $c \in C$)
    Add edge (c,r) to G.
    Unify labels of contour c and ribbon r.

Finding intra-ribbon handles Recall that we must also consider the case of a topological handle contained entirely within a ribbon. When a ribbon is identified as having non-zero genus, our task is to locate the surface loop(s) within it. To avoid writing a special-purpose algorithm for this 2D case, we create a temporary mini-volume composed of just the relevant slice, padded on both sides

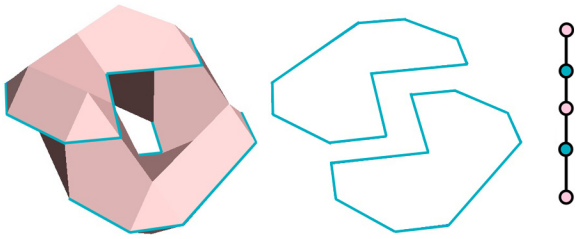
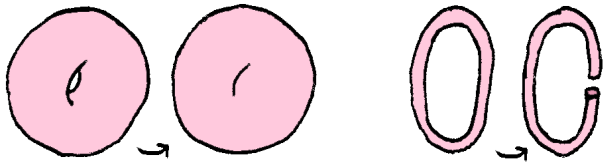


Figure 7: *Example of intra-ribbon handle. This torus tilted at an angle is formed by two "C" shaped contours. As shown on the right, the Reeb graph does not contain any cycle.*

by exterior values such that the contours are closed by end-caps. In the rare case that these contours are nested, this padding must be appropriately widened to always close contours with end-caps. Within this mini-volume, the isosurface matches the original isosurface *only* within the slice. However, it has the same genus as the original ribbon, and notably, it has no intra-ribbon handles. We then apply our regular sweep algorithm to this mini-volume along an orthogonal direction. For each cycle detected in the resulting Reeb graph, we find the restriction of the ribbon cycle to the original slice of the volume, since the geometry of the isosurface only matches there.

As a sidenote, an alternative scheme we explored to deal with intra-ribbon handles is to sweep across the volume along all 3 axis directions, hoping to detect the handle as an ordinary Reeb cycle. To our regret, we found that there can exist handles that are intra-ribbon in all 3 directions simultaneously. Fortunately, our current approach of performing orthogonal sweeps only locally (on mini-volumes) guarantees finding all handles, and is more efficient.



handle collapse handle pinching

Figure 8: *Two ways of removing a topological handle, illustrated on two tori. The "fat" torus is best repaired by collapsing the handle, and the "skinny" torus is best repaired by pinching the handle*

2.2 Measuring Topological Handle Size

Recall that a cycle in the Reeb graph identifies a cycle of ribbons forming a topological handle. There are two natural ways to remove a handle (Figure 8):

- *handle collapse* by filling the interior of the cycle, and
- *handle pinching* by breaking the ribbon cycle.

Local surface geometry determines whether the collapse or pinching operation is more appropriate, as illustrated in Figure 8. Both handle collapse and handle pinching are in fact the same operation applied to two different surface loops. (Recall that a loop is a closed curve that spans the handle.) We call this operation *loop closure*. Intuitively, loop closure removes the handle by removing a thin strip of surface about the loop, and closing the resulting two boundaries using two parallel "membranes" spanning the loop. The actual implementation of this operation on our discrete grid volume is discussed in the next section.

Two characteristic loops To find the best loop closure operation, we compute two surface loops:

- the *Reeb loop* which is the smallest loop around the ribbon cycle, and
- the *cross loop* which is the smallest loop "transversal" to the Reeb loop, *i.e.*, pinching the handle.

Thus, for a vertical torus, the Reeb loop length measures its inner circumference, and the cross loop length measures its girth. See Figure 6 and 9 for an example of both loops. Since we perform only a single sweep, it is important that we consider both types of handle removal operations. We therefore define handle size to be the smaller of the Reeb loop length and the cross loop length.

We find the **Reeb loop** using two successive searches as follows. The Reeb cycle contains at least one pair of contours in the same plane. We compute the all-points shortest paths from points on the first contour to points on the second contour, constrained to go through the lower portion of the ribbon cycle. Then using the endpoints of this first search, we continue the all-points shortest paths back to the first contour, constrained to the upper portion of the ribbon cycle. Among all shortest paths ending at their starting point, the shortest is the Reeb loop. As an implementation detail, we currently represent paths over surfels instead of the mesh vertices and edges that a MC extraction would produce, as the difference is not significant due to the regular sampling of our volume data.

We construct the **cross loop** in a similar manner. We now pretend that the surface has been cut along the Reeb loop. Starting from one side of the Reeb loop, we compute the all-points shortest paths to the points on the other side of the Reeb loop. Among all shortest paths forming cycles, the shortest is the cross loop. Note that this cross loop is not required to lie along a contour. It can cut diagonally through the volume, as shown in Figure 6 and 9.

Measure of topological handle sizes From these two characteristic loops, we can now derive a measure of the topological handle. Generally, we use the smaller of the two loops as the measure of handle size. If desired, we can provide additional user-control. For example, if the user wants to avoid removing long skinny handles, we can preserve handles that have a large ratio between the two loop sizes. Also, the user can specify that material is to be only added or only subtracted from the volume. From the orientation of any contour in the Reeb graph cycle, one can determine whether the ribbon cycle encloses a void or encloses material. It is always the case that exactly one of the two loops (Reeb loop and cross loop) encloses empty space while the other encloses the model interior. We can therefore exclude the appropriate loop if desired.

As a measure of loop size, we chose the perimeter length of the loop. This length corresponds to the extent of the cut along the surface necessary for loop closure. An alternative would be to measure the area of the loop, *e.g.*, the area of the spanning minimal surface. This area would correspond to the extent of the new surface necessary for loop closure. We have chosen loop length because it seems a tighter measure than area. Consider a handle in the shape of a wide, thin-walled vertical tube. The cross loop is then a tall, thin rectangle. Even though the loop area may be quite small, its perimeter is quite long, and will therefore be preferable to identify this handle as a large feature. Notice finally that this measure is independent of sweep direction.

Considerations for intra-ribbon handles The intra-ribbon handles require special treatment. As noted in section 2.1 we find intra-ribbon handles by performing an orthogonal sweep on a mini-volume containing the intra-ribbon handle. This approach is *guaranteed* to discover the intra-ribbon handle. However, it may not find the minimal loop to simplify the handle since we restrict our simplification to collapsing the Reeb loop. Recall that we ignore cross loops in this mini-volume because their geometries would not correspond to the original volume. Therefore, we use the following modification in practice:

- We first find just the Reeb loop within the mini-volume using an orthogonal sweep as previously described. Most intra-ribbon handles are small and have Reeb loops of size $< \ell$. For example, 22 of the 26 intra-ribbon handles in the Buddha model have Reeb loops of size 4.
- For the few intra-ribbon handles that do not have small Reeb loops, we expand our orthogonal sweep to a larger mini-volume of size 2ℓ.

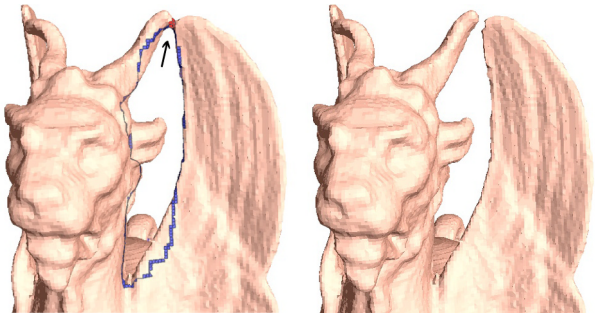


Figure 9: *Close-up of the feline mesh with the Reeb loop shown in blue, and cross loop shown in red. The right image shows the result of pinching the handle at the cross loop as done by our algorithm.*

In this expanded mini-volume, we are no longer restricted to only collapsing Reeb loops. If the intra-ribbon handle has a cross loop of size $< \ell$, the handle is simplified by closing this loop. These second passes seldom occur. This slight modification both guarantees a correct handling of intra-ribbon handles *and* finds the minimal loops.

2.3 Removing Topological Handles

The same minimal loop used to define handle size is also used to remove the handle through loop closure. We perform loop closure on the isosurface by scan-converting a surface spanning the loop into the volume grid data [15]. Since the loop is generally non-planar, one could construct some approximation to the minimal spanning surface. For efficiency, we simply use a triangle fan about the centroid of the loop. The scan-conversion writes either positive or negative scalar values in the grid, depending on the orientation of the loop (discussed in Section 2.2). This rasterization technique both collapses and pinches off handles through insertion of a thin wall. The modified isosurface is guaranteed to remain a manifold and to have no self-intersections. See Figure 10 for an example of topological simplification.

There are a few potential problems to consider. The fan of triangles closing the non-planar loop could be self-intersecting, or could intersect other regions of the surface, for instance, if the handle were to contain another, nested handle. In practice, this is unlikely since we find a minimal loop and only close it if it is small. At worst, the loop closure could introduce additional topological handles. But since we locally re-build the Reeb graph after a loop closure operation, these new handles would be processed subsequently.

3 Results and Discussion

We have run our topology simplification scheme on a number of volumes, as shown in Table 1. The Buddha, dragon, feline and David models are from laser range scans at Stanford University. The brain model is from an MRI scan from the Harvard Medical School [17].

The number of intra-ribbon handles varies strongly depending upon the data and the sweep direction. For example, the brain MRI has 120 intra-ribbon handles in the original scan direction. This high number is due to the nature of the data. MRI is typically segmented by hand, and small misalignments between these segmented contours commonly give rise to intra-ribbon handles. Sweeping the brain MRI data along an orthogonal direction produces only 6 intra-ribbon handles. Given this observation, we choose to sweep all MRI volumes in a direction orthogonal to the original data orientation. This reduces execution time since fewer mini-volumes are created. For range scans, intra-ribbon handles are less frequent, and seem to be independent of sweep direction.

Model	Grid size	#Faces	Thresh. size ℓ	Genus		#Intra-ribbon	Handles removed		Timing (minutes)
				original	simplified		#collapse	#pinching	
Buddha	$400\times400\times950$	4,736,292	9.5	106	6	26	42	58	6.5
Dragon	$500\times714\times324$	3,222,612	46.5	60	1	18	31	28	3.8
David	$885\times736\times709$	15,244,302	166.5	1063	0	76	332	731	87.5
Brain	$125\times255\times255$	688,248	32.5	366	0	6	320	46	2.8
Feline	$332\times148\times316$	653,922	4.5	6	2	1	2	2	0.2

Table 1: *Quantitative results: The handle threshold size ℓ is expressed in units of cube edge size. The number of removed handles (original genus minus simplified genus) is broken down into handle collapse and handle pinch operations. Times are shown in CPU minutes. All values listed are for the entire volume,* i.e., *for the surface and any spurious disconnected components in the volume data.*

During topology simplification, collapse and pinch operations appear with approximately equal frequency. Excess topology is generally small, in terms of both Reeb and loop sizes, and is oriented randomly throughout the volume, leading to equal likelihood of either the Reeb or cross loop having size $< \ell$.

The scatterplot in Figure 11 shows a typical distribution of handle sizes for an object with large-scale topology. Typically, extraneous handles in the isosurface are small with 90% having loop lengths of 4–8 (see Figure 10). However, there are some volumes containing handles with larger Reeb and cross loops. For laser range data, these larger loops are typically associated with spurious data, external to the intended surface. For example, whereas the surface of the dragon has predominantly small handles, one of its spurious external surface component has a handle of size 46.

The timing for our algorithm depends on the size of the volume and on the number of topological handles. It depends particularly on the number of handles that need to be simplified, since the Reeb graph must be locally rebuilt each time a handle is simplified. In general, our processing takes on the order of minutes.

We have verified the robustness of our algorithm using convoluted geometry (Figure 14) and large volumes (Table 1). Our method is also able to robustly simplify topology for large handle sizes. For example, setting ℓ to infinity produces a genus-zero Buddha, where even the large handles (with lengths up to 246) are removed (Figure 10).

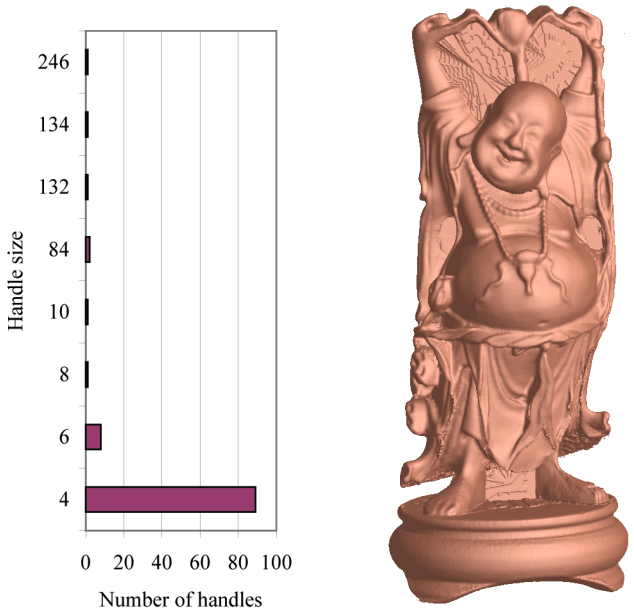


Figure 10: *Histogram of handle sizes for the original scanned Buddha model. Recall that handle size is the smaller of the Reeb and cross loop lengths. Setting the loop size threshold ℓ to infinity for topology simplification results in a genus-zero Buddha.*

3.1 Applications

Topology simplification facilitates many surface operations:

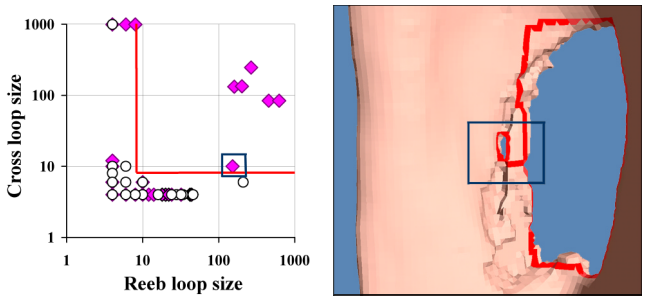
- Fewer triangles are wasted to encode topological defects during *mesh simplification*, as shown in Figures 1, 3, 14 and 13 using the progressive mesh representation of Hoppe [13]. Consequently, coarser meshes can be created, and geometric quality is improved at all levels of detail.
- Better surface parameterization improves *texture mapping*, as shown in Figure 15 using the scheme of Sander *et al.* [24]. Fewer charts are necessary to partition the surface, which results in a nicer parametric domain.
- Removal of topological defects permits *remeshing*, as shown in Figure 12 using the method of Guskov *et al.* [8]. The remesh has nice regular face sizes and allows for efficient progressive geometry compression [16] as well as many other semi-regular geometry processing algorithms [25]. The topologically clean volumes can also be more readily used for semi-regular mesh extraction [28].
- Greater *mesh compression* is achievable. With the scheme of Alliez and Desbrun [1], the compressed size of the Buddha is reduced from 838,446 bytes to 796,066 bytes.

3.2 Discussion

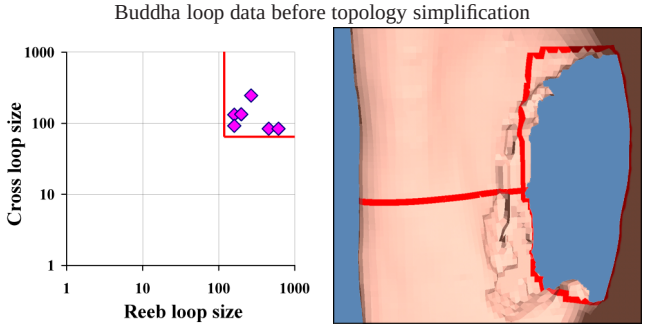
Setting the handle size threshold For our examples, we first make an initial pass over the volume to gather statistics on handle sizes, and examine these using a histogram or scatterplot (Figures 11 and 10). By looking at the relative sizes of topological handles, we select an appropriate ℓ. For most of the models, the excess topology has loop lengths in the range of 4–8. Thus, our setting of ℓ typically ranges from 10–20.

We observed that the initial statistics can change significantly as topological handles are filtered. Figure 11 shows a large handle with a small nested handle. For this configuration, the large handle has a large Reeb loop and small cross loop, and the small handle has an even smaller Reeb loop and shares the same cross loop. During topology simplification, the small handle is removed first leaving only the large handle which now has both large Reeb *and* cross loop. This phenomenon is also reflected in the scatterplots before and after topology simplification (Figure 11), where a data point near the ℓ line moves to the top right once the small handle is removed.

Handle size approximation Our method for computing the minimal loop size makes several approximations. First, the shortest loop is not computed as the geodesic over the continuous surface, but as the shortest path over the discrete surfel connectivity graph. Second, our current implementation assigns all edges in this graph a constant cost, motivated by the fact that all cubes have uniform size. Euclidean distance costs could be computed, but the resulting effect is too small in our examples to matter as the minimal loops are very small to begin with. Even for large loops, our approximation remains reasonable as shown in Figure 9.



Buddha loop data before topology simplification



Buddha loop data after topology simplification

Figure 11: *Scatterplot of the Reeb loop and cross loop lengths of the handles of the Buddha, before and after topology simplification. Hollow circles identify handles whose minimal loop encloses a void. The red lines mark the range of ℓ that keeps exactly these 6 handles. On the right we see corresponding close up view of two adjacent handles on the Buddha model with a shared small cross loop. After topology simplification (bottom), the small handle is collapsed and the larger handle now has a larger cross loop.*

Algorithm time complexity With respect to time complexity, in practice the overriding term is the traversal of the volume, which requires accessing $O(n^3)$ grid values, where n is the extent of the grid in each dimension. Typically the surface has only $O(n^2)$ surfels, and the Reeb graph only $O(n)$ nodes and edges, so the processing steps related to the surface and Reeb graph do not require significant time. However, there is processing time associated with each topological handle discovered and its subsequent measurement and possible removal. We can restrict this processing time based on the size of loop we are simplifying, *i.e.*, we can short-circuit any breadth-first search that is already less than ℓ. However, for every handle that is simplified, we must reconstruct the Reeb graph locally to account for the resulting changes. For large volumes with many changes, *e.g.*, the David volume, this reconstruction cost can be significant, but is required in order to accurately code the topology of the isosurface.

Algorithm space complexity Probably more important are the space requirements. Computing the Reeb loop for a handle requires access to the surfels in all ribbons referred to in the Reeb cycle. Accurate computation of the cross loop requires additional slices above and below the cycle. The number of additional slices is determined according to ℓ, such that we are guaranteed to find a cross loop of length less than ℓ if one exists. The worst case situation is that of a very long handle with a thin cross-section somewhere along its length. Given a strict memory budget, Reeb and cross loop computation may require reloading previous slices of the volume that have already been flushed from memory. In practice, we only keep 50 slices of volume in memory at any time, making the algorithm viable even for low-end computers.

4 Summary and Future Work

We have introduced a scheme for automatically removing topological handles from isosurfaces through direct processing of the original volume data, and demonstrated its effectiveness on several complex models. We have also demonstrated that removing topological defects is important for many subsequent modeling operations.

One area of future work is to improve the local surface geometry after handle removal. The handles removed in our test examples were so small as to be nearly invisible, so we did not consider smoothing to be important. However, it is conceivable that topological defects could be of more substantial size. Since we have information about the local region affected by the loop closure, we could smooth the newly inserted surface. For example, this smoothing would improve the visual appearance of the regions bounded by the arms in the genus-zero Buddha (Figure 10). In range data reconstruction algorithms, it is already common to smooth the unscanned, filled-in regions of the surface [4].

For larger handles, it may be desirable to use more accurate ap-

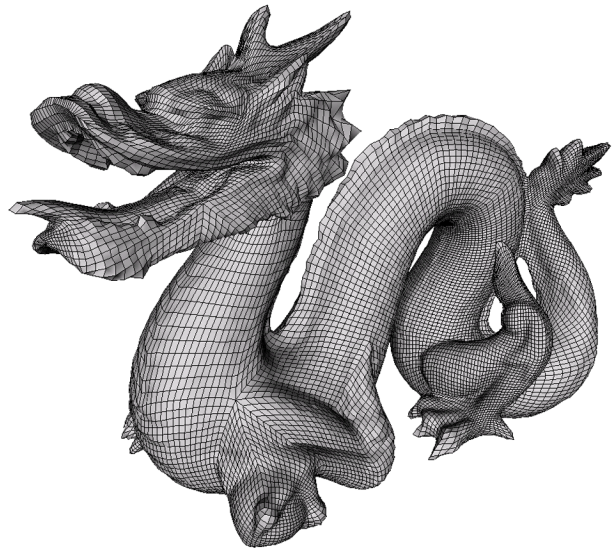
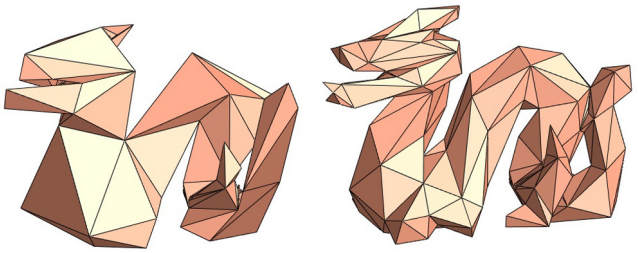
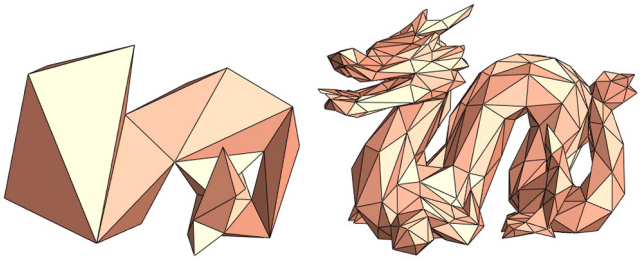


Figure 12: *A remesh of the genus 1 dragon. Remeshing the original scanned dragon with genus 46 would be nearly impossible, given the difficulty of achieving a high-quality parameterization for high-genus models.*



base mesh (752 triangles) simplified mesh (1,000 triangles)

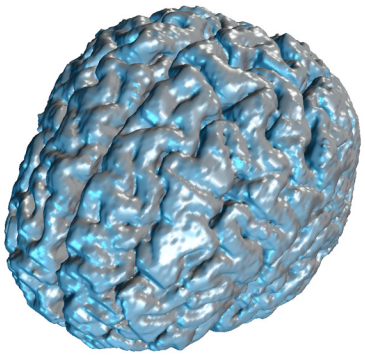
(a) Original model (genus 43)



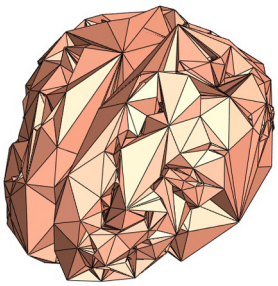
base mesh (52 triangles) simplified mesh (1,000 triangles)

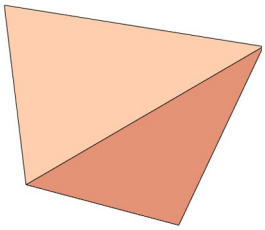
(b) Topologically simplified model (genus 1)

Figure 13: *Comparison of progressive meshes with a given triangle budget on the dragon before and after topology simplification.*



Topologically simplified (genus 0)





base mesh (5,464 triangles)
Original (genus 366)

base mesh (4 triangles)
Topologically simplified (genus 0)

Figure 14: *Comparison of the base meshes of progressive meshes on a brain model (MRI).*

proximations of true geodesic surface loops rather than the discrete graph approximation. More generally, we are interested in exploring alternative methods for measuring topological handle size.

To explore data such as MRI, some systems allow the isosurface value to be varied interactively. Efficiently removing handles in the changing isosurface is an interesting problem. Perhaps it is possible to pre-process the volume to remove topological artifacts for a range of isosurface values.

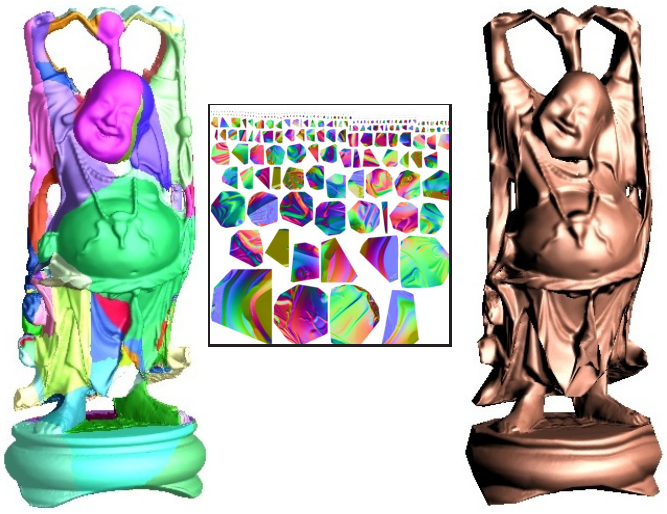
Acknowledgments This work was supported in part by the NSF (DMS-9874082, ACI-9721349, DMS-9872890, ACI-9982273), the DOE (W-7405-ENG-48/B341492), Intel, Alias|Wavefront, Pixar, Microsoft, and the Packard Foundation. We thank the computer graphics lab at Stanford University for the David model from the Digital Michelangelo library. We thank Drs. Kikinis, M. Shenton, R. McCarley, and F. Jolesz of the Harvard Medical School for the brain model. The authors thank Jacques-Olivier Lachaud for his isosurface generation code. The authors thank Nathan Litke and Andrei Khodakovsky for their assistance with remeshing, and Steven Gortler for conversations about topology.

References

[1] Alliez, P., and Desbrun, M. Progressive Compression for Lossless Transmission of Triangle Meshes. *Proceedings of SIGGRAPH* (2001), 195–202.

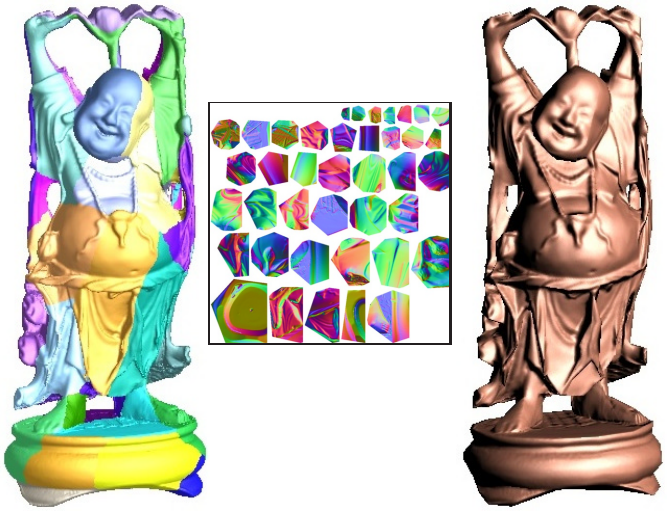
[2] Axen, U., and Edelsbrunner, H. Auditory Morse analysis of triangulated manifolds. In *Mathematical Visualization*, H.-C. Hege and K. Polthier, Eds. Springer-Verlag, Berlin, Germany, 1998, pp. 223–236.

[3] Cormen, T., Leiserson, C., and Rivest, R. *Introduction to algorithms*. MIT Press, 1990.



353 surface charts texture domain 3,700 triangles

(a) Original model (genus 101)



40 surface charts texture domain 2,000 triangles

(b) Topologically simplified model (genus 6)

Figure 15: *Comparison of normal-mapping progressive meshes before and after topology simplification. Both models refer to 512x512 texture images. The topological complexity of the original model requires many more parametric charts, shown in pseudocolor. The resulting fragmentation of the parametric domain restricts simplification.*

[4] Curless, B., and Levoy, M. A Volumetric Method for Building Complex Models from Range Images. *Proceedings of SIGGRAPH* (1996), 303–312.

[5] Edelsbrunner, H., Letscher, D., and Zomorodian, A. Topological Persistence and Simplification. *Proc. 41st IEEE Sympos. Found. Comput. Sci.* (2000), 454–463.

[6] El-Sana, J., and Varshney, A. Controlled Simplification of Genus for Polygonal Models. *Proceedings of IEEE Visualization* (1997), 403–412.

[7] Garland, M., and Heckbert, P. S. Surface Simplification Using Quadric Error Metrics. *Proceedings of SIGGRAPH* (1997), 209–216.

[8] Guskov, I., Khodakovsky, A., Schröder, P., and Sweldens, W. Hybrid Meshes. Available at http://multires.caltech.edu/pubs/hybrid.pdf, January 2001.

[9] Guskov, I., and Wood, Z. Topological Noise Removal. *Graphics Interface* (June 2001), 19–26.

[10] He, T., Hong, L., Varshney, A., and Wang, S. W. Controlled Topology Simplification. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (1996), 171–184.

[11] Hilaga, M., Shinagawa, Y., Kohmura, T., and Kunii, T. L. Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. *Proceedings of SIGGRAPH* (2001), 203–212.

[12] Hilton, A., Toddart, A. J., Illingworth, J., and Windeatt, T. Reliable surface reconstruction from multiple range images. *Fourth European Conference on Computer Vision I* (1996), 117–126.

[13] Hoppe, H. Progressive Meshes. *Proceedings of SIGGRAPH* (1996), 99–108.

[14] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. Surface reconstruction from unorganized points. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2 (1992), 71–78.

[15] Kaufman, A. Scan-conversion of Polygons. *Proceedings of Eurographics* (1987), 197–208.

[16] Khodakovsky, A., Schröder, P., and Sweldens, W. Progressive Geometry Compression. *Proceedings of SIGGRAPH* (2000), 271–278.

[17] Kikinis, R., and et al. A Digital Brain Atlas for Surgical Planning, Model Driven Segmentation and Teaching. *IEEE Transactions on Visualization and Computer Graphics* (1996).

[18] Lachaud, J.-O. Topologically Defined Iso-surfaces. In *Proc. 6th Discrete Geometry for Computer Imagery (DGCI)* (1996), vol. 1176 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 245–256.

[19] Levoy, M., and others. The Digital Michelangelo Project: 3D Scanning of Large Statues. *Proceedings of SIGGRAPH* (2000), 131–144.

[20] Lorensen, W. E., and Cline, H. E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Proceedings of SIGGRAPH 21*, 4 (1987), 163–169.

[21] Massey, W. *Algebraic Topology: An Introduction*. Harcourt, Brace & World, Inc., 1967.

[22] Nooruddin, F., and Turk, G. Simplification and Repair of Polygonal Models Using Volumetric Techniques. Research Report 99-37, Georgia Tech, 1999.

[23] Popovic, J., and Hoppe, H. Progressive Simplicial Complexes. *Proceedings of SIGGRAPH* (1997), 217–224.

[24] Sander, P., Snyder, J., Gortler, S., and Hoppe, H. Texture Mapping Progressive Meshes. *Proceedings of SIGGRAPH* (2001), 409–416.

[25] Schröder, P., and Sweldens, W., Eds. *Digital Geometry Processing*. Course Notes. ACM Siggraph, 2001.

[26] Shattuck, D. W., and Leahy, R. M. Automated Graph Based Analysis and Correction of Cortical Volume Topology. *IEEE Transaction on Medical Imaging* (2001).

[27] Shinagawa, Y., and Kunii, T. L. Constructing a Reeb Graph Automatically from Cross Sections. *IEEE Computer Graphics and Applications 11*, 6 (1991), 44–51.

[28] Wood, Z., Desbrun, M., Schröder, P., and Breen, D. Semi-Regular Mesh Extraction from Volumes. In *Proceedings of Visualization* (2000), pp. 275–282.