Removing Excess Topology From Isosurfaces

ZOË WOOD
Caltech
HUGUES HOPPE
Microsoft Research
MATHIEU DESBRUN
University of Southern California
and
PETER SCHRÖDER
Caltech

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 handles. These nearly invisible artifacts hinder subsequent operations like mesh simplification, remeshing, and parametrization. In this article we present a practical method for removing handles in an isosurface. Our algorithm makes an axis-aligned sweep through the volume to locate handles, compute their sizes, and selectively remove them. The algorithm is designed to facilitate out-of-core execution. It finds the handles by incrementally constructing and analyzing a Reeb graph. The size of a handle is measured by a short nonseparating cycle. 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 benefits for subsequent surface processing.

Categories and Subject Descriptors: I.3.0 [**Computer Graphics**]: General

General Terms: Algorithms, Performance

Additional Key Words and Phrases: 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 commonly obtained using laser range scanners, computed tomography (CT), or magnetic resonance imaging (MRI). These acquisition techniques frequently produce volume

This work was supported in part by the National Science Foundation (NSF) grants DMS-9874082, ACI-9721349, DMS-9872890, ACI-9982273, CCR-0133983, DMS-0221666, DMS-0221669, and EEC-9529152; the Department of Energy (DOE) (W-7405-ENG-48/B341492); Intel; Alias Wavefront; Pixar; Microsoft; and the Packard Foundation.
Authors' present addresses: Z. Wood, Computer Science Department, Cal Poly, San Luis Obispo, CA 93407, email: zwood@csc.calpoly.edu; H. Hoppe, Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399; http://research.microsoft.com/~hoppe; M. Desbrun, Computer Science Department, University of Southern California, 941 West 37th Place, Los Angeles, CA 90089-0781; email: desbrun@csc.edu; P. Schröder, Caltech, MS 256-80, 1200 East California Blvd., Pasadena, CA 91125; email: ps@cs. caltech.edu.

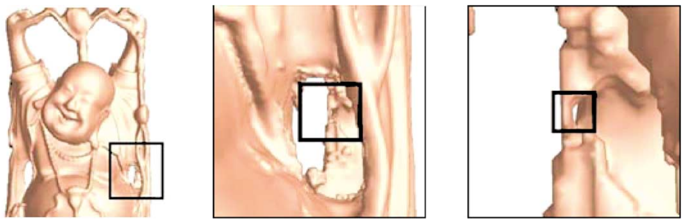

Fig. 1. Sequence of progressively closer views revealing an extraneous handle in the Buddha mesh.

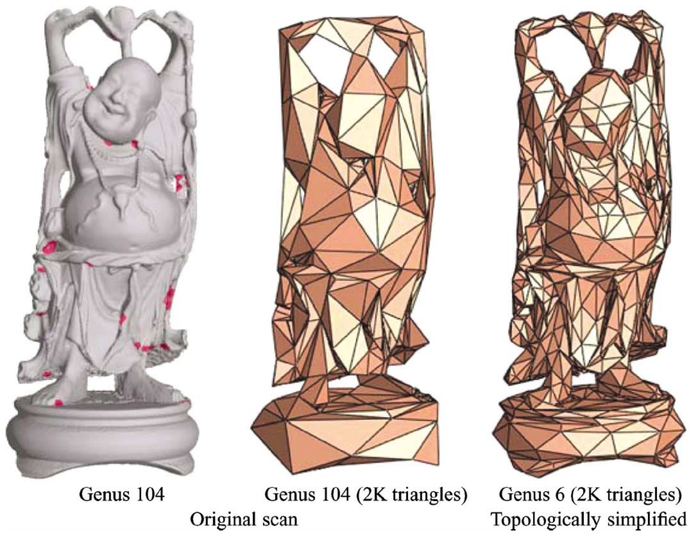

Fig. 2. This scanned Buddha has genus 104 instead of the expected 6. Regions with extraneous handles are highlighted in red. The two images on the right compare mesh simplification results before and after topology simplification for a given triangle budget.

data from which *isosurfaces* are extracted [Curless and Levoy 1996; Hilton et al. 1996; Hoppe et al. 1992; Levoy et al. 2000; Lorensen and Cline 1987; Wyvill et al. 1986].

For surface reconstruction, one advantage of an isosurface representation is that it naturally supports models of arbitrary genus, that is, with any number of handles. A *handle* is a region of the surface with genus $g = 1$ (various formal definitions exist, see, e.g., Francis and Weeks [1999]). An example of an isosurface with nontrivial genus is the Buddha statue used in Figure 2 that has genus 6. Unfortunately, reconstructed isosurfaces may have much 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 1. 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 nonzero genus. In general, these topological defects are caused by a number of factors, including sampling density, sampling noise, misalignment of scans, and grid discretization.

While often invisible, extraneous handles create significant problems for subsequent geometry processing like model simplification, smoothing, and parametrization. As seen in Figure 2, traditional mesh simplification preserves all handles, resulting in inferior overall quality at coarse resolutions. Also, topological artifacts hinder any processing that must parametrize 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 [Jaume et al. 2002; Shattuck and Leahy 2001].

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 [Massey 1967]. 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). Multiple disconnected components can be addressed independently. Our problem of topology simplification corresponds to reducing surface genus, that is, removing handles. The ideal choice of which handles to remove is subjective, since some may be inherent to the real world model. While topology simplification could proceed by locating handles and repeatedly asking the user for guidance, we sought an automatic solution.

Approach Overview. Our approach distinguishes handles based on a geometric measure of handle size. Using a greedy method to isolate geometrically succinct handles, our method targets removing handles with a small size. The isolated local handles can be more easily measured and removed one-by-one as we do not require global access and analysis of the entire surface topology. Our method sweeps through the volume grid and builds a graph (called an *augmented Reeb graph*) to represent the topology of the isosurface. This graph is used to isolate handles which are subsequently measured and selectively removed. Our algorithm is greedy and processes a single local handle at a time, before moving on to finding the next one.

The algorithm measures and removes a handle if its size is smaller than a user-provided threshold ℓ. For our application of topology simplification, we define the appropriate measure of the size of a handle to be the length of the shortest *nonseparating cut*. A nonseparating cut is a closed curve that leaves the surface connected [Aleksandrov 1956]. For example, a sphere has no nonseparating cuts. See Figure 12 for an illustration of such cycles. The genus of a surface is defined as the maximum number of simple nonseparating cuts that do not intersect. A simple curve is one that does not cross itself. To simplify the topology of a surface, we reduce the number of nonseparating cycles by cutting the surface along the cycle and filling in the cuts with a surface membrane. To support out-of-core processing of the data, our algorithm does not find globally minimal-length nonseparating cuts; instead, our method finds nearly shortest nonseparating cycles for each handle.

The contributions of our method are as follows:

Handle Size Estimation and Local Repair. We introduce a simple measure of handle size to be the length of a nonseparating cut, and remove all handles with a size smaller than a user defined threshold. Cutting along such a cycle helps retain as much as possible of the fine geometrical detail of the model.

Volumetric Modification. To remove a handle, we alter the scalar values of the volume, thus indirectly modifying the isosurface. Rather than attempting to repair the defects on a mesh already extracted from the volume [Guskov and Wood 2001], our approach operates on the volume representation directly. The natural ordering of the volume data in planar slices allows for an efficient ordered traversal of the data. This ordering facilitates the development of out-of-core algorithms to process very large datasets. Another advantage of operating directly on a volume is that even when we alter discrete grid samples to simplify the topology, the final isosurface will remain a manifold. Finally, since our algorithm creates a topologically simplified volume, this volume can then be used for surface extraction or other applications that depend on a topologically accurate volumetric representation, for example cortex labeling [Jaume et al. 2002] or 3D morphing.

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 $885 \times 709 \times 736$ grid, and much larger models now exist [Levoy et al. 2000]. In order to remove excess topology from these scanned models, the volume is processed using a sweep method so that the data access pattern is highly regular and only a few slices need to be stored in memory at any given time. While our algorithm builds a data structure on-the-fly, typically of size $O(n)$, the method is input-sensitive (see Section 3.2 for details). For typical scanner data, for example, all the example models and any model without the pathological case of handles spanning

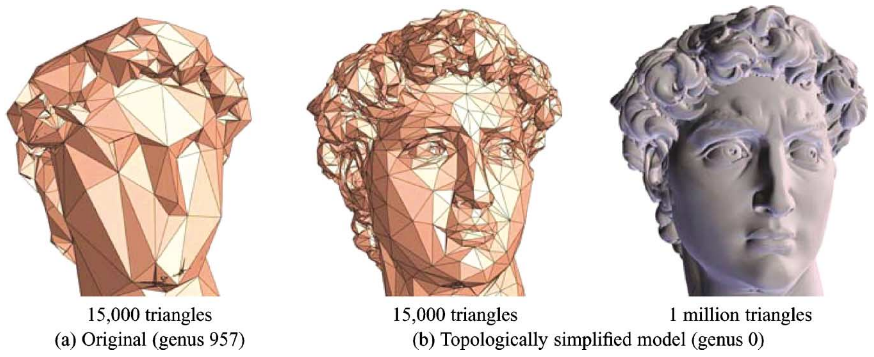

Fig. 3. Comparison of progressive meshes of the David model before and after topology simplification. On the left, many triangles are wasted representing invisible topological artifacts. Note that the topologically simplified mesh on the right shows no visible artifacts from the topology simplification process.

the entire extent of the volume, our algorithm can typically be executed out-of-core even on personal computers with limited memory capacities.

1.1 Related Work

Reeb graphs and Discrete Morse Functions. Our approach is related to Morse theory, which examines the relationship of the critical points of a smooth function defined on a smooth manifold to the topology of the manifold [Milnor 1963]. Since we are only interested in identifying handles on a surface, we do not attempt to identify every critical point on the surface (e.g., we are not interested in maxima and minima). Similar to the work of Axen [1999], Guskov and Wood [2001], and Wood et al. [2000], we examine how the trace of a discrete wavefront, induced by a distance function, changes as it progresses over a 2-manifold. Regions where the wavefront splits and merges as it passes over the surface are related to the location of saddle points on the surface. We refer the interested reader to the work of Axen [1999] for a thorough description of the use of a height function as a Morse function on triangulated manifolds and techniques to isolate exact critical points. Rather than find exact critical points on the surface, we track regions where a discrete wavefront splits and merges as it traverses the surface. These regions are well defined given our reconstruction rules (see Section 2) and regions with nonisolated or degenerate critical points are handled appropriately.

Reeb graphs [Reeb 1946] are often used to analyze surface topology. Given a scalar function f defined on a surface, a Reeb graph tracks the connected components of the pre-image of the function. For example, 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. A typical Reeb graph will have a node for each contour and an edge for each connected component of the surface between the contours (see Figure 4 and Section 2.1). Shinagawa et al. [1991] use this framework for the reconstruction of surfaces from contours. Axen and Edelsbrunner [1998], Hilaga et al. [2001], and Wood et al. [2000] analyze Reeb graphs induced by a geodesic distance function with respect to a seed point. Reeb graphs have alternatively been used to construct the medial axis of polyhedral objects [Lazarus and Verroust 1999], to compute seed sets for tracing isosurfaces [Carr et al. 2003; Kreveld et al. 1997], to recognize shapes [Hilaga et al. 2001], and to remesh surfaces [Attene et al. 2001; Wood et al. 2000].

Mesh-Based Topology Simplification. Guskov and Wood [2001] remove topological noise from already extracted meshes. They repeatedly grow ϵ-balls over the surface, and remove any handle enclosed within such a ball via mesh surgery. While this approach is simple and effective, their definition of topological

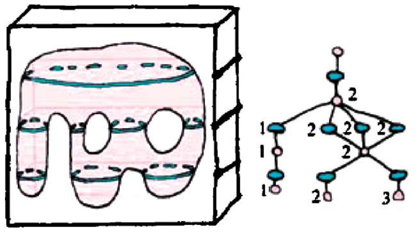

Fig. 4. An isosurface and its corresponding Reeb graph for a bottom-to-top sweep of the volume. In the graph, contour nodes are shown in blue, and ribbon nodes in pink. Also shown on the graph are component labels, here represented as numbers.

feature size fails to detect long thin handles, since they do not fit in a small ball. In addition, we prefer to operate on the volume data, since an isosurface will always remain a manifold after topological repair.

Using the concept of alpha hulls, El-Sana and Varshney [1997] reduce surface genus by retessellating small handles in a model. Their algorithm creates candidate tessellation regions by heuristically detecting crease edges in mechanical CAD models. One difference with our approach is that we evaluate whether to retain or simplify a handle based on a geometric measure defined on the handle itself.

Volume-Based Topology Simplification. Nooruddin and Turk [2003] 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 handles to close. However, the operators affect the entire volume, resulting in the smoothing of geometry and thus loss of fine detail. Extensions to this approach were recently presented by Bischoff et al. [2002]. We prefer a more targeted approach that measures the sizes of the handles and preserves geometrical detail in regions away from topological artifacts.

Shattuck and Leahy [2001] address the specific problem of constructing a genus-zero model of the human cortex from MRI scans for use in cortical flattening and mapping. Their method removes all handles without regard to size, and always breaks handles along axis-aligned planes (Figure 12 shows an example where their strategy would fail to find a short loop to break the handle).

Recent work by Szymczak and Vanderhyde [2003] addresses extraction of topologically simple isosurfaces. This work orders voxels based on distance from the desired isosurface and carves away voxels, allowing only a limited number of topology changes during the carving. While this method is fast, it does not allow much control over the changes to the final isosurface. For example, this method would simplify a long thin handle by filling the interior, instead of breaking the thin waist of the handle. Related methods include topology-preserving level set methods [Han et al. 2001], which constrain the topology of the implicit curve or surface during segmentation.

Edelsbrunner et al. [2002] use alpha complexes to generate a filtration, a history of the evolution of complexes. A filtration allows for a combinatorial definition of topological feature size. Zomorodian expands this work in his thesis [2001] and presents a practical algorithm to apply topology simplification to a variety of topological spaces. This work could be applied to filter small handles from volumetric data. To properly remove the handles from volume data requires the construction of a three-dimensional Morse complex. Recent work addresses this issue [Edelsbrunner et al. 2003].

Model Simplification. Several algorithms simplify topology as a byproduct of model simplification, for example, Garland and Heckbert [1997], He et al. [1996], and Popovic and Hoppe [1997]. These methods can result in nonmanifold structures which would hinder parametrization as much as the original topological artifacts. In addition, since these methods simultaneously simplify geometry and topology, removing topological artifacts invariably involves loss of geometrical detail. Our focus is on simplifying topology while preserving geometrical detail as much as possible.

Cut Graphs. Our approach shares common themes with work on cutting a surface into a single topological disk [Colin de Verdière and Lazarus 2002; Dey and Schipper 1995; Erickson and Har-Peled 2002; Gu et al. 2002; Kartasheva 1999; Lazarus et al. 2001; Vegter and Yapp 1990]. These approaches typically analyze the topology of the entire surface. Erickson and Har-Peled [2002] propose a greedy algorithm to compute a nearly minimal *cut graph*. A cut graph is a collection of edges that cuts a surface into a disk. Their approach must analyze the entire surface, compute a cut-graph for the surface and then find the nearly-shortest essential loops one at a time. We search for nonseparating cuts only in a local region, one handle at a time. While we are not guaranteed to find the globally minimum cuts, our algorithm can operate out-of-core and generate fast approximations.

2. ALGORITHM DETAILS

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 (e.g., Lachaud [1996], Lorensen and Cline [1987], or Wyvill et al. [1986]) defines a set of polygons. Each cube may have up to 4 polygons. The polygons from all cubes together form a discrete representation of the isosurface. We assume that the connectivity of the polygons is predetermined, for example, by some table-driven isosurface generation algorithm. We use the connectivity rules of Lachaud [1996] due to the fact that they produce a closed oriented surface without singularities or self-intersections. For isosurface generation, given a volume defined by a scalar function $F(x)$, the interior is defined as $F(x) < 0$, while exterior is defined as $F(x) \geq 0$.

Our approach uses an axis-aligned *sweep* through the volume which visits the grid data along parallel data *planes*. The isosurface intersects each such plane along a set of *contours* (oriented closed polylines) as depicted on Figure 4 and 8. 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. In our setting, an isolated handle is a connected surface region which is a union of a subset of adjacent ribbons with $g = 1$.
Our approach can be summarized as:

—sweep through the volume to *encode* the topology in an augmented Reeb graph (see Section 2.1).

—*isolate* handles using the augmented Reeb graph (see Section 2.2).

—for each handle found, *measure* its size (see Section 2.3).

—if the size is sufficiently small, *remove* the handle (see Section 2.4).

We now present each of these steps in more detail.

2.1 Encoding the Topology in an Augmented Reeb Graph

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

Reeb graphs have commonly been used to analyze surface topology. For example, in the work of Shinagawa [1991], a Reeb graph is constructed from predetermined cross-sectional contours defined by a height function. In such a setting, *a priori* information about the topology of the initial shape is required to guarantee that the Reeb graph exactly matches the topology of the input shape. In our setting, the surface connectivity and topology of the surface between planar cross-sections is determined

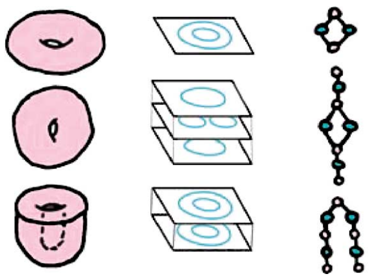
Fig. 5. Example of surfaces and their associated contours and augmented Reeb graphs. The examples are: a torus on its side, an upright torus, and a bowl-like surface.

by the connectivity information of the ribbons within z intervals. By correctly traversing and encoding the surface connectivity, we can encode the topology of the isosurface in a Reeb graph.

Traversal. Reeb graphs have also been used to encode the topology of a surface by encoding the relationship of critical points of the surface [Reeb 1946]. Critical points correspond to the topology of a surface and for discrete 2-manifolds can be identified using Morse theory. For example, a distance function defined on the vertices of the isosurface which assigns each vertex of the isosurface a unique distance value can be used to identify critical points such as saddle points [Axen 1999]. These saddle points could subsequently be used to localize handles in the surface. Likewise, by using a face-by-face distance function and examining the boundary of the distance function for discrete intervals, critical faces adjacent to handles can be identified [Guskov and Wood 2001]. For the volume setting, we could take a similar approach and traverse each voxel that the isosurface intersects and analyze the current boundary of the reconstructed surface to identify critical polygons. Such a polygon-by-polygon traversal and analysis of the boundary of the distance function guarantees that the topology of the isosurface would be correctly identified (using a similar combinatorial proof to that used in Guskov and Wood [2001]). Each split and subsequent merge of the boundary of the distance function would correspond to a cycle in the Reeb graph. We employ a related method with an optimization for the volume setting.

Specifically, we adopt a modified approach, which analyzes the boundary of a distance function traversal at discrete z intervals of the volumetric grid. Such planar slices are a natural choice for our setting as an ordered traversal through the slices allows for the out-of-core processing of the volume data. In contrast, a polygon-by-polygon traversal, that is, a breadth-first traversal of the polygons, requires highly irregular data access into the volume. Additionally, the Reeb graph will be significantly less dense by analyzing the distance function at discrete z intervals. We therefore analyze the boundary at discrete z intervals and only use a polygon-by-polygon traversal when necessary (details follow under the *Traversal revisited* heading).

For our setting, we construct an *augmented* Reeb graph which stores geometric information for each contour *and* ribbon of the traversal (see Figure 5). Contour nodes in the augmented Reeb graph correspond to a distance function traversal of the surface, and cycles in the graph correspond to handles. These cycles and the geometric information stored in the ribbon nodes can be used to later isolate handles in the model. Reeb graph construction occurs concurrently with the traversal of the volume.

Constructing Ribbons and Contours. We analyze the isosurface one slice at a time. 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 within the slice to find connected sets of polygons (in the slice) and edges (in the planes), respectively. Contours are constructed by searching from an arbitrary edge in the plane until the contour is closed. Ribbons are constructed by running a breadth-first traversal in the slice starting with the polygons adjacent to one contour and ending with the polygons

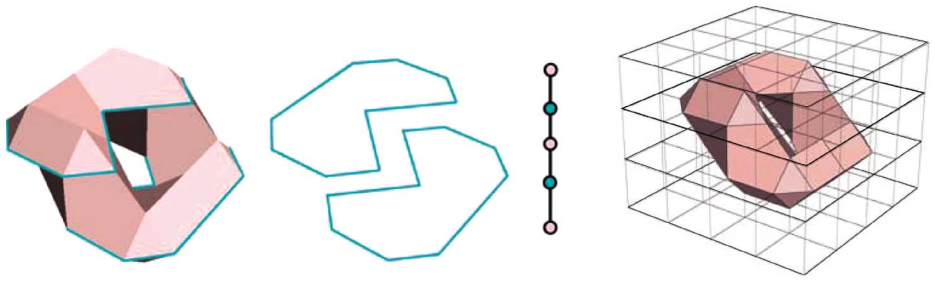

Fig. 6. Example of an intraribbon handle. This torus tilted at an angle is formed by two "C" shaped contours. As shown in the figure on the left, the Reeb graph does not contain any cycle. On the right, we see the volume grid overlaid on this isosurface.

adjacent to the previous contour. 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 to 8. Wood [2003] contains pseudo-code and more details on ribbon, contour, and augmented Reeb graph construction.

Traversal Revisited. The optimization used by our method to analyze the distance function at discrete z intervals has consequences. Specifically, when a handle is entirely contained within a ribbon, (i.e., within a slice of the volume), it is not initially detected during the traversal. One such *intraribbon handle* is generated by the 6×6 cube grid shown in Figure 6. In practice, these intraribbon handles occur for 1–10% of the slices of a volume. With a simple modification to our algorithm we can detect these intraribbon handles and modify our traversal to ensure they are identified. We detect these intraribbon handles by computing the Euler characteristic of the isosurface within each z interval. If the slice has nonzero genus, it contains handles. For such cases, we analyze the distance function on a polygon-by-polygon basis within the relevant slice. Specifically, we use a breadth-first traversal over the polygons in the current slice, similar to the method used in Guskov and Wood [2001]. Such a traversal will guarantee that we identify the topology of the intraribbon handles. In addition, confining the per-voxel traversal to the slice keeps our surface access local, allowing our method to support out-of-core processing. For intraribbon handles, contour construction is modified to be defined by the polygon breadth-first-traversal for the slice so that the Reeb graph will correspondingly encode the topology of the intra-ribbon handle. As a final check, we confirm overall correspondence of the augmented Reeb graph for each interval by ensuring a match between the Euler characteristic of the surface traversed thus far with the number of cycles in the Reeb graph.

2.2 Isolating Handles Using the Augmented Reeb Graph

The volume traversal and augmented Reeb graph construction proceed as we sweep through the data. Cycles in the augmented Reeb graph correspond to handles which are detected incrementally. This progressive detection allows for handle removal to occur concurrently during the sweep. Our approach is as follows:

Finding Cycles in the Reeb graph. To detect a handle, the algorithm needs to locally differentiate in the Reeb graph between the merging of (a) two previously *disconnected* components (i.e., a lone saddle point) and (b) two previously *connected* components (i.e., a handle forming from the second of a pair of saddle points). Both of these events are encoded in the Reeb graph by the merging of two contours to one ribbon, see Figure 4 for an example of both cases. To distinguish between these two cases we associate a label with both ribbons and contours, that identifies the connected component to which they belong. In our setting, the only way that a Reeb cycle can form is when two contour nodes in the previously visited plane are added to a single ribbon node in the Reeb graph. When adding such graph-edges, we test whether the two contour nodes have the same component label. If so, they belong to the same connected

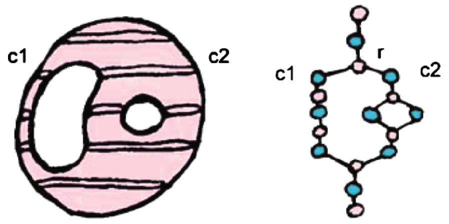

Fig. 7. Example of a surface and its augmented Reeb graph with adjacent handles. The ribbon r has the previous contours $C = \{c1, c2\}$. When we discover the Reeb cycle associated with contours $c1$ and $c2$, we construct the cycle path by finding the shortest path *from* $c1$ *to* $c2$.

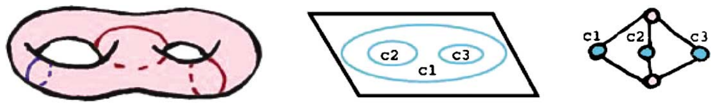

Fig. 8. A two-holed torus and its associated planar cross-section and augmented Reeb graph. To find the locally shortest nonseparating cut (shown in blue), we explore all pairs of child contours with the same component label (i.e., $\{c1, c2\}$, $\{c1, c3\}$ and $\{c2, c3\}$ where the minimal length nonseparating cut is associated with $\{c1, c2\}$).

component and a Reeb cycle is formed. In any case, after the graph-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 [Cormen et al. 1990], taking negligible time.

When a Reeb cycle is detected, we need to isolate the associated handle in the surface. For the purpose of topology simplification, we wish to isolate handles that are geometrically localized in the surface. To isolate handles for each Reeb cycle that is detected, we perform a breadth-first search through the graph to find the shortest graph cycle, starting from one of the like-labeled contour nodes, for example, $c1$ to the other $c2$ as shown in Figure 7. The Reeb cycle path consists of alternating ribbon and contour nodes and defines a *handle*. Note that for a given z interval, a ribbon may have k child contours with the same component labels, with $k \geq 2$. Although the local genus of the surface is $k - 1$ we need to consider all pairs of child contours as this allows us to test each of the reasonable ways to isolate a handle. For example, in Figure 8, in order to identify the best nonseparating cut, we must consider all pairs of child contours. Such a case can correspond to a region with a degenerate critical point, for example, possibly a region with a multiple saddle. In such a case, our method of checking the possible combinations of child contours relates to splitting a multiple saddle into simple saddles (similar to Zomorodian [2001]). Typically ribbons only have two child contours and the need to explore multiple pairs of child contours is infrequent.

The following pseudo-code summarizes the key parts of the algorithm for detecting cycles and isolating handles (see Figure 7):

```
function Add_ribbon_to_Reeb_graph(ribbon $r$, Reeb Graph $G$)
    Add ribbon $r$ as node in Reeb graph $G$.
    label($r$) := unique_label().
    Identify all 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 Reeb 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 $r$.
```

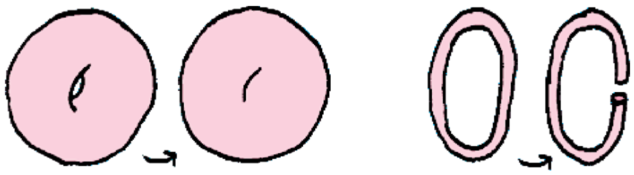

Fig. 9. Two ways of removing a handle, illustrated on two tori. The “fat” torus is best repaired by collapsing the handle, and the “skinny” torus is best repaired by pinching the handle.

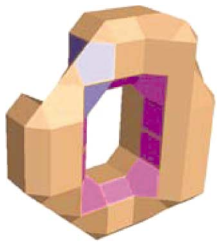

Fig. 10. A torus with the Reeb and cross loops shown. The Reeb loop is shown in magenta and the cross loop is shown in blue.

2.3 Measuring Handle Size

Recall that a cycle in the Reeb graph identifies a subset of ribbons forming a handle. In general to remove a handle, we reduce the number of nonseparating cycles for the surface. By collapsing a nonseparating cut for each handle, we conceptually either fill in the interior of a handle or we pinch open the handle. Local surface geometry determines which method is more appropriate, as illustrated in Figure 9. Both methods are, *in fact*, the same operation applied to two different nonseparating cuts. We call this operation *loop closure*. Topologically, the loop closure operation collapses the loop to a single point, removing the handle. In terms of geometry, 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 implementation of this operation on our discrete grid volume is discussed in the next section.

Two Nonseparating Cuts. Given a handle, to choose the more appropriate loop closure operation, we compute two nonseparating cuts. For the sake of discussion we distinguish and name these loops depending on the orientation of our Reeb graph:

—the *Reeb loop* is a nearly shortest loop around the Reeb cycle, and

—the *cross loop* is a nearly shortest loop transversal to the Reeb loop.

On the irregularly shaped torus, shown in Figure 10, the Reeb loop is shown in magenta, and the cross loop is shown in blue. There are many possible nonseparating cuts for a given handle. Since our goal is to simplify the topology in a way that minimizes geometric changes to the volume, we attempt to find tight fitting loops of short length.

We find the Reeb loop by constructing a nonseparating cut that matches the Reeb cycle. We can do this by cutting the Reeb cycle and then finding the shortest path from one side of the cut to the other. Observe that *any* one of the contours in the Reeb cycle are nonseparating curves that can be used to cut the Reeb cycle. Intuitively, this corresponds to cutting along a contour of the handle to open it into an cylinder open on both ends, see Figure 11. We construct a locally short nonseparating cut that follows the Reeb cycle by computing the shortest path from one side of such a contour to the other. In the

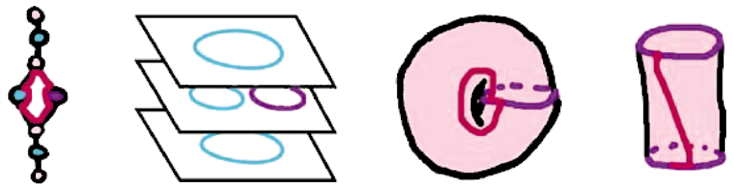

Fig. 11. Illustration of the process of identifying the Reeb loop for a torus. In this case, we are searching for a loop on the surface of the torus that corresponds to the red cycle in the augmented Reeb graph shown on the left. The search is started from one of the contours in the Reeb-cycle, shown in purple in the middle two images.

intuitive setting of the cylinder, this corresponds to computing the shortest path from the top of the cylinder to the bottom. This loop is locally nearly minimal because we *close* the loop by only traversing along the contour. We guarantee that the loop that we find matches a given Reeb cycle by restricting the area that we search for the loop to the region of the isolated handle.

We construct the cross loop in a similar manner. Starting from one side of the Reeb loop, we compute the 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.

Measure of Handle Sizes. From these two nonseparating cuts, we can now derive a measure of the handle. We generally define the handle size to be the *smaller* of the Reeb loop length and the cross loop length. 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 (i.e., by only changing nodes in the volume with $F(x) < 0$ or vice-versa). From the orientation of any contour in the Reeb graph cycle, one can determine whether the ribbon cycle encloses a void ($F(x) \geq 0$) or encloses material ($F(x) < 0$) and exclude the appropriate loop if desired.

As a measure of loop size, we chose the 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, for example, the area of the spanning minimal surface. We have chosen loop length because on our examples this typically was a tighter measure than area. The user may specify an area metric instead of length if this is deemed more appropriate for a particular application.

2.4 Removing 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. Since the loop is generally nonplanar, 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 [Kaufman 1987], depending on the orientation of the loop (discussed in Section 2.3). This rasterization technique both collapses and pinches off handles through insertion of a thin wall (see Figure 12). The modified isosurface, once extracted from the volume data, is guaranteed to remain a manifold and to have no self-intersections.

There are a few potential problems to consider. Topologically, the handle can always be cut along a nonseparating cut, reducing the genus of the model. However, depending upon how the surface is embedded in $\mathbb{R}^3$ the fan of triangles closing the nonplanar 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 has never occurred since we only simplify *small* handles coming from either measurement

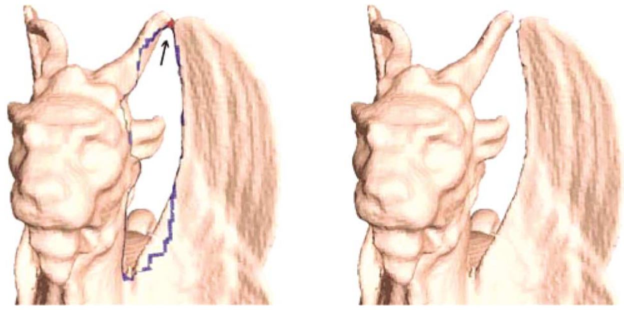

Fig. 12. Close-up of the feline mesh with the Reeb loop shown in blue, and cross loop shown in red. The right image shows our algorithm's output after collapsing the cross loop of the handle.

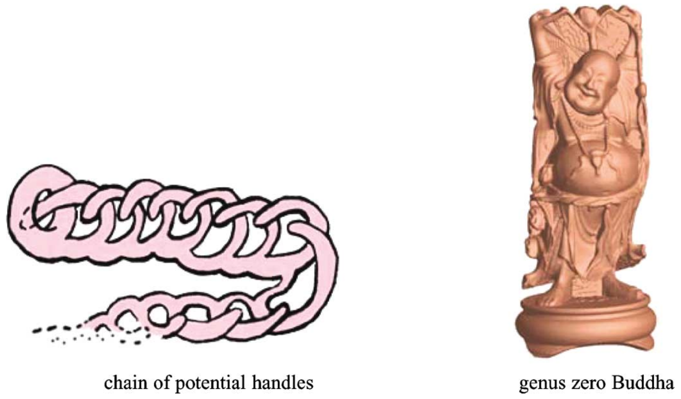

Fig. 13. Loop closure consequences: on the left is an illustration of a conceptual setting where loop closure could generate new handles. On the right is an image of the results of applying loop closure to all handles in the Buddha model (now genus zero).

imperfections or misalignment of scans. At worst, the loop closure could introduce additional handles. To address this, we locally rebuild the Reeb graph after a loop closure operation. If any new components or handles were introduced in the previous simplification step, they will appear in the reconstructed Reeb graph and be subsequently processed.

In theory, the introduction of new handles could cause termination problems if each collapse always created a new handle. See Figure 13 of a chain consisting of one handle and tails threaded through one another: collapsing the hole(s) creates a new handle. In a discrete volume grid this kind of cascading handle creation is limited by the size of the features that can be represented by the grid samples. Furthermore, in practice, the issue of obstructions has never caused the creation of new handles for our approach when simplifying small extraneous topology; even for large loops that do not contain obstructions, our simple closure routine performs as expected (see Figure 13). Finally, if some pathological volume did have termination issues, the algorithm can be modified to apply loop closure operations that either only add material, or only remove material as explained in the previous section. Since the volume data is discrete, this will guarantee termination, that is, even the most twisted configuration could be simplified by filling in the entire volume.

3. RESULTS AND DISCUSSION

We have run our topology simplification algorithm on a number of volumes as shown in Table I. The Buddha, dragon, feline, and David models are from laser range scans at Stanford University. The brain

Table I. Quantitative Results

Model	Grid size	#Faces	Thresh. size ℓ	Genus before	Genus after	Loops collapsed	#Intra-ribbon	Timing (minutes)
David	885 × 736 × 709	15,244,302	166.5	1063	0	1063	76	87.5
Buddha	400 × 400 × 950	4,736,292	9.5	106	6	100	26	6.5
Dragon	500 × 714 × 324	3,222,612	46.5	60	1	59	18	3.8
Brain1	125 × 255 × 255	688,248	32.5	366	0	366	6	2.8
Brain2	125 × 255 × 255	452,050	14.5	21	0	21	6	0.7
Brain3	125 × 255 × 255	529,012	10.5	41	0	41	4	0.6
Brain4	125 × 255 × 255	699,566	14.5	50	0	50	7	1.7
Feline	332 × 148 × 316	653,922	4.5	6	2	4	1	0.2

The handle threshold size ℓ is expressed in units of cube edge size. Times are shown in CPU minutes for a Pentium Xeon 2 Ghz. All values listed are for the entire volume, that is, for the surface and any spurious disconnected components in the volume data.

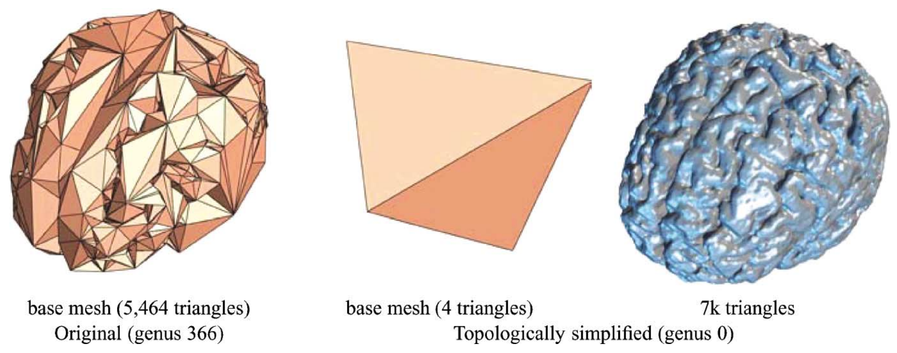
base mesh (5,464 triangles)
Original (genus 366)
base mesh (4 triangles)
7k triangles
Topologically simplified (genus 0)

Fig. 14. Comparison of the base meshes of progressive meshes on a brain model (MRI).

models are from MRI scans from the Harvard Medical School [Kikinis et al. 1996]. We have demonstrated the practicality and robustness of our algorithm using convoluted geometry (Figure 14) and large volumes (Table I). Since our algorithm locally reconstructs the Reeb graph after every topological change, it guarantees that we are accurately identifying all of the topology of the surface, even as its topology evolves. All of our experiments have performed as expected and removed all handles with length less than ℓ.

The timing for our algorithm depends on the size of the volume and on the number of 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, see Table I.

The histogram in Figure 19 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 19). 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 in Figure 15 has predominantly small handles, one of its spurious external surface components has a handle of length 46.

3.1 Applications

Topology simplification facilitates many surface operations:

—Fewer triangles are wasted to encode topological defects during *mesh simplification*, as shown in Figures 2, 3 and 14 using the progressive mesh representation of Hoppe [1996]. Consequently, coarser meshes can be created, and geometrical quality is improved at all levels of detail.

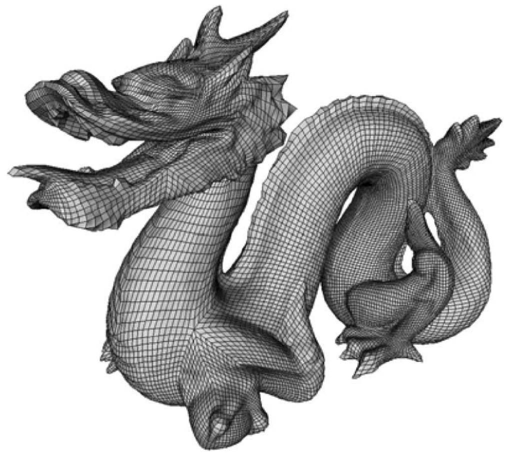

Fig. 15. A remesh of the genus 1 dragon. Given the difficulty of achieving a high-quality parametrization for high-genus models, remeshing the original dragon with genus 46 would be quite challenging and require numerous elements in the base domain.

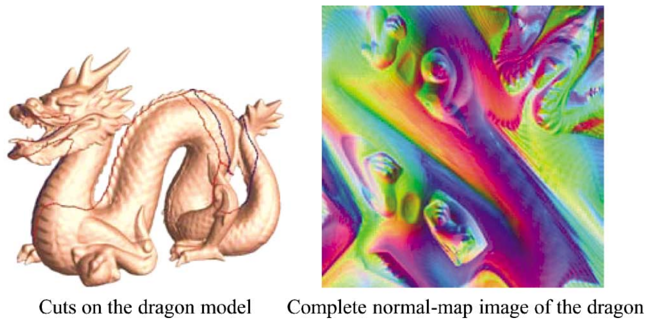

Fig. 16. Geometry Image of the genus 1 dragon model, showing the cuts used to parametrize the entire model onto a single chart. Parameterizing the 500K face dragon onto a unit square would cause large distortion with the original genus 43 model.

—Better surface parametrization improves *texture mapping*, as shown in Figure 18 using the method of Sander et al. [2001]. Fewer charts are necessary to partition the surface, which results in a nicer parametric domain.

—Removal of topological defects greatly facilitates *remeshing*, as shown in Figure 15 using the method of Guskov et al. [2002]. The remesh has nice regular face sizes and allows for efficient progressive geometry compression [Khodakovsky et al. 2000] as well as many other semi-regular geometry processing algorithms [Schröder and Sweldens 2001]. The topologically simplified volumes can also be more readily used for semi-regular mesh extraction [Wood et al. 2000]. Applications such as geometry images [Gu et al. 2002], as shown in Figure 16, which remesh the entire surface to a completely regular structure by parametrizing the surface to a disk, would suffer from large distortion if applied to surfaces with many topological artifacts.

—Medical applications benefit from operating on topologically accurate volumes. For example, the approach of Jaume et al. [2002], requires genus zero brain models and volumes to propagate cortex labels correctly, (Figure 14 shows brain data with excess topology.) Using our method to obtain topologically accurate volumes, Jaume et al. are able to propagate cortex labels from one labeled volume to others. As shown in Figure 17.

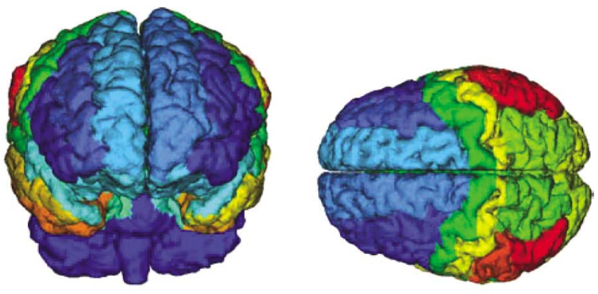

Fig. 17. Two different views of a brain model in which cortex labels have been propagated from one brain to the next through the method of Jaume et al. [2002].

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 scatterplot or histogram (Figure 19). By looking at the relative sizes of 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.

Algorithm Time Complexity. The overriding time complexity term for the algorithm 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 $O(n^2)$ polygons, 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, our algorithm can be qualified as input-sensitive since the most significant processing time in practice is associated with each handle discovered and its subsequent measurement and possible removal. This processing time is dominated by the time complexity of the breadth-first searches run to compute the length of the Reeb and cross loop. For each loop, the complexity is $O(d^2)$, where d is the minimal length of each loop. Intuitively, this measure corresponds to the area of the surface covered during the search.

A final concern for time complexity is the fact that for every handle that is simplified, we must reconstruct the Reeb graph locally to account for the resulting changes. The cost of this reconstruction is on the order of $\ell \times n$ (where ℓ slices with n polygons need to be reconstructed after the topology changes).

Algorithm Space Complexity. Perhaps more important are the space requirements. In practice, we *only* keep a small number b of slices of the volume in memory at any time to limit I/O access, making the algorithm viable even for low-end computers. The choice of buffer size b is flexible and can change due to the size of the volume and the memory resources available. We used a buffer of size $b = 50$ for all of the example volumes. All of our operations have strong spatial coherence, and in practice, we found that we rarely reload the same slice more than twice.

In addition to the buffer of volumes slices, the algorithm stores the Reeb graph $O(n)$ for the surface, and some limited local number of polygons. Typically, we store $b \times n$ polygons, since polygons below the bottom of the current buffer are freed to minimize memory use. However, the algorithm is input-sensitive and the worst case situation is that of a space filling handle spanning the entire extent of the volume. In this worst-case setting, the memory requirements may reach $O(n^3)$ to store all of the polygons in rare, pathological volumes. For all the example models, polygon storage was within $O(n)$ and our algorithm simplified the topology in an out-of-core fashion.

Local polygons are stored in order to compute loops for any handle that is being processed. They can also quickly be recomputed using a table look-up. In general, computing the Reeb loop for a handle

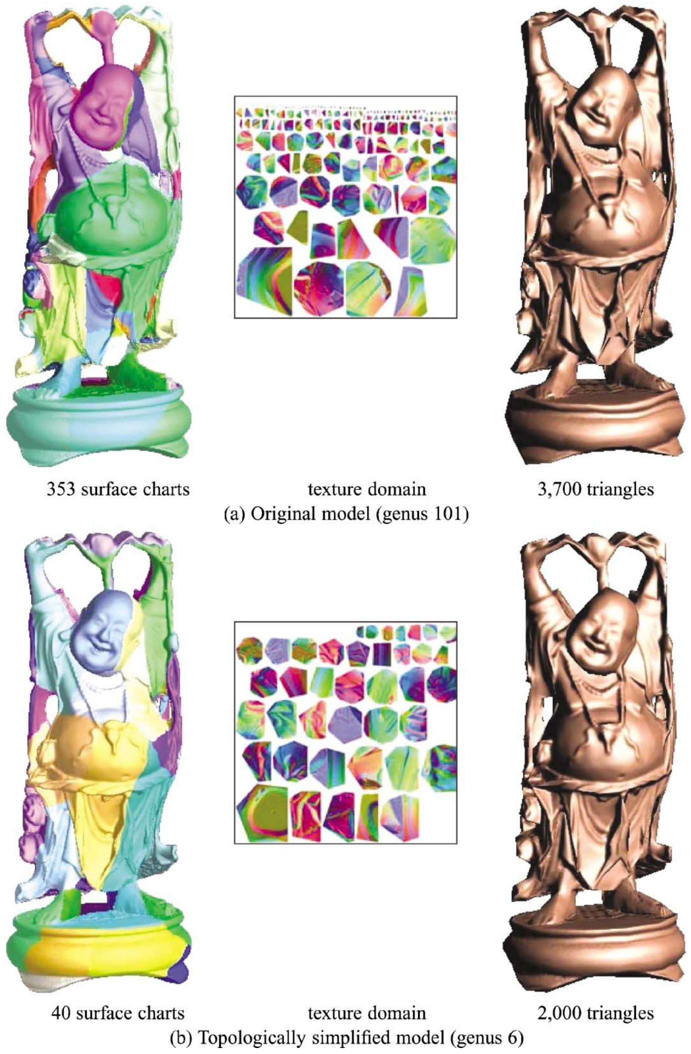

Fig. 18. Comparison of normal-mapping progressive meshes before and after topology simplification. Both models refer to 512 × 512 texture images. The topological complexity of the original model requires many more parametric charts, shown in pseudo-color. The resulting fragmentation of the parametric domain restricts simplification.

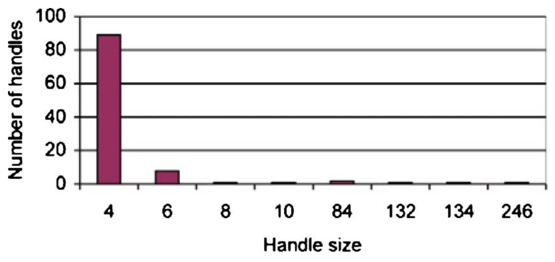

Fig. 19. Histogram of handle sizes for the original scanned Buddha model. Recall that handle size is the smaller of the Reeb and cross loop lengths.

requires access to the polygons 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.

Since polygons below the *bottom* of the current buffer are freed to minimize memory use, Reeb and cross loop computation for handles with length $>b$ may require reloading the buffer with previous slices of the volume that have already been flushed from memory. This reloading is only necessary to re-allocate polygons and all computations can be done in sequence, locally on the volume data. Reloading previous buffers is rare since few handles have large extents. For instance, the Buddha model required the reloading of previous buffers for only 5 (the largest ones) of the 106 handles found, while the brain models never required any reloading.

4. SUMMARY AND FUTURE WORK

We have introduced an algorithm for automatically removing 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 artifacts is important for many subsequent modeling operations.

We would like to extend this work to other settings besides volume data and are presently exploring techniques to use a geodesic distance to construct a Reeb graph for arbitrary meshes. This work involves exploring loop closure operations that avoid self-intersections.

For larger handles, it may be desirable to use more accurate approximations of true geodesic non-separating cuts rather than the discrete graph approximation. More generally, we are interested in exploring alternative methods for measuring 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 preprocess the volume to remove topological artifacts for a range of isosurface values. For such a setting, the work of Zomorodian [2001] is promising.

ACKNOWLEDGMENTS

We thank: the Computer Graphics Lab at Stanford University for the David model from the Digital Michelangelo library; Drs. Kikinis, M. Shenton, R. McCarley, and F. Jolesz of the Harvard Medical School for the brain models; Jacques-Olivier Lachaud for his isosurface generation code; Nathan Litke and Andrei Khodakovsky for their assistance with remeshing; Sylvain Jaume for cortex labeling collaboration; Steven Gortler for conversations about topology and Eitan Grinspun for his assistance with editing.

REFERENCES

ALEKSANDROV, P. 1956. *Combinatorial Topology*. Vol. 1. Graylock Press.

ATTENE, M., BIASOTTI, S., AND SPAGNUOLO, M. 2001. Re-meshing techniques for topological analysis. In *International Conference of Shape Modeling and Applications* (Genova, Italy). IEEE Computer Society Press, Los Alamitos, Calif., 142–151.

AXEN, U. 1999. Computing morse functions on triangulated manifolds. In *Proceedings of the SIAM Symposium on Discrete Algorithms (SODA)*. 850–851.

AXEN, U. AND EDELSBRUNNER, H. 1998. Auditory morse analysis of triangulated manifolds. In *Mathematical Visualization*, H.-C. Hege and K. Polthier, Eds. Springer-Verlag, Berlin, Germany, 223–236.

BISCHOFF, S. AND KOBBELT, L. 2002. Isosurface reconstruction with topology control. In *Proceedings of Pacific Graphics*. 246–255.

CARR, H., SNOEYINK, J., AND AXEN, U. 2003. Computing contour trees in all dimensions. *Comput. Geom. 24*, 2, 75–94.

COLIN DE VERDIÈRE, É. AND LAZARUS, F. 2002. Optimal system of loops on an orientable surface. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science* (Vancouver, B.C., Canada). IEEE Computer Society Press, Los Alamitos, Calif., 627–636.

CORMEN, T., LEISERSON, C., AND RIVEST, R. 1990. *Introduction to Algorithms*. MIT Press, Cambridge, Mass.

CURLESS, B. AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH*. ACM, New York, 303–312.

DEY, T. K. AND SCHIPPER, H. 1995. A new technique to compute polygonal schema for 2-manifolds with applications to null-homotopy detection. *Disc. Comput. Geom. 14*, 93–110.

EDELSBRUNNER, H., HARER, J., NATARAJAN, V., AND PASCUCCI, V. 2003. Morse complexes for piecewise linear 3-manifolds. In *Proceedings of the 19th Annual ACM Symposium on Computer Geometry*. ACM, New York, 98–101.

EDELSBRUNNER, H., LETSCHER, D., AND ZOMORODIAN, A. 2002. Topological persistence and simplification. *Disc. Comput. Geom. 28*, 511–533.

EL-SANA, J. AND VARSHNEY, A. 1997. Controlled simplification of genus for polygonal models. In *Proceedings of IEEE Visualization*. IEEE Computer Society Press, Los Alamitos, Calif., 403–412.

ERICKSON, J. AND HAR-PELED, S. 2002. Optimally cutting a surface into a disk. In *Proceedings of SOCG*. ACM, New York, 244–253.

FRANCIS, G. AND WEEKS, J. 1999. Conway's ZIP proof. *Amer. Math. Monthly 106*, 393–399.

GARLAND, M. AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. *Proceedings of SIGGRAPH*. ACM, New York, 209–216.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry images. *Proceedings of SIGGRAPH*. ACM, New York, 355–361.

GUSKOV, I., KHODAKOVSKY, A., SCHRADER, P., AND SWELDENS, W. 2002. Hybrid meshes: multiresolution using regular and irregular refinement. In *Proceedings of the 18th Annual Symposium on Computational Geometry*. ACM, New York, 264–272.

GUSKOV, I. AND WOOD, Z. 2001. Topological noise removal. *Graph. Interf.*, 19–26.

HAN, X., XU, C., AND PRINCE, J. L. 2001. A topology preserving deformable model using level sets. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*. IEEE Computer Society Press, Los Alamitos, Calif., 765–770.

HE, T., HONG, L., VARSHNEY, A., AND WANG, S. W. 1996. Controlled Topology Simplification. *IEEE Trans. Visual. Comput. Graph. 2*, 2, 171–184.

HILAGA, M., SHINAGAWA, Y., KOHMURA, T., AND KUNII, T. L. 2001. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of SIGGRAPH*. ACM, New York, 203–212.

HILTON, A., TODDART, A. J., ILLINGWORTH, J., AND WINDEATT, T. 1996. Reliable surface reconstruction from multiple range images. *Fourth European Conference on Computer Vision I*, 117–126.

HOPPE, H. 1996. Progressive meshes. In *Proceedings of SIGGRAPH*. ACM, New York, 99–108.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. In *Computer Graphics (Proceedings of SIGGRAPH 92)*. ACM, New York, 71–78.

JAUME, S., MACQ, B., AND WARFIELD, S. K. 2002. Labeling the brain surface using a deformable multiresolution mesh. In *MICCAI*. 451–457.

KARTASHEVA, E. 1999. The algorithm for automatic cutting of three-dimensional polyhedron of h-genus. In *International Conference of Shape Modeling and Applications* (Aizu, Japan). IEEE Computer Society Press, Los Alamitos, Calif., 26–33.

KAUFMAN, A. 1987. Scan-conversion of polygons. In *Proceedings of Eurographics*. 197–208.

KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2000. Progressive geometry compression. In *Proceedings of SIGGRAPH*. ACM, New York, 271–278.

KIKINIS, R., SHENTON, M. E., IOSIFESCU, D. V., MCCARLEY, R. W., SAIVIROONPORN, P., HOKAMA, H. H., ROBATINO, A., METCALF, D., WIBLE, C. G., PORTAS, C. M., DONNINO, R., AND JOLESZ, F. A. 1996. A digital brain atlas for surgical planning, model driven segmentation and teaching. *IEEE Trans. Visual. Comput. Graph.*, 232–241.

KREVELD, M. V., VAN OOSTRUM, R., BAJAJ, C., PASCUCCI, V., AND SCHIKORE, D. 1997. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the 13th ACM Symposium on Computational Geometry*. ACM, New York, 212–219.

LACHAUD, J.-O. 1996. Topologically Defined Iso-surfaces. In *Proceedings of the 6th Discrete Geometry for Computer Imagery (DGCI)*. Lecture Notes in Computer Science, Vol. 1176. Springer-Verlag, New York, 245–256.

LAZARUS, F., POCCHIOLA, M., VEGTER, G., AND VERROUST, A. 2001. Computing a canonical polygonal schema of an orientable triangulated surface. In *Proceedings of the ACM SOCG*. ACM, New York, 80–89.

LAZARUS, F. AND VERROUST, A. 1999. Level set diagrams of polyhedral objects. In *Proceedings of the ACM Solid Modeling'99* (Ann Arbor, Mich.). ACM, New York, 130–140.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The digital Michelangelo project: 3D scanning of large statues. In *Proceedings of SIGGRAPH*. ACM, New York, 131–144.

LORENSEN, W. E. AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH*. ACM, New York, 163–169.

MASSEY, W. 1967. *Algebraic Topology: An Introduction*. Harcourt, Brace & World, Inc., New York.

MILNOR, J. 1963. *Morse Theory*. Princeton University Press, Princeton, N.J.

NOORUDDIN, F. AND TURK, G. 2003. Simplification and repair of polygonal models using volumetric techniques. *IEEE Trans. Visual. Comput. Graph. 9*, 2, 191–205.

POPOVIC, J. AND HOPPE, H. 1997. Progressive simplicial complexes. In *Proceedings of SIGGRAPH*. ACM, New York, 217–224.

REEB, G. 1946. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus Acad. Sci. de Paris*. 847–849.

SANDER, P., SNYDER, J., GORTLER, S., AND HOPPE, H. 2001. Texture mapping progressive meshes. In *Proceedings of SIGGRAPH*. ACM, New York, 409–416.

SCHRÖDER, P. AND SWELDENS, W., Eds. 2001. Digital Geometry Processing. Course Notes. In *ACM SIGGRAPH*. ACM, New York.

SHATTUCK, D. W. AND LEAHY, R. M. 2001. Automated graph based analysis and correction of cortical volume topology. *IEEE Trans. Med. Imag.* 1167–1177.

SHINAGAWA, Y. AND KUNII, T. L. 1991. Constructing a reeb graph automatically from cross sections. *IEEE Comput. Graph. Appl. 11*, 6, 44–51.

SZYMCZAK, A. AND VANDERHYDE, J. 2003. Extraction of topologically simple isosurfaces from volume datasets. In *Proceedings of Visualization 2003*. IEEE Computer Society Press, Los Alamitos, Calif.

VEGTER, G. AND YAPP, C. K. 1990. Computational complexity of combinatorial surfaces. In *Proceedings of the 6th Annual Symposium on Computational Geometry*. 93–111.

WOOD, Z. 2003. Computational topology algorithms for discrete 2-manifolds. Ph.D. dissertation. California Institute of Technology.

WOOD, Z., DESBRUN, M., SCHRÖDER, P., AND BREEN, D. 2000. Semi-regular mesh extraction from volumes. In *Proceedings of Visualization*. 275–282.

WYVILL, B., MCPHEETERS, C., AND WYVILL, G. 1986. Data structure for soft objects. *Vis. Comput. 2*, 4, 227–234.

ZOMORODIAN, A. 2001. Computing and comprehending topology: Persistence and hierarchical morse complexes. Ph.D. dissertation. University of Illinois at Urbana-Champaign.

Received April 2003; revised November 2003, February 2004; accepted March 2004