An Out-of-core Algorithm for Isosurface Topology Simplification

Zoë Wood	Hugues Hoppe	Mathieu Desbrun	Peter Schröder
Caltech	Microsoft Research	U. of So. Cal.	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 paper we present an efficient 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 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 by a short 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.

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

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). 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* [Curless and Levoy 1996; Hilton et al. 1996; Hoppe et al. 1992; Levoy and others 2000]. Similarly, CT or MRI produce data volumes from which isosurfaces are extracted [Lorensen and Cline 1987].

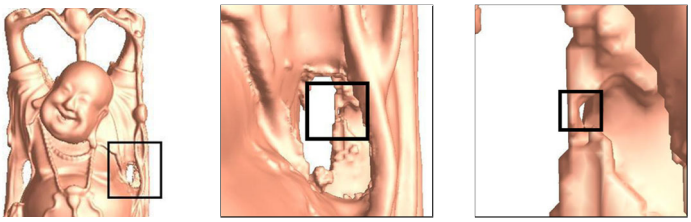
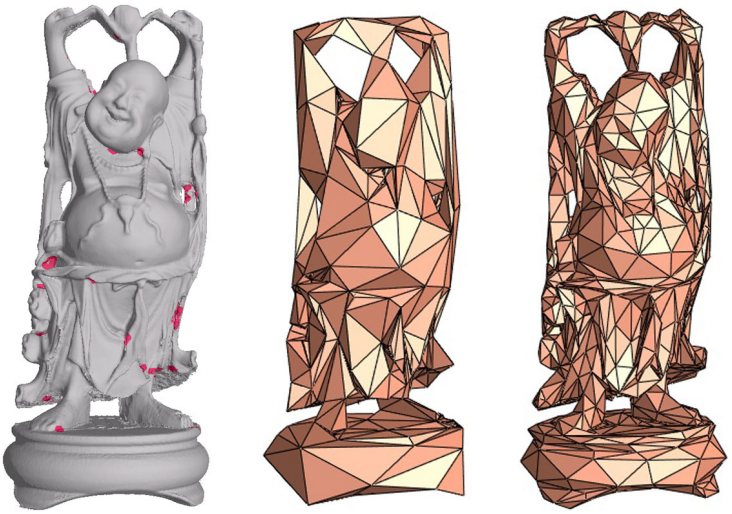


Fig. 1. Sequence of progressively closer views revealing an extraneous handle in the Buddha mesh.



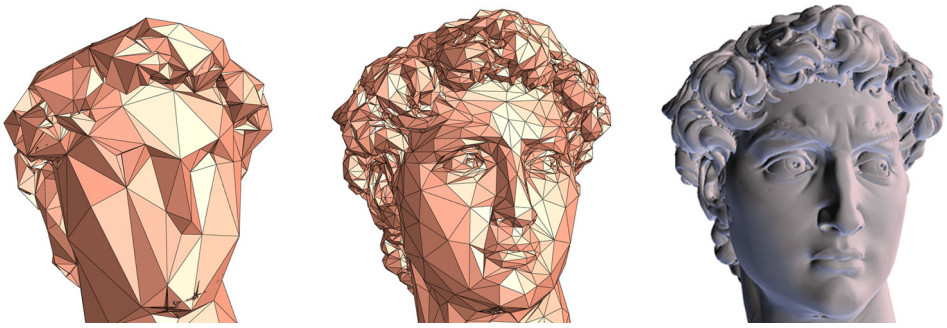
Genus 104 Genus 104 (2K triangles) Genus 6 (2K triangles)
Original scan Topologically simplified

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.

For surface reconstruction, one advantage of an isosurface representation is that it naturally supports models of arbitrary genus, *i.e.*, with any number of "handles." A *handle* is a toroidal region of the surface with genus $g = 1$ (various formal definitions exist, see for example, "Conway's ZIP proof" [Francis and Weeks 1999]). An example of an isosurface with arbitrary genus is the Buddha statue used in Figure 2 that 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 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 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.

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 [Shattuck and Leahy 2001; Jaume et al. 2002].

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 [Guskov and Wood 2001], our approach operates on the volume representation directly, as this offers advantages of efficiency and robustness. In terms of efficiency the advantage of the volume setting is the natural ordering of the data in the form of planar slices. This ordering allows



15,000 triangles 15,000 triangles 1 million triangles
(a) Original (genus 957) (b) Topologically simplified model (genus 0)

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.

us to develop *out-of-core* algorithms to process very large dataset. Operating directly on a volume is robust because even when we alter discrete grid samples in the volume to simplify the topology, the final isosurface *will remain a manifold* [Lachaud 1996].

Our algorithm identifies topology in the volume through the application of techniques associated with Morse theory [Milnor 1963]. The topology is coded in a Reeb graph [Reeb 1946], where cycles in the Reeb graph correspond to handles. See Section 1.1 for details. In order to measure the size of handles on the surface, we examine them one by one and consider cutting this region along a *non-separating cut*. A non-separating cut leaves the surface connected [Aleksandrov 1956]. By subsequently pinching each of the two open boundaries of such a cut to a point, the genus of the handle is reduced ($g = 0$). See Figure 11 for an example. Using the length of this cut as a measure of the size of the handle, we choose either to retain the handle or remove it. Our method sweeps through the volume grid to locate handles, compute their sizes, and selectively remove them, accessing only a small buffer of the volume at a time. The contributions of our method are the following:

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 and others 2000]. The algorithm is applied to such volumes using out-of-core methods. The volume is processed using a sweep method, so the data access pattern is highly regular. We encode surface topology as the sweep progresses using a Reeb graph, requiring only a few slices in memory at any time.

Fast identification of handles Handles are efficiently identified during the sweep, as cycles in the Reeb graph as it is incrementally constructed. We detect *all* handles during the sweep.

Handle size estimation and local repair Some models have genus that should be preserved, such as the handles formed by the Buddha's arms. We introduce a simple measure of handle size to be the length of a non-separating 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. Since properly-extracted isosurfaces are always

manifold [Lachaud 1996], operating on the volume is robust. In contrast, traditional "mesh surgery" must deal with issues of surface self-intersection and non-manifoldness. Since our algorithm creates a topologically clean 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.

Our goal is the removal of small excess topology. Applying our simplification method to large handles that are obstructed by another piece of the isosurface may result in disconnected components or an additional handle. We discuss this issue in more detail in Section 2.3. Our algorithm is designed to detect all handles in an iso-surface. We propose a simple metric to measure the size of these handles and a simplification method suited to the removal of *small* extraneous handles. Our method operates directly on the volume data, is out-of-core and produces topologically simplified volume data and isosurfaces.

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 connectivity 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 (for example we are not interested in maxima and minima). We therefore take an approach that is loosely related to discrete Morse theory [Forman 2002]. Similar to the work of [Axen 1999; Guskov and Wood 2001; 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 on the wavefront. Note that we do not find exact critical points on the surface, instead, we track critical regions of the surface where a discrete wavefront splits and merges. Each of these regions is well defined given our reconstruction rules, (see Section 2) and regions with non-isolated or degenerate critical points are handled appropriately. For more information see Section 2.1.

We store and track the changes of the discrete height function with the use of a *Reeb graph* [Reeb 1946]. Specifically, given a scalar function f, 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 (see Figure 4). The Reeb graph tracks how these contours split and merge as z varies and is often used to analyze surface topology, since cycles in the graph correspond to handles. 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. Because these geodesic-based methods require a breadth-first traversal of the surface, the irregular accesses to the volume make out-of-core processing difficult. Reeb graphs (sometimes called contour trees) 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. 2000; Kreveld et al. 1997], to recognize shapes [Hilaga et al. 2001], and to re-mesh surfaces [Wood et al. 2000; Attene et al. 2001].

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. This approach is simple and effective, however it has several drawbacks. Namely, their definition of topological feature size fails to detect long thin handles, since they do not fit in a small ball (see inset figure). In addition, we prefer to operate on the volume data, since an isosurface will always remain a manifold even after topological repair [Lachaud 1996].

Using the concept of alpha hulls, El-Sana and Varshney [1997] 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. One difference with our approach is that we evaluate whether to retain or simplify a handle based on a topological metric defined on the handle itself.

Edelsbrunner *et al.* [2000] 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].

Volume-based topology simplification Nooruddin and Turk [1999] 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.* [Bischoff and Kobbelt 2002]. We prefer a more targeted approach that provides analysis of the sizes of the handles and exactly 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 11 shows an example where their strategy would fail to find a short loop to break the handle).

Model simplification Several algorithms simplify topology as a byproduct of model simplification, *e.g.*, [Garland and Heckbert 1997; He et al. 1996; Popovic and Hoppe 1997]. These methods can result in non-manifold 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 [Lazarus et al. 2001; Erickson and Har-Peled 2002; Gu et al. 2002; Kartasheva 1999; Colin de Verdière and Lazarus 2002]. These approaches typically

analyze the topology of the entire surface. Our problem is slightly different as we only consider reducing the genus of the surface, not cutting it into a disk. Erickson and Har-Peled [2002] address the task of optimally cutting a surface into a disk. They propose a greedy algorithm to compute a nearly minimal *cut graph*. A cut graph is a collection of edges that cut a surface into a disk. Their approach must analyze the entire surface, compute a cut-graph for the surface and then find nearly-shortest essential loops one at a time. Our search for non-separating cuts is localized to process one handle at a time. This means that we are not guaranteed to find the globally minimum cuts. However, our algorithm can operate out-of-core and generate fast approximations of short non-separating cuts used to simplify the topology,

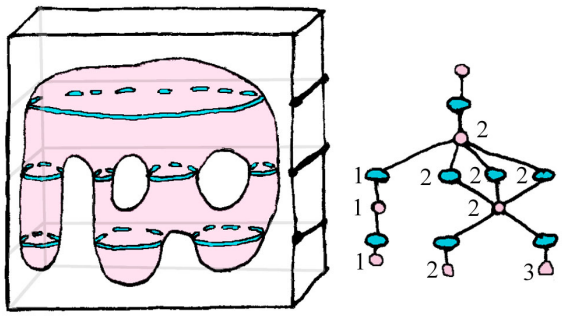


Fig. 4. An isosurface and its corresponding Reeb graph for a bottom to top sweep of the volume. In the graph, contour nodes are shown in blue, and ribbon nodes in pink. Also shown on the graph are component labels, here represented as numbers.

2. OUR APPROACH

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] or [Lorensen and Cline 1987]) 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. For our algorithm, the important element is the connectivity of the polygons, as this connectivity defines the topology of the surface. Our algorithm never requires the construction or storage of a triangulation of the surface. We assume that the connectivity of the polygons is pre-determined, 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 nor self-intersections [Lachaud 1996]. Lachaud's table has proven properties by restricting data to have well defined interior and exteriors, *i.e.*, for a scalar function $F(x)$, the interior is defined as $F(x) < 0$, while exterior is defined as $F(x) \geq 0$. This is similar to a standard general position argument, and creates a well defined isosurface, *i.e.*, the surface is perturbed away from the volume grid nodes.

Our approach uses a height function to construct a Reeb graph representing the topology of the surface. An axis-aligned *sweep* through the volume 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. Given the reconstruction rules of Lachaud, the isosurface is well defined and likewise all the ribbons and contours are well defined.

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). Our problem of topology simplification corresponds to reducing surface genus, *i.e.*, removing 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. However, for completeness we simplify the topology of all the components in the volume.

Our goal is to locate 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 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 remove all handles whose measured 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 minimum-length *non-separating cut*. A non-separating cut leaves the surface connected [Aleksandrov 1956]. For example, a sphere has no non-separating cuts. In a torus, such a cut will be one of a pair of the many generator loops that form a basis of the handle [Munkres 2000]. In order to support out-of-core processing of the data, our algorithm does not find globally minimal-length non-separating cuts. Instead our method finds locally short non-separating cycles for each handle. See Figure 11 for an illustration of such cycles. Cutting the surface along such a cycle and then pinching each new boundary of the cut to a point removes the handle.

Approach overview Our approach can be summarized as:

- Sweep through the volume to *locate* all 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.

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 (V), edges (E), and faces (F) for each individual component of the surface that would be generated during isosurface mesh extraction [Lachaud 1996]. 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.

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 or z-interval 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 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 and 8. Cycles in the Reeb graph correspond to handles on the surface.

Note that for constructing the Reeb graph of a surface, one must choose the rate to sample the height function. In our discrete setting, the ideal sampling rate is the largest step that can be taken while still capturing all the topology of the surface. In some settings there is no choice. For example, in work done by Shinagawa [1991], a Reeb graph is constructed from predetermined cross sectional contours only. 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 by the connectivity information of the ribbon between z intervals.

We choose to sample the height function at the 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. However, the consequence of this sampling choice is that when a handle is entirely contained within a ribbon, (*i.e.*, within a slice of the volume), it does not initially appear as a cycle in our Reeb graph. One such *intra-ribbon handle* is generated by the 6×6 cube grid shown in Figure 5. In practice, these intra-ribbon handles occur for 1-10% of the total slices for a volume. We detect these intra-ribbon handles by computing the Euler characteristic of each surface ribbon. If a ribbon has non-zero genus, it obviously contains handles. The solution for such cases is to locally modify the height function to be a geodesic function defined on the vertices within the slice. Constructing a Reeb graph from contours defined by an ordering on the vertices within the slice will guarantee that we encode the intra-ribbon handles in our Reeb graph. In addition, confining the per vertex geodesic traversal to the slice keeps our surface access local, allowing our method to remain out-of-core. We discuss our method in more detail later in Section 2.1.

Finding cycles in the Reeb graph Reeb cycles 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. To detect a handle, the algorithm needs to locally differentiate in the Reeb graph between a lone saddle point and a pair of saddle points that form a handle. 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. Such a labeling allows us to locally differentiate 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). 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

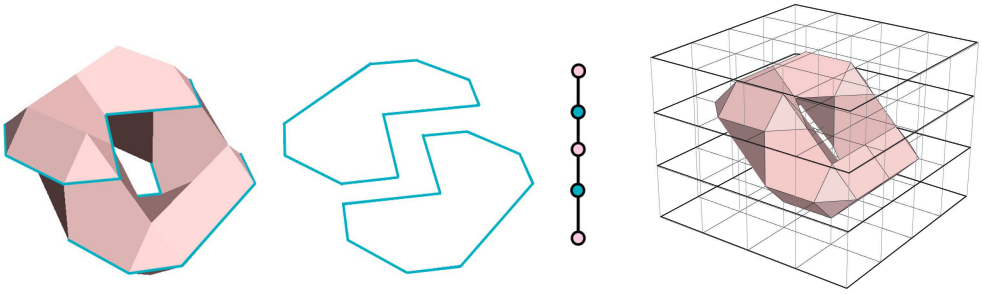


Fig. 5. Example of intra-ribbon handle. This torus tilted at an angle is formed by two "C" shaped contours. As shown in the figure on the left, the Reeb graph does not contain any cycle. On the right we see the volume grid overlaid on this isosurface. Observe that the slice between the two "C" shaped contours is equivalent to a torus with boundaries.

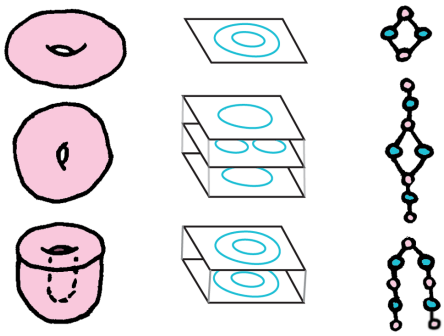


Fig. 6. Example surfaces and their associated contours and Reeb graphs. The examples are: a torus on its side, an upright torus, and a bowl-like surface.

the two contour nodes have the same component label. If so, they belong to the same connected 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. The process of isolating the geometric extent of a handle is a combinatorial problem as a surface can be 'cut' into isolated handles in many different ways. For the purpose of computer graphics, it is reasonable to isolate handles that are geometrically localized in the surface. Each of the handles that we consider, correspond to a cycle in our Reeb graph. Each of these cycles correspond to the height function splitting into two components and then merging back together. As the height function is spatially localized, the cycles will correspond to geometrically succinct regions. 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, *e.g.*, $c1$ 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

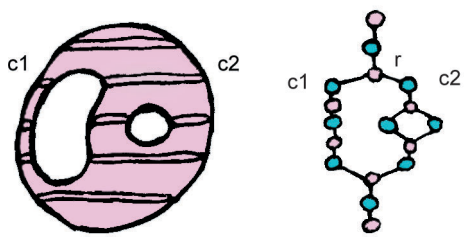


Fig. 7. Example surface and its Reeb graph with adjacent handles. The ribbon r has the previous contours $C = \{c1, c2\}$. When we discover the Reeb cycle associated with contours $c1$ and $c2$, we construct the cycle path by finding the shortest path *from* $c1$ *to* $c2$.

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 non-separating 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]).

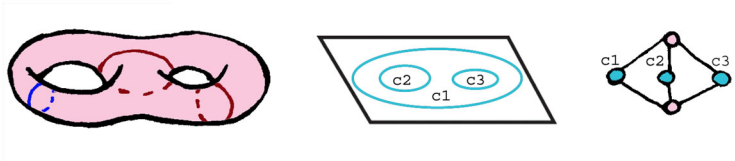


Fig. 8. A two-holed torus, the associated planar cross section, and its associated Reeb graph. To find the locally shortest non-separating cut (shown in blue), we must explore all pairs of child contours with the same component label (*i.e.*, $\{c1, c2\}$, $\{c1, c3\}$ and $\{c2, c3\}$ where the minimal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

The following pseudocode 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.

Finding intra-ribbon handles Recall that we must also consider the case of a handle contained entirely within a ribbon. For each slice of the volume, we keep track of the

Euler characteristic of the surface and if the Euler characteristic does not match the number of cycles in the Reeb graph, we know that the previous slice must contain at least one intra-ribbon handle. To detect the handle(s) we modify our height function to locally be a discrete geodesic function defined on the faces in the current slice. Specifically, we use a breadth-first traversal over the faces in the current slice. We construct a contour after adding each neighboring face one at a time. This face by face traversal is limited to only the polygons in the current slice. Such a construction guarantees that our Reeb graph will correctly encode a cycle for each intra-ribbon handle. Note that an intra-ribbon handle relates to a non-isolated critical point. Altering our algorithm to construct a contour after adding each face within the slice, resolves this issue by isolating distinct critical regions, (*i.e.*, where the contours split from one component into two and vice versa). We could traverse the entire volume with a discrete geodesic, however, the breadth-first traversal of the surface would cause irregular access to the volume data, making out-of-core computation impossible. Once a Reeb cycle is found for an intra-ribbon handle, the previously described approach to isolate handles is applied.

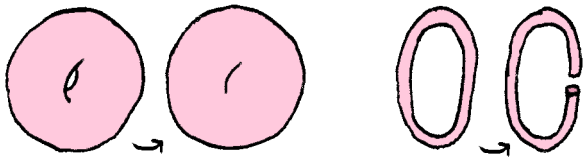


Fig. 9. Two ways of removing a handle, illustrated on two tori. The "fat" torus is best repaired by collapsing the handle, and the "skinny" torus is best repaired by pinching the handle

.

2.2 Measuring Topological Handle Size

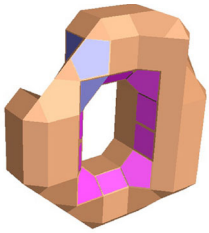
Recall that a Reeb cycle in the Reeb graph identifies a cycle of ribbons forming a handle. There are two natural ways to remove a handle (Figure 9). In order to remove either of the handles shown in Figure 9, we reduce the number of non-separating cycles for the surface. Recall that genus is defined as the maximum number of simple non-separating cuts that do not intersect. By collapsing a non-separating cut we conceptually either fill in the interior of a handle or we pinch open the handle (Figure 9). 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 non-separating 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 actual implementation of this operation on our discrete grid volume is discussed in the next section.

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

- the *Reeb loop* is the locally short loop around the Reeb cycle, and
- the *cross loop* is the locally short loop "transversal" to the Reeb loop.

For a given sweep direction, a loop that may be called a Reeb loop would alternately be

called a cross loop from an orthogonal sweep direction. On the irregularly shaped torus, shown on the right, the Reeb loop is shown in magenta, and the cross loop is shown in blue. Since we perform a sweep in only a single direction, it is important that we consider both types of handle removal operations. There are many possible non-separating 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 define handle size to be the smaller of the Reeb loop length and the cross loop length.



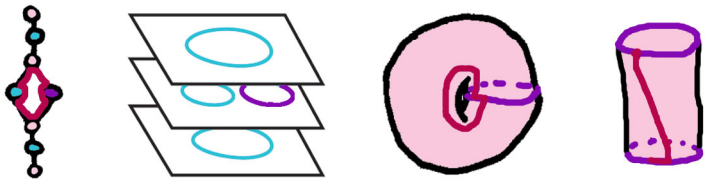


Fig. 10. Illustration of the process of identifying the Reeb loop for a torus. In this case, we are searching for a loop on the surface of the torus that corresponds to the red cycle in the Reeb graph shown on the left. The search is started from one of the contours in the Reeb-cycle, shown in purple in the middle two images.

We find the **Reeb loop** by constructing a non-separating 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 non-separating curves, which 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 10. We construct a locally short non-separating cut that follows the Reeb cycle by computing the shortest path from one side of such a contour to the other. In the 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. The Reeb cycle contains at least one pair of contours in the same plane, thus we start our search from one such contour and only consider returning paths that have passed through the other contour.

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 non-separating cuts, we can now derive a measure of the handle. Generally, we use the smaller of the two cuts 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. 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

Model	Grid size	#Faces	Thresh. size ℓ	Genus before	Genus after	#Intra-ribbon	Loops collapsed #Reeb	Loops collapsed #cross	Timing (minutes)
David	$885\times736\times709$	15,244,302	166.5	1063	0	76	332	731	87.5
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
Brain1	$125\times255\times255$	688,248	32.5	366	0	6	320	46	2.8
Brain2	$125\times255\times255$	452,050	14.5	21	0	6	12	9	0.7
Brain3	$125\times255\times255$	529,012	10.5	41	0	4	25	15	0.6
Brain4	$125\times255\times255$	699,566	14.5	50	0	7	11	39	1.7
Feline	$332\times148\times316$	653,922	4.5	6	2	1	2	2	0.2

Table I. 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 handles collapsed by Reeb or cross loop. Times are shown in CPU minutes for a 1 Ghz, Pentium 4. All values listed are for the entire volume, *i.e.*, for the surface and any spurious disconnected components in the volume data.

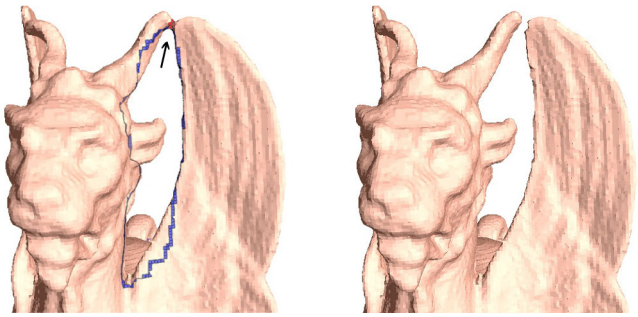


Fig. 11. Close-up of the feline mesh with the Reeb loop shown in blue, and cross loop shown in red. The right image shows our algorithm's output after collapsing the cross loop of the handle.

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 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, *e.g.*, if filling a long narrow opening in a wide surface is a desired result. Note that by evaluating both loops, this measure is independent of sweep direction.

2.3 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 [Kaufman 1987]. 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 col-

lapses and pinches off handles through insertion of a thin wall. The modified isosurface, once extracted from the volume data, is guaranteed to remain a manifold and to have no self-intersections [Lachaud 1996].

There are a few potential problems to consider. Our algorithm provides the essential information to always successfully remove a handle, *i.e.*, the location of a locally short non-separating cut for each handle. Topologically, the handle can always be closed along this loop, reducing the genus of the model. However, depending upon how the surface is embedded in $\mathbb{R}^3$ 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 has never occurred since we only simplify *small* handles. 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 halting problems if each collapse always created a new handle. We have chosen a relatively simple method for performing loop closure due to the fact that our target application is to remove small excess topology from models. In practice, the issue of obstructions has never caused the creation of new handles or halting problems for our approach when simplifying small extraneous topology. In addition, even for large loops that do not contain obstructions, our simple closure routine performs as expected (see inset Buddha figure). However, for an alternative application that targeted closing large loops it would be important to add a criterion to check for obstructions and alter the closing routine accordingly.

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 models are from an MRI scan from the Harvard Medical School [Kikinis et al. 1996].

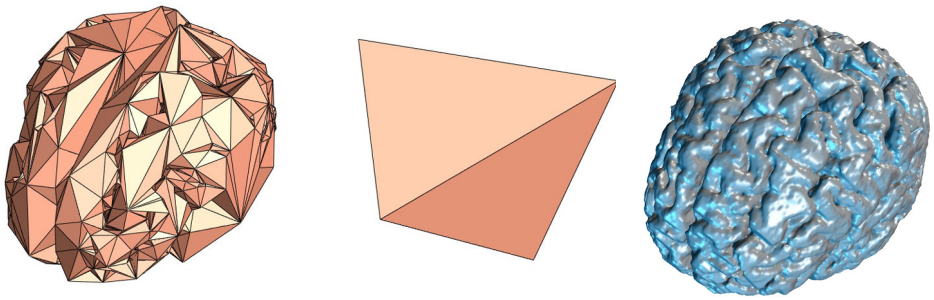
We have demonstrated the robustness of our algorithm using convoluted geometry (Figure 12) and large volumes (Table I). Since our algorithm locally reconstructs the Reeb graph after every topological change, it guarantees that we are accurately identifying all of the topology of the surface, even as its topology evolves. In practice our method has always 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.

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

The scatterplot in Figure 18 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 17). 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 components has a handle of length 46.



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

Fig. 12. Comparison of the base meshes of progressive meshes on a brain model (MRI).

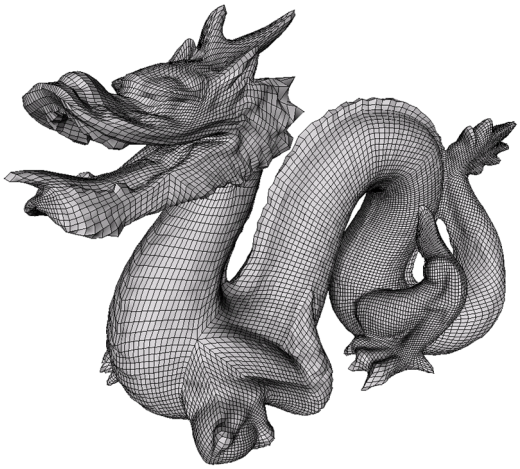
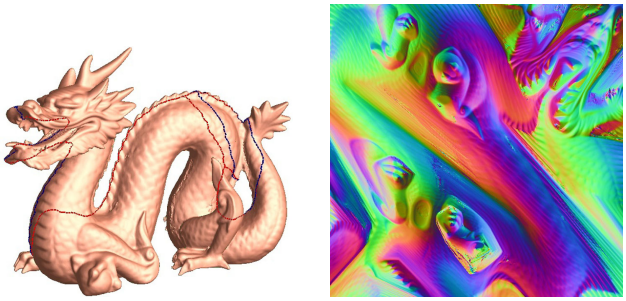


Fig. 13. A remesh of the genus 1 dragon. Given the difficulty of achieving a high-quality parametrization for high-genus models, remeshing the original dragon with genus 46 would be quite challenging and require numerous elements in the base domain.

3.1 Applications

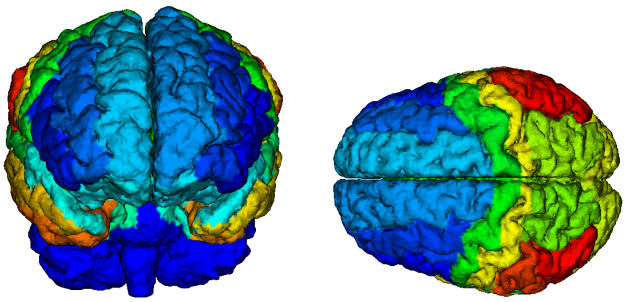
Topology simplification facilitates many surface operations:

- Fewer triangles are wasted to encode topological defects during *mesh simplification*, as shown in Figures 2, 3 and 12 using the progressive mesh representation of Hoppe [1996]. Consequently, coarser meshes can be created, and geometrical quality is improved at all levels of detail.



Cuts on the dragon model Complete normal-map image of the dragon

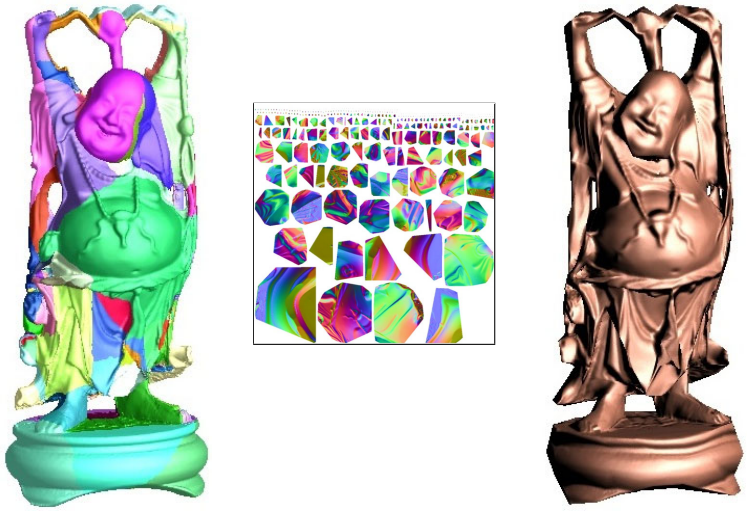
Fig. 14. Geometry Image of the genus 1 dragon model, showing the cuts used to parametrize the entire model onto a single chart. Parameterizing the 500K face dragon onto a unit square would cause large distortion with the original genus 43 model.



Labeled cortex (genus 0)

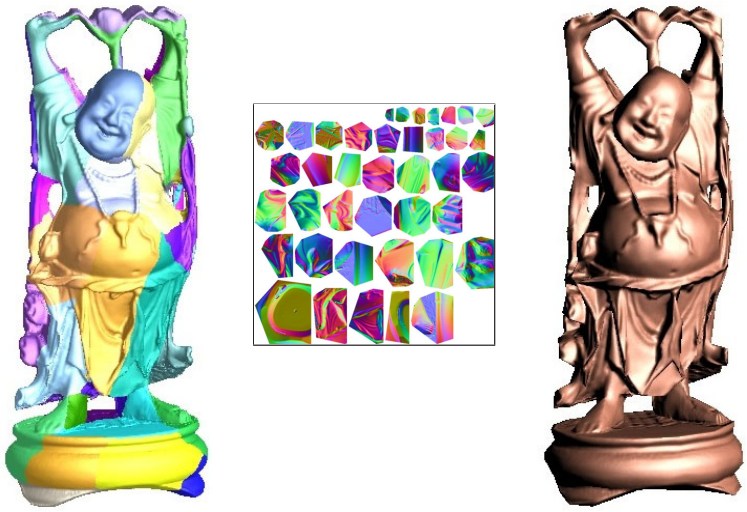
Fig. 15. Two different views of a brain model in which cortex labels have been propagated from one brain to the next through the method of Jaume *et al.*.

- Better surface parametrization improves *texture mapping*, as shown in Figure 16 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 13 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 clean 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 14, 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 such a cortex labeling benefit from operating on topologically clean volumes. For example, the approach of Jaume *et al.* [2002], requires genus zero brain models and volumes to propagate cortex labels correctly. See Figure 12 for an illustration of excess topology in brain data. Using our method to obtain topologically clean volumes, Jaume *et al.* are able to propagate cortex labels from one labeled volume to others. See Figure 15.



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)

Fig. 16. 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.

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 (Figure 18) or histogram (Figures 17. 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.

We observed that the initial statistics can change significantly as handles are simplified. Figure 18 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 18), where a data point near the ℓ line moves to the top right once the small handle is removed. Note that this effect would be present in any topology simplification that sequentially treats handles locally one by one, when handles are adjacent to one another as seen in Figure 18. As adjacent handles are simplified, the topology and measure of that topology changes. There is more than one way to simplify the topology of a shape. In order to support out-of-core operations on large data we have chosen to analyze topology locally in terms of handles. Since our intended application is the removal of topological artifacts, we have chosen an approach that automatically removes excess topology of a given size. An alternative application could integrate visualization of the handles and their non-separating cuts to allow the user more fine control over the order in which loops are collapsed if so desired. For our setting, we found that our method performed well and achieved the desired results.

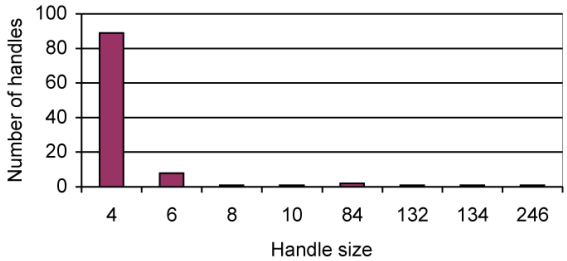


Fig. 17. Histogram of handle sizes for the original scanned Buddha model. Recall that handle size is the smaller of the Reeb and cross loop lengths.

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, there is processing time 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 $O(d \log d)$, where d is the number of polygons in the local handle. This measure correspond to running Dijkstra's algorithm. In the worst case situation the complexity for a given handle could be $O(n^2)$, if the entire surface were composed of one large handle.

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

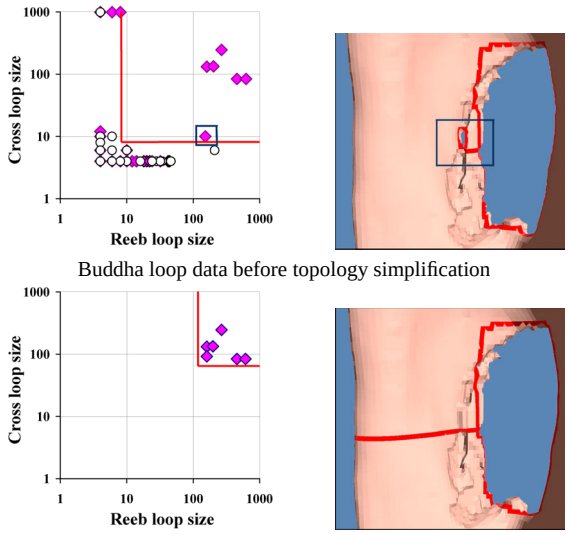


Fig. 18. Scatterplot of the Reeb loop and cross loop lengths of the handles of the Buddha, before and after topology simplification. Hollow circles identify handles whose minimal loop encloses a void. The red lines mark the range of ℓ that keeps exactly these 6 handles. On the right we see corresponding 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.

practice, we only keep 50 slices of volume in memory at any time, making the algorithm viable even for low-end computers. The choice of buffer size is flexible and can change due to the size of the volume and the memory resources available. 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, $P < O(n^2)$. Typically we store $\approx 50 \times n$ polygons, since polygons below the bottom of the current buffer are erased to minimize memory use. These local polygons are stored in order to compute loops for any handle that is being processed, however, they can also quickly be recomputed using a table look-up [Lachaud 1996]. In general computing the Reeb loop for a handle 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. The worst case situation is that of a very long handle with a thin cross-section somewhere along its length. In this worst case setting, the memory requirements may reach $O(n^2)$ to store all of the polygons, in practice this does not happen.

Since polygons below the *bottom* of the current buffer are erased to minimize memory use, Reeb and cross loop computation for handles with length > 50 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.

4. SUMMARY AND FUTURE WORK

We have introduced a 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. This mesh setting poses challenges for out-of-core approaches.

Another 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 some settings and applications could require the removal of topological features of a more substantial size. Since we have information about the local region affected by the loop closure, we could smooth the newly inserted surface. In range data reconstruction algorithms, it is already common to smooth the unscanned, filled-in regions of the surface [Curless and Levoy 1996; Davis et al. 2002].

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 pre-process the volume to remove topological artifacts for a range of isosurface values. For such a setting, the work of Zomorodian [2001] is promising.

Acknowledgments This work was supported in part by the NSF (DMS-9874082, ACI-9721349, DMS-9872890, ACI-9982273 CCR-0133983), 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; 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 and Afra Zomorodian 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*. IEEE Computer Society, Genova, Italy, 142–151.

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

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. 2000. Computing contour trees in all dimensions. In *Symposium on Discrete Algorithms*. 918–926.

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, Canada, 627–636.

CORMEN, T., LEISERSON, C., AND RIVEST, R. 1990. *Introduction to algorithms*. MIT Press.

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

DAVIS, J., MARSCHNER, S. R., GARR, M., AND LEVOY, M. 2002. Filling holes in complex surfaces using volumetric diffusion. In *First International Symposium on 3D Data Processing, Visualization, and Transmission*.

EDELSBRUNNER, H., HARER, J., NATARAJAN, V., AND PASCUCCI, V. 2003. Morse complexes for piecewise linear 3-manifolds. In *Proc. 19th Ann. ACM Sympos. Comput. Geom.* 98–101.

EDELSBRUNNER, H., LETSCHER, D., AND ZOMORODIAN, A. 2000. Topological persistence and simplification. *Proc. 41st IEEE Sympos. Found. Comput. Sci.*, 454–463.

EL-SANA, J. AND VARSHNEY, A. 1997. Controlled simplification of genus for polygonal models. *Proceedings of IEEE Visualization*, 403–412.

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

FORMAN, R. 2002. A user's guide to discrete morse theory. In *Seminaire Lotharingien de Combinatoire*. 48.

FRANCIS, G. AND WEEKS, J. 1999. Conway's zip proof. *Amer. Math. Monthly*.

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

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

GUSKOV, I., KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2002. Hybrid meshes. In *ACM SOCG*.

GUSKOV, I. AND WOOD, Z. 2001. Topological noise removal. *Graphics Interface*, 19–26.

HE, T., HONG, L., VARSHNEY, A., AND WANG, S. W. 1996. Controlled Topology Simplification. *IEEE Transactions on Visualization and Computer Graphics 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. *Proceedings of SIGGRAPH*, 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. *Proceedings of SIGGRAPH*, 99–108.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2, 71–78.

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

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

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

KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2000. Progressive geometry compression. *Proceedings of SIGGRAPH*, 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 Transactions on Visualization and Computer Graphics*.

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

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

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

LAZARUS, F. AND VERROUST, A. 1999. Level set diagrams of polyhedral objects. In *ACM Solid Modeling'99*. Ann Arbor, Michigan, USA.

LEVOY, M. AND OTHERS. 2000. The digital michelangelo project: 3d scanning of large statues. *Proceedings of SIGGRAPH*, 131–144.

LORENSEN, W. E. AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of SIGGRAPH 21*, 4, 163–169.

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

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

MUNKRES, J. 2000. *Topology*. Prentice Hall.

NOORUDDIN, F. AND TURK, G. 1999. Simplification and repair of polygonal models using volumetric techniques. Research Report 99-37, Georgia Tech.

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

REEB, G. 1946. Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction munerique. *Comptes Rendus Acad. Sci. de Paris*, 847–849.

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

SCHRÖDER, P. AND SWELDENS, W., Eds. 2001. *Digital Geometry Processing*. Course Notes. ACM Siggraph.

SHATTUCK, D. W. AND LEAHY, R. M. 2001. Automated graph based analysis and correction of cortical volume topology. *IEEE Transaction on Medical Imaging*.

SHINAGAWA, Y. AND KUNII, T. L. 1991. Constructing a reeb graph automatically from cross sections. *IEEE Computer Graphics and Applications 11*, 6, 44–51.

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

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