Extraction of Topologically Simple Isosurfaces from Volume Datasets

Andrzej Szymczak and James Vanderhyde

Georgia Tech

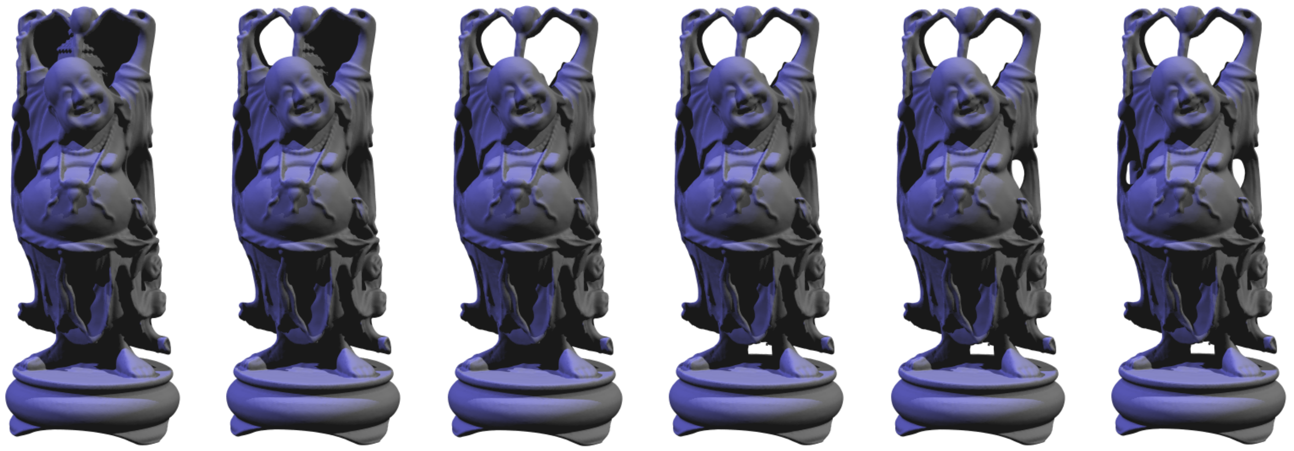

Figure 1: Buddhas of increasing genus. First, the two handles formed by the arms are created, then one on the bottom and, finally, two on the sides.

Abstract

There are numerous algorithms in graphics and visualization whose performance is known to decay as the topological complexity of the input increases. On the other hand, the standard pipeline for 3D geometry acquisition often produces 3D models that are topologically more complex than their real forms. We present a simple and efficient algorithm that allows us to simplify the topology of an isosurface by altering the values of some number of voxels. Its utility and performance are demonstrated on several examples, including signed distance functions from polygonal models and CT scans.

I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling

Keywords: isosurface, topology, genus

1 Introduction

The performance of a number of algorithms in graphics and visualization is known to decay with the increase of the topological complexity of the input. Examples include work on resampling and parametrization [25, 16, 19], geometry compression [35, 17, 2, 27], texturing [32] and many others. On the other hand, the standard 3D geometry reconstruction algorithms often produce 3D models that are topologically much more complex than their real forms. This is due to both inaccuracy and noise generated by the present 3D scanning devices and the insensitiveness of the reconstruction algorithms to the topological structure of the reconstructed object. This paper introduces a simple and efficient technique allowing to extract topologically simple approximate isosurfaces from volume datasets. It can either work directly with 3D acquisition methods in which the data is captured in a volumetric form (like CT or MRI) or when a volumetric model is one of the intermediate stages [26]. It can also be used as a component in a volumetric topology simplification procedure for polygonal models in a manner similar to [30].

2 Overview of the Algorithm

Our algorithm takes as input a volume dataset with an isovalue and a nonnegative integer T specifying the desired genus of the output isosurface. Without any loss of generality, we can assume that the isovalue is equal to zero, the voxels inside the volume bounded by the isosurface are negative and the voxels outside that volume are positive. We will also assume that the isosurface does not intersect the boundary of the volume, i.e. that the boundary voxels have a positive value. This assumption can easily be enforced by adding a 'border' of width 1 consisting of voxels of a positive value around the original volume.

Volume datasets that appear in practice frequently contain considerable amount of noise. In particular, it is common that the isosurface in the input volume consists of several connected components, some of them small and insignificant. In many cases, one is interested only in the largest connected component. Therefore, below we will concentrate on a variant of our procedure that produces the outer connected component of the part of isosurface contained in the largest connected component of the union of all negative voxels. An alternative variant of the algorithm (treating the sum of the connected component count and the geni of all connected components of the isosurface as a measure of topological complexity and providing the user with control over this quantity for the output surface) is briefly discssed in Section 7. We will proceed in five stages:

1. **Cleanup.** The largest connected component of the union of negative voxels is computed. The values of all of the negative voxels that do not belong to it are changed to an arbitrary pos-

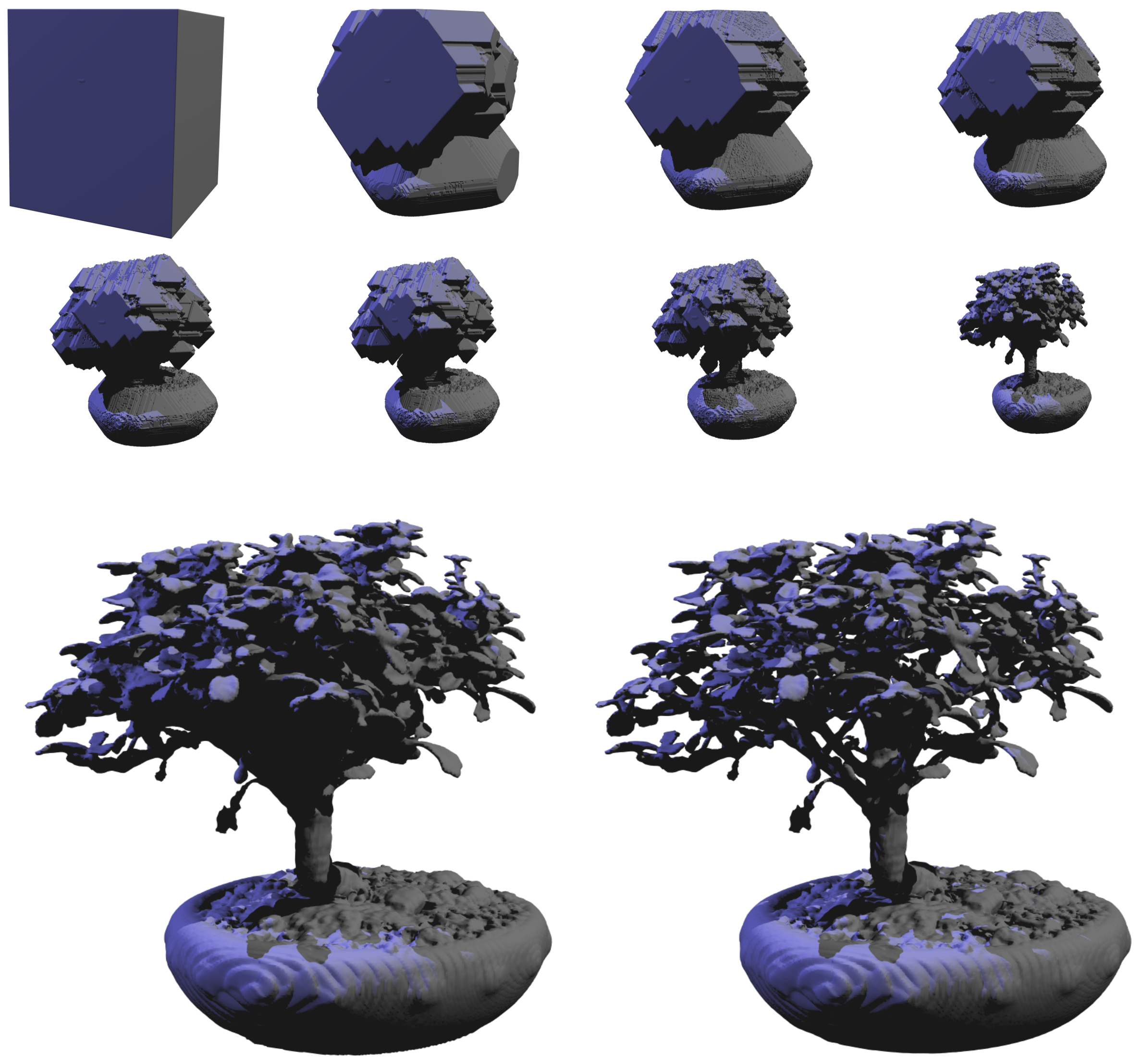

Figure 2: Top: snapshots of the carving process (with just one resolution level) for the bonsai dataset. Bottom: the isosurface extracted using our algorithm (left, genus 0, obtained using no topology-altering voxel removals) and the largest connected component of the original isosurface (right, genus 463).

itive number. After this operation, the union of all negative voxels is connected.

2. **Distance Function Approximation.** The values of the voxels outside the isosurface are altered so that they are close to the distance to the isosurface. This is done in a way preserving the geometry of the isosurface, i.e. so that the isosurfaces in the original and altered volumes are the same.

3. **Topology-Sensitive Carving.** An operation that removes a boundary voxel with a positive value while preserving the topology (more precisely: homotopy type) is performed iteratively, starting from a bounding box of the isosurface that is a union of voxels. If no topology-preserving boundary voxel removal operation is possible, a topology-altering one is executed. The total number of topology-altering operations can be at most T. Thus, the procedure terminates either when all positive voxels have been removed or when no topology-preserving operation is possible and T topology-altering operations have already been executed. In Section 6 we will see that the number of topology-altering voxel removal operations practically gives the user precise control over the genus of the output isosurface. To make the boundary of the carved set close to the original isosurface (in particular, open large handles before small ones) we apply the greedy heuristic: the voxels are removed in the order of decreasing value (i.e. approximate distance to the isosurface). Snapshots of the carving process are shown in Figure 2. Figure 1 shows isosurfaces obtained from the signed distance field of the Buddha model for $T = 0, 1, 2, 3, 4$ and 5.

4. **Volume Update.** The positive voxels in the set constructed in the previous step are made negative.

5. **Isosurfacing.** A variant of the Marching Cubes algorithm that produces an isosurface bounding a volume of the same homotopy type as the union of negative voxels [4, 20] is applied.

Our procedure can be made considerably more efficient by adopting a multiresolution approach. After the distance function approximation stage, a pyramid representation of the volume (see

[5]) is built based on the ‘minimum’ filter, which computes the minimum of voxel values in a $2 \times 2 \times 2$ block. The carving stage (without topology-altering voxel removals) of our algorithm is first performed on the lowest resolution. Then, the resolution is increased by subdividing the voxels and the carving algorithm is restarted. This process is repeated until the original resolution is reached. All the remaining stages of the algorithm, including the topology-altering voxel removal operations, are run on the original resolution.

3 Related Work

Our algorithm is related to a number of results concerning topology simplification, level set methods, reconstruction from point clouds and the issue of computing and representing topological features in 3D data.

3.1 Topology Simplification

The problem of simplifying the topology of 3D surfaces has been studied by a number of researchers. A three-dimensional hole-filling algorithm very similar in spirit to ours is discussed in [1]. However, it is limited to binary volumes, while our approach offers sub-voxel accuracy of the output surface. We also provide a simple way of specifying the desired genus of the output and an efficient multiresolution implementation.

The algorithm of [37] aims to achieve the same goal as ours: to simplify the topology of an isosurface by introducing small and local changes to the volume. However, in contrast to our approach, it explicitly analyzes the topology of the input isosurface based on its Reeb Graph. For each small handle, an associated non-separating loop in the isosurface is found and the values of voxels located along its spanning disk are changed so that the handle disappears. Since we do not explicitly find and annihilate non-separating loops in the isosurface, our algorithm is more efficient for highly complex datasets. For example, the bonsai dataset of resolution $256 \times 256 \times 256$ shown in Figure 2 took 73 seconds to process on an 850MHz P3 machine (together with the isosurface extraction phase), while [37] reports the running time of 2.8 minutes for a brain dataset of resolution $125 \times 255 \times 255$ and with an isosurface of a comparable topological complexity (on a dual P4 machine). Our algorithm is also significantly simpler.

The work [18] discusses a procedure for removing topological noise (i.e. small handles) from triangulated 3D surfaces based on a principle similar to [37]. First, small loops on the input surface that are not contractible are found. Then, the surface is cut along these loops and the resulting boundary loops are filled with triangulated topological disks.

An algorithm that simplifies the topology as well as the geometry of a surface by rolling a sphere over it and filling up the regions that are not accessible to it is proposed in [12]. The method is reported to work very well with CAD models. Our approach is different in that it performs solely topological simplification, attempting to minimize the changes to geometry. In particular, our method would not fill cavities that are merely geometric features of the isosurface.

A way to incorporate both preservation and controlled simplification of topology of isosurfaces into a geometric volume simplification procedure is described in [13]. The multiresolution variant of the algorithm discussed in this paper can be viewed as a superposition of simplification (building a pyramid representation) and a topology-preserving refinement operations (supersampling and carving). However, our goal is to simplify the topology of an isosurface while attempting to preserve its geometry as faithfully as possible. It is not clear how the algorithm of [13] can be used for this task in an equally straighforward way.

3.2 Level Set Methods

A method to ensure preservation of topology in deformable models is described in a series of papers [20, 21]. It can also be used to extract topologically simple isosurfaces, but its reliance on deformable models makes it significantly harder to code and less efficient.

The level-set–based surface reconstruction algorithm with hole filling capability described in [39] also bears some similarities to our approach. It presents a method of reconstructing a surface from an arbitrary 3D set. It uses level set method to find a surface that is minimal with respect to an objective function defined using the distance from the input set (essentially, the L^p norm of the sidtance restricted to the surface). The carving stage of our algorithm can be thought of as a greedy approach to a similar optimization problem: to supplement a voxel set with more voxels that are as close to it as possible in a way that reduces its topological complexity.

3.3 Reconstruction from Point Clouds

The papers [10, 14, 15] restate the problem of reconstructing 3D models from unorganized points using the terminology of dynamical systems, which also applies to our approach. Setting smoothness and discretization issues aside, the carving stage of our algorith is similar to computing the global attractor of the gradient vector field defined by the distance function from the isosurface. The topology-altering operations are, in turn, analogous to the reduction operations by pairing and cancellation as described in [14].

3.4 Computing and Representing Topology

The papers [31, 6, 34, 36] present efficient algorithms for analyzing the topological structure of the level sets. In [11], the authors consider the topology simplification problem in the context of alpha shapes. We feel that the techniques introduced in these papers can help provide guaranteed control over the topology of the output surface of our scheme as well as more precisely describe its topological properties.

4 The Basic Algorithm

In this section we describe the five basic steps of our algorithm, leaving the discussion of the multiresolution implementation for the next section.

4.1 Cleanup

In many cases, especially for common CT scans, the union of the negative voxels in the input volume contains small connected components due to noise. Such components might also appear as sampling artifacts in distance fields. The purpose of this stage is to remove these connected components.

First, we label all the connected components and compute their sizes (in our case, the numbers of voxels, but other size measures can also be used) using the depth-first search algorithm [8]. Then, all values of the negative voxels outside of the largest connected component are altered to arbitrary positive values.

4.2 Distance Function Approximation

The distance function approximation step influences the order in which the voxels are removed in the carving phase, making it close to the ordering with respect to decreasing distance from the isosurface. This allows us to avoid the topological locks discussed in Section 7 and keeps the distance between the exact isosurface and its approximation produced by our algorithm small.

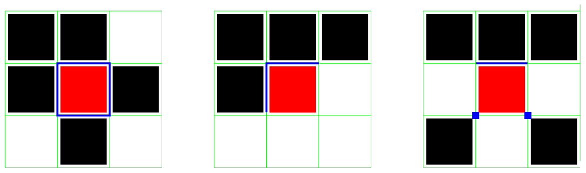

Figure 3: Does the removal of the red voxel change the topology (of the set A formed by red and black voxels)? The set $I(A,V)$ is shown in blue. In the case shown on the left, the set $\delta V \setminus I(A,V)$ is empty (has zero connected components). Therefore, the removal of V changes the topology. In the case shown in the middle, both $I(A,V)$ and $\delta V \setminus I(A,V)$ have one connected component and therefore there is no topology change. Finally, for the configuration shown on the right, the set $I(A,V)$ has three connected components. Therefore, the removal of V induces a topology change.

The values of the positive samples away from the isosurface are altered so that they increase together with the distance from the isosurface. In order to preserve the original geometry of the isosurface, we do not change values of the positive samples that are adjacent to a negative sample. We first compute the maximum value M of a positive sample connected to a negative sample by a grid edge. Then, we compute the distance $d(V)$ of each sample V from the set of negative samples. Our implementation uses the Manhattan distance rather than the Euclidean distance for efficiency (clearly, fast marching-based distance algorithm of [33] can be used instead). Finally, the value of each positive sample that is not connected by a grid edge to a negative sample is changed to $M + d(V)$.

4.3 Carving

We start from a bounding box A of the output isosurface that is a union of voxels. The order in which the voxels are carved out of A is governed by a priority queue, initialized to hold all positive voxels on the boundary of A and ordered by the voxel values. The following loop (L) forms the basis of our procedure.

1. Extract a voxel from the priority queue

2. If the removal of that voxel (call it V) would not change the topology:

- **2a.** Remove V from A
- **2b.** Insert all positive voxels that belong to A and are neighbors of V into the priority queue, unless they are already queued

3. If the queue is not empty, go to 1.

The test for topology change in step 2 is based on counting connected components of the intersection of the boundary of V (denoted by δV) and the set A with voxel V removed (i.e. the union of all voxels in A except for V). If both that intersection (which we denote by $I(V;A)$) and its complement in δV, i.e. $\delta V \setminus I(V;A)$, each have exactly one connected component, the voxel removal does not change the topology (for an illustration in the 2D case, see Figure 3). Otherwise, it does. Clearly, this test can be completed in constant time. In practice, performing it offline for all possible configurations of a voxel's neighbors and storing the results in a lookup table leads to considerable savings of computational time. Since each voxel has 26 neighbors and each of them can be in two possible states (in A or not), there are 2^{26} configurations. Thus, the lookup table requires 2^{26} bits or 8MB of memory. Clearly, it is possible to reduce the number of configurations and therefore also the size of the lookup table by exploiting symmetry, but it does not appear to be worthwhile in practice.

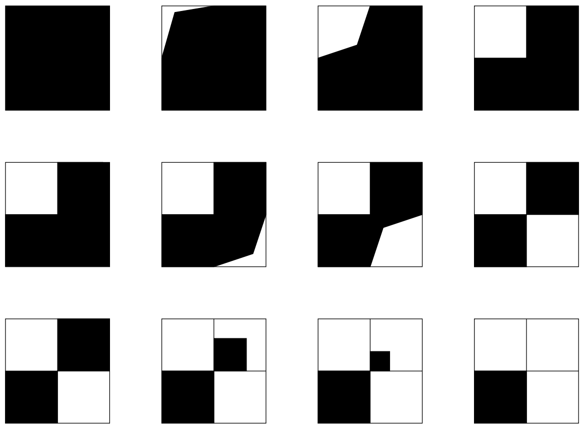

Figure 4: Three topology-preserving voxel removals from a 2D two by two square. Each row shows a deformation of the set prior the removal onto the set resulting from the removal.

Our topology preservation test for a voxel removal operation is equivalent to verifying the *contractibility* of $I(V;A)$. A topological space X is contractible if and only if it can be continuously deformed to a point within itself, i.e. if there exists a continuous family of continuous mappings $F_t : X \to X$ with $t \in [0,1]$ such that F_0 is the identity and F_1 is a constant map. Another topological concept closely related to this procedure is that of *strong deformation retract*. A topological space $Y \subset X$ is a strong deformation retract of X if X can be deformed (collapsed) continuously into Y without moving the points of Y, i.e. if there exist a continuous family of maps (called a deformation) $H_t : X \to X$ such that each H_t restricted to Y is the identity, H_0 is the identity on X and the image of H_1 is equal to Y. In fact, the set generated by the loop (L) described at the beginning of this section is a strong deformation retract of the bounding box that it started with. The loop (L) can be viewed as a construction of the deformation of one onto the other (Figure 4). Since the bounding box is contractible and any strong deformation retract of as contractible set is known to be contractible [23], the final set is also contractible. In particular, it has no voids or handles.

The procedure as described thus far always produces a contractible voxel set containing all the negative voxels. In order to give the user control over the topology of the output, we allow a user-prescribed number of voxel removals that *do* change the topology. A topology-altering voxel removal can be executed whenever the priority queue becomes empty and A still contains positive voxels. The removed voxel is chosen as the boundary voxel in A of the largest value or, equivalently, the one in A that has a positive value and failed the topology preservation test (line 2) at the earliest time during the carving phase. In particular, this choice causes large handles of the set represented by the negative voxels to be opened before the small ones (see Figure 1). Our implementation simply keeps all voxels that failed the test in a FIFO queue and searches it for the first voxel that has not been removed yet. After the topology-altering voxel removal is performed, all neighbors of the removed voxel that belong to A are inserted into the priority queue and the loop (L) is restarted. The whole process is iterated until there are no positive voxels in A or until T topology-altering voxel removal operations have been performed and no topology-preserving voxel removal operation is possible.

Clearly, the total running time of the distance function approximation and carving stages is $O(n \log n)$, where n is the number of voxels in the bounding box.

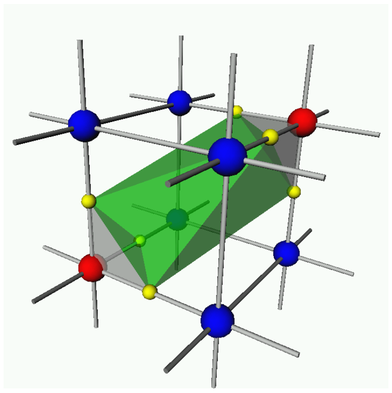

Figure 5: Example entry in the lookup table of [4]. Samples of positive value are shown in blue and the negative samples - in red. The yellow spheres are put at the midpoints of edges joining samples of different signs. The triangles that are stored in the lookup table are shown in green.

4.4 Volume Update

This part of the procedure forces the signs of the voxels to be consistent with the set carved in the previous step (positive outside and negative inside). Since no negative voxels are outside A, this amounts to changing the values of all positive voxels that are inside A to negative values.

4.5 Isosurface Extraction

So far our algorithms have concentrated on voxel sets, while virtually any real application requires sub-voxel accuracy in the output. The isosurfacing algorithm that is to work with our scheme needs to be able to transfer the topology from the voxel set consisting of negative voxels to the volume bounded by the isosurface. The standard algorithm introduced in the classical paper [28] would not work for this task: a simple counterexample can be obtained by placing negative values at two opposite corners of a grid cube and making all the other samples positive. The volume bounded by the isosurface extracted based on the [28] is disconnected while the voxel set defined by the negative value voxels is connected. However, for the three-dimensional variant of the scheme of [4], the isosurface turns out to be connected. In general, one can prove that its inside always has the same homotopy type (see [23] for the definitions) as the union of all negative voxels. The overall scheme of their algorithm is the same as that of [28]: the vertices of the isosurface are points on the grid edges joining samples of different signs. Their locations are computed assuming that the value varies linearly over the edge. The vertices within a single grid cube are then joined by triangles using the information stored in a lookup table. The lookup table is indexed by all possible configurations of positive and negative values at the vertices of a grid cube. In [4], it is built as follows. Let H be the convex hull of the set of points consisting of all of the cube's vertices of negative value and all centers of edges joining vertices having values of different signs. All triangles of H that are not contained in the boundary of the grid cube are put into the lookup table (Figure 5).

5 Multiresolution Implementation

The idea of image pyramids [5] allows us to considerably speed up our algorithm. We use subsampling to reduce the resolution of the volume after the distance function approximation stage. We use the minimum function as the subsampling filter. Thus, we group the voxels in $2 \times 2 \times 2$ blocks and form the subsampled volume from the minimum values of voxels within each of the blocks. In particular, this means that the voxels in the lower resolution volume corresponding to blocks which have at least one negative voxel of the higher resolution volume will be negative, and therefore spared in the carving phase. The loop (L) described in Section 4.3 is then run on the subsampled volume. When it terminates, the resolution is increased by replacing each voxel with the corresponding eight voxels of the original dataset. The carving stage as described in Section 4.3 is then executed with the priority queue initialized to hold all positive boundary voxels. Finally, the volume update and isosurfacing stages are performed (on the original resolution).

Clearly, the idea described above can also work for more than two levels of resolution, leading to additional speedup for large volumes. Our running time measurements show that the multiresolution implementation is an order of magnitude faster than an implementation of the basic algorithm. Figure 6 shows images of the voxel sets carved by our algorithm on consecutive levels of resolution for the signed distance function from the Buddha model.

Let us note that the speedup may come at a price of lower quality of the output mesh in the areas where the volume has been modified. This is because the multiresolution approach distorts the ordering of the voxel removal operations by their value (approximate distance from the input isosurface). This is not an important issue when the handles to be removed are small, since the change to the original volume is confined to their small neighborhood. For large handles, the multiresolution variant produces results that appear more irregular than the single-resolution algorithm. However, the two surfaces always tend to stay close to each other. For example, the Hausdorff distance (measured using the MESH tool [3]) between genus zero reconstructions of the Dragon model obtained using the single resolution variant of our algorithm and a multiresolution variant with 3 levels is equal to about 6.5 times the length of edges of the grid, while the distance from both of the reconstructed models to the original isosurface are about three times more. For genus 1 reconstructions for the same model, the Hausdorff distance between any of the two reconstructions and the original turns out to be less than the edge length of the grid. We have observed similar results for other models. Figure 8 shows closeups of reconstructions obtained using single resolution and multiresolution variants of our algorithm in the area inside the largest handle of the original isosurface. genus zero reconstruction of the dragon model. When removing large handles and the quality of the output surface in the areas where the volume is altered is of importance, one can use surface smoothing algorithms to improve the smoothness of the output isosurface obtained either using the single resolution or multiresolution variant of the algorithm.

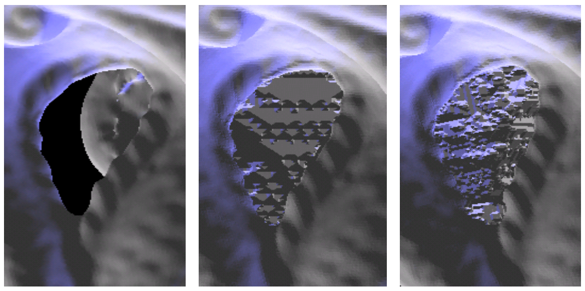

Figure 8: A closeup of a handle in the dragon model. On the left, we show a view through the handle of the original isosurface. The other two images show a similar view of two genus zero approximate isosurfaces extracted using our method (single resolution variant in the center and multiresolution variant with 3 resolution levels on the right).

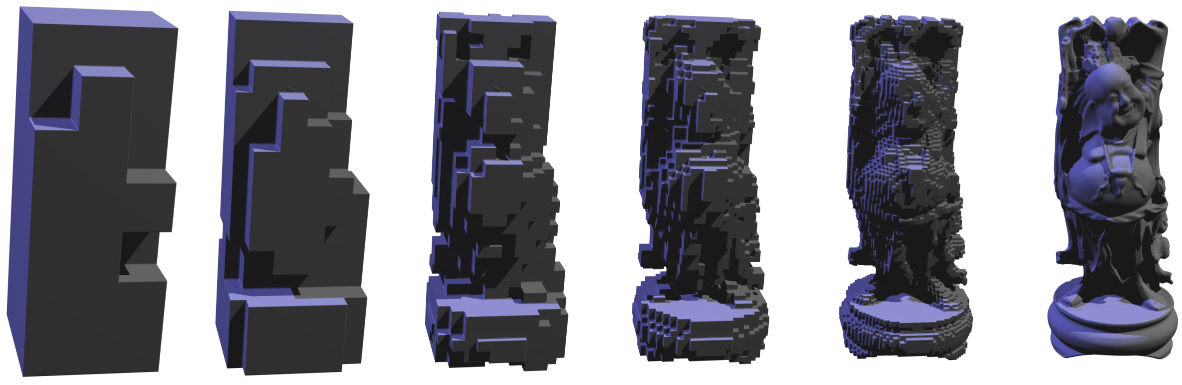

Figure 6: Volumes of increasing resolutions produced by the multiresolution algorithm for the signed distance function from the Buddha model. The last picture shows the output isosurface (genus 0).

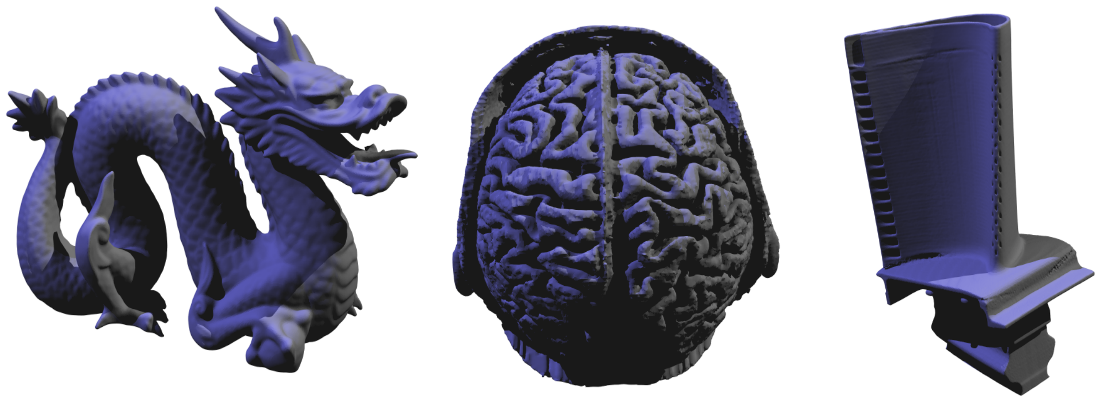

Figure 7: Isosurfaces obtained using our procedure. The dragon has genus 1 (and was obtained with $T = 1$), brain and blade - genus 0 ($T = 0$).

dataset	size	running time (seconds)
Buddha1	$215 \times 214 \times 516$	102
Buddha2	$181 \times 181 \times 434$	58
Dragon1	$365 \times 233 \times 516$	160
Dragon2	$307 \times 196 \times 434$	101
Blade1	$239 \times 306 \times 516$	151
Blade2	$206 \times 263 \times 444$	100
Bonsai (CT scan)	$256 \times 256 \times 256$	73
Brain (CT scan)	$256 \times 256 \times 167$	70
Skull (CT scan)	$256 \times 256 \times 68$	20
Engine (CT scan)	$256 \times 256 \times 110$	43

Table 1: Running times (on an 850MHz P3 system) for the test datasets.

6 Experimental Results

We have tested our algorithm for several signed distance functions from well known polygonal volumes and CT scans.

The information about the test datasets and the running time of our procedure is given in Table 1. We do not show the dependence of the running time on the number of topology-altering voxel removals since it is a matter of only a fraction of a second for all the test cases.

Figure 7 shows examples of isosurfaces obtained using our procedure. In spite of lack of theoretical guarantee (see Section 7), for all our test datasets genus of the output exhibits the same behavior: it is equal to the user-specified number T of topology altering operations, until it reaches the first Betti number (i.e. the number of through-holes) of the largest connected component of the union of all negative voxels in the input volume.

7 Discussion

We introduced a simple and efficient algorithm algorithm for extracting topologically simple isosurfaces from regularly sampled volume data. Experiments show that our method provides the user with precise control over the genus of the extracted isosurface. Such control is particularly desirable when the true topology of the isosurface to be extracted is known a priori. In such cases, our algorithm can be used to filter out the topological noise present in the isosurface.

We have concentrated on a variant of the algorithm that can only produce a connected isosurface. Alternatively, the cleanup step can be given up. This allows one to construct isosurfaces approximating the union of all outer components of the input isosurface. In this case, the parameter T i.e. the number of allowed topology-altering operations provides the user control over the *topological complexity* of the output isosurface. The topological complexity of a (possibly disconnected) surface is computed by summing the geni of all connected components and the number of connected components.

Even though our algorithm is so well behaved in practice, it seems to have few provable properties. We can certainly guarantee that if no topology-preserving operations are performed then the output isosurface has genus zero (see Section 4.3). However,

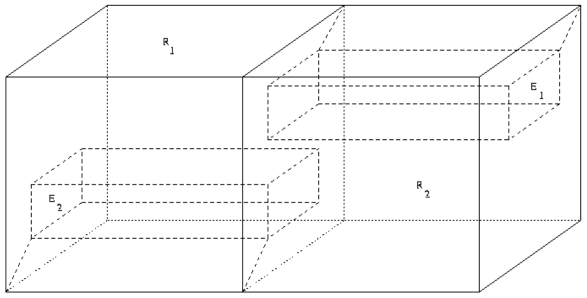

Figure 9: The house with two rooms. It defines two 'rooms' R_1 and R_2. In order to enter R_i one has to walk through the corridor through the other room, starting with the 'door' E_i. The house consists of the walls of the two rooms with the corridors' entrances and exits removed, the corridors' boundaries and the two rectangular walls connecting each of the corridors to the outer wall.

if the input isosurface is connected and has genus zero, the output isosurface does not have to be the same as the input surface. An example can be constructed using one of the well-known subsets of the 3D space that are retracts of R^3 but are not collapsible, e.g. the dunce hat [38] or the house with two rooms [7, Section I.2], briefly described in Figure 9. Consider a house with two rooms built of small voxels. Select its contractible 'dense' subset of voxels A (i.e. such that every cube intersecting the House has a cube of A close to it). For example, A can be a maximal voxel tree of this subset of voxels. By a voxel tree we mean a set of voxels for which the graph having the voxels as its nodes and with edges joining the nodes corresponding to intersecting voxels is a tree. Let the voxel values in this set be negative and the values of all the other voxels be positive. The algorithm will start by computing the approximate distance from the negative voxels, which is roughly equal to the distance from the House itself. Therefore, during the carving stage, our algorithm will be carving the House. When the set becomes one voxel wide, the algorithm will fail to find a voxel whose removal preserves the topology and will terminate. The final set will resemble the House and will be different from the tree consisting of negative voxels. We have not observed any topological locks as described above for any of our test datasets.

In future, we are planning to investigate ways to minimize the changes to the volume (e.g. the number of voxels whose values are altered in the volume update stage) needed to obtain an isosurface of prescribed topology. An approach that appears promising is to combine the results of our algorithm and a variation that grows the voxel set from a seed voxel inside the isosurface instead of carving it from its bounding box. The growing approach would work better for removing thin and long handles, for which it may be more natural to cut them. We are also planning to modify our algorithm so that it works for partially defined volume datasets like those considered in [9]. Even though our procedure requires volume traversal in a non memory coherent way, it is possible to implement it using an adaptive octtree representation rather than a full volume pyramid as described in this paper. Therefore, we hope that it will be effective for large volume datasets as well.

References

[1] Z. Aktouf, G. Bertrand, and L. Perroton. A three-dimensional holes closing algorithm. *Pattern Recognition Letters*, pages 523–531, 2002.

[2] Pierre Alliez and Mathieu Desbrun. Valence-driven connectivity encoding for 3D meshes. In *Eurographics 2001 Proceedings*, pages 480–489, 2001.

[3] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. Mesh: Measuring the errors between surfaces using the hausdorff distance. In *Proc. of the IEEE Conference in Multimedia and Expo (ICME) 2002*, volume 1, pages 705–708, August 2002.

[4] P. Bhaniramka, R. Wenger, and R. Crawfis. Iso-contouring in higher dimensions. In *Proceedings IEEE Visualization 2000*, pages 267–273, 2000.

[5] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communication*, 31(4):532–540, 1983.

[6] H. Carr, J. Snoyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003.

[7] M. M. Cohen. *A Course in Simple-Homotopy Theory*. Springer-Verlag, 1970.

[8] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[9] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH 1996*, pages 4–9, August 1996.

[10] H. Edelsbrunner. Surface reconstruction by wrpping finite point sets in space. In B. Aronov, S. Basu, J. Pach, and M. Sharir, editors, *Ricky Pollack and Eli Goodman Festschrift*, to appear.

[11] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *IEEE Symposium on Foundations of Computer Science*, pages 454–463, 2000.

[12] Jihad El-Sana and Amitabh Varshney. Controlled simplification of genus for polygonal models. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization '97*, pages 403–412, 1997.

[13] Thomas Gerstner and Renato Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings IVisualization 2000*, pages 259–266, 2000.

[14] J. Giesen and M. John. Surface reconstruction based on a dynamical system. *Computer Graphics Forum*, 21:363–371, 2002.

[15] J. Giesen and J. Matthias. The flow complex: A data structure for geometric modeling. In *Symposium on Discrete Algorothms 2003*, pages 285–294, 2003.

[16] X. Gu, S. Gortler, and H. Hoppe. Geometry images. In *Proceedings of ACM SIGGRAPH 2002*, pages 355–361, 2002.

[17] Stefan Gumhold and Wolfgang Strasser. Real time compression of triangle mesh connectivity. In *Proceedings ACM SIGGRAPH 98)*, pages 133–140, July 1998.

[18] I. Guskov and Z. Wood. Topological noise removal. In B. Watson and J. W. Buchanan, editors, *Proceedings of Graphics Interface 2001*, pages 19–26, 2001.

[19] Igor Guskov, K. Vidimce, Wim Sweldens, and Peter Schröder. Normal meshes. In *Proceedings of ACM SIGGRAPH 2000*, pages 95–102, 2000.

[20] X. Han, C. Xu, and J. L. Prince. A topology preserving deformable model using level sets. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR2001)*, pages 765–770, December 2001.

[21] X. Han, C. Xu, and J. L. Prince. A topology preserving geometric deformable model and its application in brain cortical surface reconstruction. In S. Osher and N. Paragios, editors, *Geometric Level Set Methods in Imaging, Vision and Graphics*. Springer Verlag, August 2002.

[22] J. C. Hart. Morse theory for implicit surface modeling. In H-C. Hege and K. Polthier, editors, *Mathematical Visualization*, pages 257–268. Springer-Verlag, October 1998.

[23] John G. Hocking and Gail S. Young. *Topology*. Addison-Wesley Publishing Company, Reading, Mass., 1961.

[24] Daniel J Jobson, Zia-ur Rahman, and Glenn A Woodell. Retinex image processing: Improved fidelity to direct visual observation. In *Proceedings of the IS&T Fourth Color Imaging Conference: Color Science, Systems, and Applications*, volume 4, pages 124–125, 1995.

[25] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proceedings of ACM SIGGRAPH 98*, pages 95–104, 1998.

[26] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project. In Kurt Akeley, editor, *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 131–144, New York, 2000. ACM, ACM Press / ACM SIGGRAPH.

[27] Hlio Lopes, Jarek Rossignac, Andrzej Szymczak, Alla Safanova, and Geovan Tavares. Edgebreaker: A simple algorithm for surfaces with handles. In *7th ACM SIGGRAPH Symposium on Solid Modeling and Its Applications*, 2002.

[28] W. E. Lorensen and H. E. Cline. Marching cubes: A high-resolution 3d surface reconstruction algorithm. *ACM Computer Graphics*, 21(3):163–169, 1987.

[29] J. W. Milnor. *Morse Theory*. Princeton University Press, Princeton, NJ, 1963.

[30] F.S. Norruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. Technical Report GIT-GVU-99-37, GVU Center, Georgia Institute of Technology, 1999.

[31] V. Pascucci and K. Cole-McLaughlin. Efficient computation of the topology of level sets. In *Proceedings Visualisation 2002*, 2002.

[32] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings ACM SIGGRAPH 2001*, pages 409–416, 2001.

[33] J. A. Sethian. Fast marching methods. *SIAM Review*, 2(41):199–235, 1999.

[34] S. P. Tarasov and M. N. Vyalyi. Construction of contour trees in 3d in $O(n \log n)$ steps. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, pages 68–75, June 1998.

[35] C. Touma and Craig Gotsman. Triangle mesh compression. In *Proceedings Graphics Interface 1998*, pages 26–34, 1998.

[36] M. vanKreveld, R. vanOostrum, C. L. Bajaj, and D. L. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the 13th Annual Symposium on Computational Geometry*, pages 212–220, June 1997.

[37] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Isosurface topology simplification. Technical Report MSR-TR-2002-28, Microsoft Research, 2002.

[38] E. C. Zeeman. On the dunce hat. *Topology*, 2:341–352, 1964.

[39] H. K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *1st IEEE Workshop on Variational and Level Set Methods, in conjunction with the 8th International Conference on Computer Vision (ICCV), Vancouver, Canada*, pages 194–202, 2001.