Topology Preserving and Controlled Topology Simplifying Multiresolution Isosurface Extraction

Thomas Gerstner

Department for Applied Mathematics
University of Bonn
gerstner@iam.uni-bonn.de

Renato Pajarola

Information and Computer Science
University of California Irvine
pajarola@acm.org

Abstract

Multiresolution methods are becoming increasingly important tools for the interactive visualization of very large data sets. Multiresolution isosurface visualization allows the user to explore volume data using simplified and coarse representations of the isosurface for overview images, and finer resolution in areas of high interest or when zooming into the data. Ideally, a coarse isosurface should have the same topological structure as the original. The topological genus of the isosurface is one important property which is often neglected in multiresolution algorithms. This results in uncontrolled topological changes which can occur whenever the level-of-detail is changed. The scope of this paper is to propose an efficient technique which allows preservation of topology as well as controlled topology simplification in multiresolution isosurface extraction.

CR Categories: G.1.2 [Numerical Analysis]: Approximation—Approximation of Surfaces and Contours I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: tetrahedral grid refinement, implicit surface approximation, level–of–detail, topological genus, critical points

1 Introduction

Isosurface extraction is a very common and useful tool for the visualization of volume data. In the last years, the resolution of volumetric data sets has been increasing dramatically. This applies to typical measurement data arising e.g. in medical imaging as well as to output of large–scale numerical simulations.

The need for multiresolution methods becomes apparent when such large data sets have to be visualized interactively as required by many applications such as computer aided surgery or scientific visualization. Without multiple resolutions, the number of triangles which make up an isosurface can easily exceed the number of triangles that can be rendered at interactive frame rates. Also, the isosurface extraction phase may simply be too slow in order to be able to change the isovalue interactively.

Multiresolution methods address both of these problems. Basically they enable the user to control the accuracy of the extracted isosurfaces by reducing the mesh complexity (or vice versa). This way, he or she is able to explore large volume data using simplified approximations of the isosurface to view the overall structure of the isosurface, and to use more detailed resolutions when zooming into the data set for an inspection of a particular area. The user can also adjust the isovalue interactively at coarser levels–of–detail.

The different representations of the isosurface should aid the user in navigating and exploring the data set. They should prevent him or her from missing interesting parts and provide a correct impression of the grand structure of the data. The topological genus of the isosurfaces is certainly one variable the user should have control over.

A multiresolution isosurface extraction algorithm is called topology preserving if all coarser isosurface approximations have the same topological genus as the highest resolution isosurface. It is called topology simplifying if the genus changes for different LODs. In the latter case, however, the user should be able to control the extent and type of possible topological changes.

In this paper we present a method which allows preservation and controlled simplification of topology for multiresolution isosurfaces under real–time constraints. We will use a methodology which is based on tetrahedral volume meshes instead of the triangle surface meshes. Although the focus will here be on regular gridded data, the approach can be generalized to curvilinear or unstructured grids generated by bisection in a straightforward manner.

The remainder of this paper is organized as follows. In Section 2 we review previous work. Section 3 briefly describes the construction of multiresolution isosurfaces based on tetrahedral bisection. Algorithms for topology preservation are then considered in Section 4. Controlled topology simplification methods are discussed in Section 5. The conclusions in Section 6 complete the paper.

2 Related Work

Very often, an isosurface represented as a triangle mesh is extracted in a preprocessing step. The extraction can be very time–consuming when standard marching algorithms [14, 18] are used. Therefore, a variety of methods have been designed to speed up the extraction step [11, 13, 22], or to limit the extraction to the visible triangles [7, 12]. General topological problems in isosurface generation have been addressed in [3, 16, 21, 24].

One way to turn an isosurface into a multiresolution representation is to apply a mesh simplification algorithm on the extracted triangle mesh. For a detailed overview of the large variety of available methods see the recent surveys of [10, 19]. In this context, controlled topology simplification of isosurface meshes has been considered in [1, 9]. The whole approach has several disadvantages, though. It requires extraction of the isosurface at the finest resolution first, which makes interactive adjustments of the isovalue impossible unless isosurfaces for all possible isovalues are extracted. Moreover, the construction of a multiresolution triangle mesh hierarchy is generally slow and needs involved data structures.

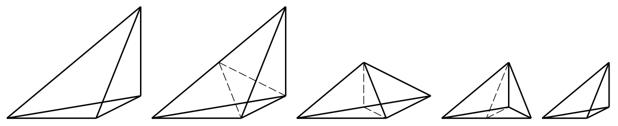
Figure 1: Decomposition of a tetrahedron by recursive bisection.

On the other hand, it is possible to extract multiresolution isosurfaces directly from the volume data. Thereby, a multiresolution hierarchy is not inferred on the isosurface itself, but on the underlying 3D data set. A coarser isosurface is simply defined as the isosurface of a less detailed approximation of the data. Isosurface extraction is then usually done in a top–down fashion. Starting with an initial approximation, details are added in areas where an error indicator shows a large local error with respect to the data on the finest resolution. If the error drops below a user–defined threshold, the algorithm stops the refinement locally and extracts the isosurface at the current LOD. In some cases the isosurface does not even have to be stored as a triangle mesh but is directly rendered during the data traversal.

If the data domain is refined adaptively (i.e. not uniformly) it can happen that the extracted isosurface contains holes (cracks) at transition zones where the mesh resolution changes. Different solutions have been devised for this problem, including remeshing [4, 8], point insertion [20], filling, adaptive projection [17], and saturation of the error indicator [6, 7].

In the following, we introduce a straightforward and natural way to incorporate topology questions into adaptive multiresolution isosurface extraction based on the error saturation technique.

3 Multiresolution Isosurface Extraction

In this section, we will shortly explain the construction of multiresolution isosurfaces based on tetrahedral meshes. This has already been described in detail in previous works but for clarity and reader's convenience we repeat the basic steps here.

3.1 Tetrahedral Bisection

The multiresolution approach considered here is based on recursive bisection of tetrahedra. This refinement scheme is well known from numerical methods for partial differential equations [15] and has recently also found its way into computer graphics and visualization [6, 23]. In contrast to octree approaches, where a trilinear interpolation is used, the data models of the triangulated isosurface and the volumetric grid coincide (they use both piecewise linear interpolation). Therefore a lot of problems arising from different data models are eliminated. This also includes problems related to the topology of the isosurfaces.

Let us consider a nested hierarchy of tetrahedral grids $\mathcal{T}^l$ with level $0 \leq l \leq l_{\text{max}}$. The tetrahedra are refined by *recursive bisection*: for a tetrahedron T the midpoint of the longest edge $e_{\text{ref}}(T)$ is chosen as a new node $x_{\text{ref}}(T)$, and the tetrahedron is split at the face spanned by $x_{\text{ref}}(T)$ and the two nodes of T opposite to $e_{\text{ref}}(T)$ into two child tetrahedra $C_1(T)$ and $C_2(T)$ (Figure 1). Through recursive application a binary tree hierarchy is inferred on the tetrahedra.

We assume that the input volume data is arranged in a uniform grid with n^3 nodes, $n = 2^k + 1$. The initial tetrahedral mesh $\mathcal{T}^0$ consists of the six tetrahedra whose vertices are adjacent corners of the cube and which all share the same diagonal of that cube (see Figure 2 left). Thus, all refinement vertices $x_{\text{ref}}(T)$ will fall onto grid points of the original data set.

The tetrahedral grid hierarchy does not need to be constructed explicitly. The binary tree of tetrahedra can be traversed implicitly

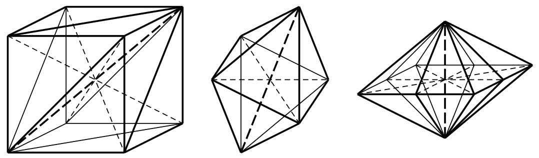
Figure 2: The three types of surrounding polyhedra which arise during the refinement. Refinement edges are bold–dashed, spoke edges are light–dashed.

and all required information can be computed on–the–fly using the procedure below. It is initially called for the six tetrahedra on the coarsest mesh $\mathcal{T}^0$ with their longest edge given by (x_1, x_2):

```
recursive_descent(Coord x1, x2, x3, x4, Int l) {
   xref=(x1+x2)/2;
   if (l < lmax) {
     if ((l mod 3) == 0) {
       recursive_descent(x1, x3, x4, xref, l+1);
       recursive_descent(x2, x4, x3, xref, l+1);
     }
     else {
       recursive_descent(x1, x3, x4, xref, l+1);
       recursive_descent(x2, x3, x4, xref, l+1);
     }
   }
}
```

Using linear interpolation inside each tetrahedron, a piecewise linear function on $\mathcal{T}^l$ uniquely described by the data values $f(x_i)$ at the corresponding nodes is obtained.

3.2 Extraction Algorithm

The adaptive multiresolution isosurface algorithm is based on a depth first traversal of the binary tree. On every tetrahedron for a stopping criterion is checked. If it is true, the algorithms stops and extracts the local isosurface using the look–up table of the marching tetrahedra algorithm [18]. Otherwise, the two children are visited recursively. If the algorithms stops on a specific tetrahedron T and refines another tetrahedron which shares the refinement edge, an inconsistency occurs at the hanging node $x_{\text{ref}}(T)$. This leads to cracks in the isosurface. Therefore, it is necessary to ensure that whenever a tetrahedron is refined, all tetrahedra sharing its refinement edge are refined as well. This can be achieved through definition of *error indicators* η on the refinement vertices, i.e. $\eta(T) = \eta(x_{\text{ref}}(T))$, and choosing $\eta(T) < \varepsilon$ as a stopping criterion for some user specified threshold value ε.

If the error indicator fulfills the saturation condition

$$\eta(T) \geq \eta(C_i(T)),\ i = 1, 2,$$

for all $T \in \mathcal{T}^l$ with $l < l_{\text{max}}$, no hanging nodes can occur for all possible values of ε [17, 23]. If an error indicator η does not fulfill this condition it can easily be adjusted in a preprocessing step. In a level–wise bottom up traversal of the hierarchy it is possible to construct a minimal saturated error indicator $\bar{\eta}$ by setting

$$\bar{\eta}(T) := \max\{\eta(T), \bar{\eta}(C_1(T)), \bar{\eta}(C_2(T))\}.$$

Furthermore, the traversal of the binary tree is also stopped locally if the tetrahedron is not a candidate for an intersection with the isosurface. In our case, it is checked whether the current isovalue is contained in the interval consisting of all data values inside the tetrahedron. This information can either be explicitly computed in a bottom–up traversal of the tree [22] or obtained from already available error indicator values [7]. This way, the complexity of the extraction algorithm is of the order of the output (i.e. the number of drawn triangles), nearly independent of the size of the input.

4 Topology Preservation

The shape of the extracted isosurfaces will greatly depend on the choice of the error indicator. There is a large variety of geometric error indicators that can be used to control tetrahedral grid refinement for visualization purposes [7]. However, in general they don't allow to control possible changes in the topological structure of the isosurface, even if a conservative error measurement (e.g. based on the L_∞–norm) is used.

4.1 Critical Points

One important notion in topology is that of *critical points*. Critical points can be defined as points in space where an isosurface would change its genus or number of components. Since our data model is piecewise linear, critical points can only arise at vertices of the tetrahedral mesh. Note that this would not be true for octahedral meshes and trilinear interpolation.

The basic idea to incorporate topology preservation into our multiresolution algorithm is the following assumption: if an adaptive tetrahedral mesh contains all critical points, then any approximate isosurface is of the same genus as the corresponding isosurface on the finest resolution.

Critical points, however, have to be defined hierarchically based on the tetrahedral bisection hierarchy since non–critical points on the finest resolution can become critical with respect to a coarser resolution [2]. Therefore, a refinement vertex is called a *hierarchical critical point* if the genus of an isosurface restricted to all tetrahedra sharing the refinement edge changes when the tetrahedra are refined. If an adaptive tetrahedral mesh contains all hierarchical critical points, topology preservation has been achieved.

4.2 Lookup–Tables

Although hierarchical critical points can be found in a preprocessing step, their identification should be done as efficiently as possible. To extract isosurfaces for isovalues $f(x_{\text{ref}}) - \tau$ and for $f(x_{\text{ref}}) + \tau$ (τ small), and to compare their connectivity would be a tedious and too time–consuming task. However, since the topology of the isosurface only depends on the relative ordering of the data values at the vertices of the tetrahedra surrounding the refinement edge, critical points can be identified quite efficiently based on look–up tables. In [23], however, an overly simplified and conservative table has been proposed which generates many unnecessarily refined tetrahedra.

Let us define the *surrounding polyhedron* of a refinement edge as the boundary of all adjacent tetrahedra sharing the edge. In our case of regular gridded data sets and recursive bisection, three types of polyhedra will arise: a triangulated cube, an octahedron, and a diamond (see Figure 2). The cube will apply to tetrahedra of levels $(l \bmod 3) = 0$, the octahedron to levels $(l \bmod 3) = 1$, and the diamond to levels $(l \bmod 3) = 2$.

The m vertices of a surrounding polyhedron are marked with a $+$ sign if the data value at the vertex is larger than the value at the refinement vertex, and with a $-$ sign if it is smaller. The look–up table for each type of polyhedron consists of the 2^m possible cases and contains a bit indicating if the refinement vertex is critical or not. Simplified look–up tables can be constructed by the identification of all symmetry classes for the $+/-$ bit-patterns of the different types of polyhedra and checking for criticality in each of these classes.

Figure 3 shows the resulting 25 classes for the cube. Not shown is case 1 where all vertices have the same sign which would lead to a local minimum or maximum which is always critical. The images show isosurfaces after refinement for an isovalue of $f(x_{\text{ref}})$. Critical points arise if two or more components of an isosurface have a (non–manifold) point connection at the refinement vertex in the middle of the cube. This is true for classes 4b, 5a, 5b, 7b, 8b, 12b, and 13. In these cases, changing the isovalue slightly in positive or negative direction would lead to different connectivities of the isosurface components.

The main symmetry classes are identical to the marching cubes look–up table [14], but special care has to be taken to include a few more subclasses. These subclasses arise since the endpoints of the refinement edge (from front–bottom–left to back–top–right) have different connectivity than the other vertices of the polyhedron.

4.3 Automatic Construction of Look–up Tables

Manual construction of look–up tables is of course potentially erroneous since subcases may easily be forgotten or wrongly classified. However, there is an efficient and exact way to construct look–up tables for an automatic identification of critical points automatically. It consists of the following four steps:

1. construct the edge graph of the polyhedron,
2. delete all edges between a $+$ and a $-$ node from the graph,
3. count the remaining connected components of the graph,
4. if the number of components is 1 or greater than 2, then the refinement vertex is critical.

As an example, let us consider the case shown for class 13 in Figure 3. The vertices of the cube are numbered front to back, bottom to top, left to right. This way, the signs of the vertices are given by $--++++--$. Note that the cube is triangulated and therefore every quadrilateral face of the cube has a diagonal edge as depicted in Figure 2.

1. edge graph for the cube:

	$-$	$-$	$+$	$+$	$+$	$+$	$-$	$-$
$-$	$\times$							
$-$	$\times$	$\times$						
$+$	$\times$		$\times$					
$+$	$\times$	$\times$	$\times$	$\times$				
$+$	$\times$				$\times$			
$+$	$\times$	$\times$			$\times$	$\times$		
$-$	$\times$		$\times$		$\times$		$\times$	
$-$		$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

2. delete edges:

	$-$	$-$	$+$	$+$	$+$	$+$	$-$	$-$
$-$	$\times$							
$-$	$\times$	$\times$						
$+$			$\times$					
$+$			$\times$	$\times$				
$+$					$\times$			
$+$					$\times$	$\times$		
$-$	$\times$						$\times$	
$-$		$\times$					$\times$	$\times$

3. count components (transitive hull):

	$-$	$-$	$+$	$+$	$+$	$+$	$-$	$-$
$-$	$\times$							
$-$	$\times$	$\times$						
$+$			$\times$					
$+$			$\times$	$\times$				
$+$					$\times$			
$+$					$\times$	$\times$		
$-$	$\times$	$\times$					$\times$	
$-$	$\times$	$\times$					$\times$	$\times$

4. 3 components $\rightarrow$ critical

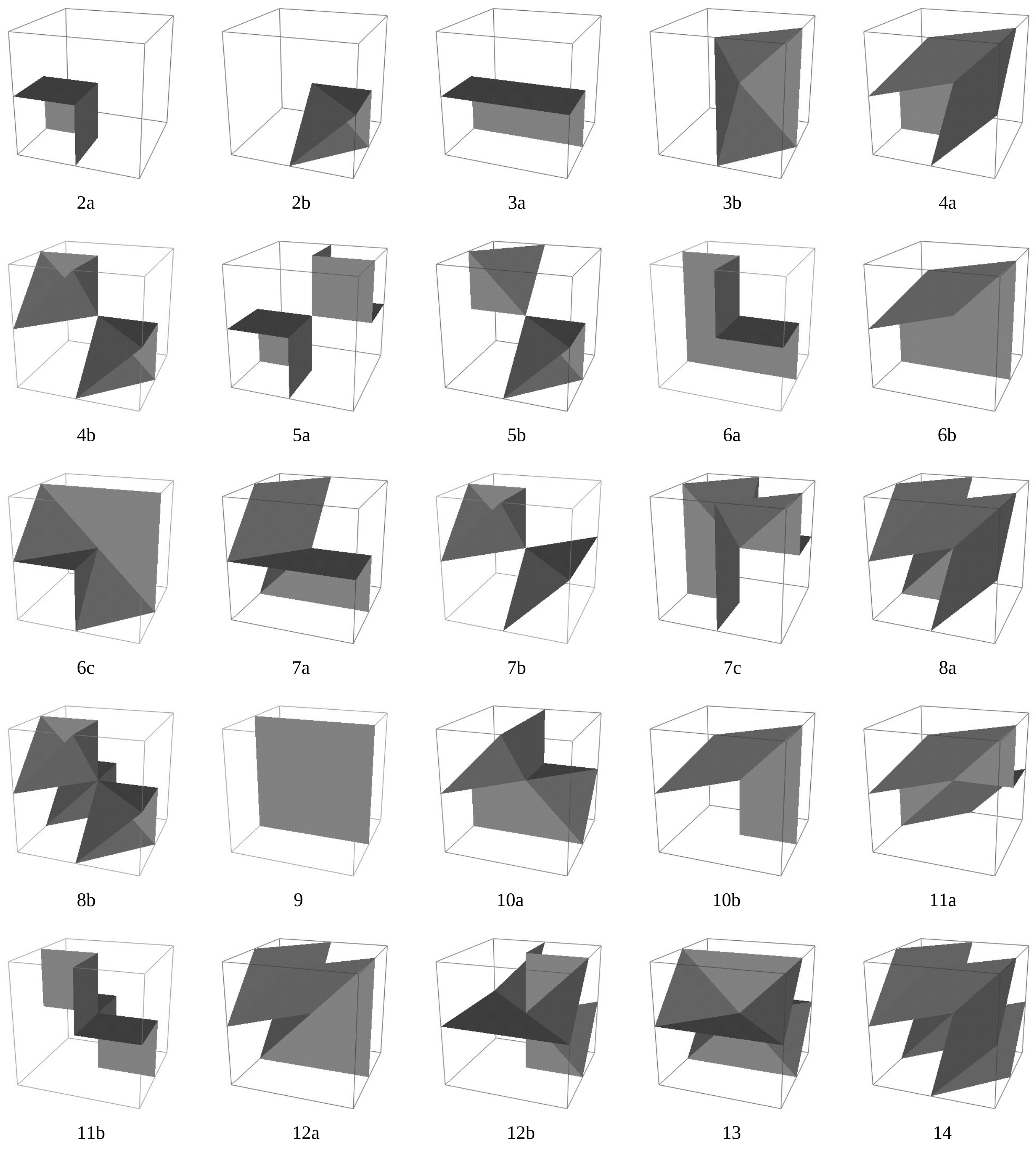

Figure 3: Topological classes for the cube.

The total running time of the algorithm is of order $O(2^m \cdot m^2)$ for each type of polyhedron. Of course, this computation has to run only once in case of regular grid refinement. Accordingly, for recursive bisection refinement, critical points will arise in 68/256 cases for the cube, in 8/64 cases for the octahedron, and in 400/1024 cases for the diamond.

Critical points located on boundary of the data set do not need any special treatment. The existing lookup–tables can be used if missing vertices (respectively their signs) are mirrored across the boundary. This way, boundary critical points will also be classified correctly.

Note that this automatic construction of look–up tables is not restricted to regular grids, but can be applied to any irregular tetrahedral mesh generated by edge bisection.

4.4 Critical Intervals

A straightforward way to incorporate topology preservation into the multiresolution isosurface extraction algorithm is to set error indicator values at hierarchical critical points to infinity (as proposed in [23]). However, for a given isosurface only a subset of all critical points is really relevant and many tetrahedra would be refined needlessly.

Therefore, every critical point will be assigned a *critical interval*. The critical interval is defined as the range of isovalues for which the isosurface changes topology when the tetrahedra are refined at the critical point. With this information, the extraction algorithm is able to refine only at those critical points whose critical interval contains the current isovalue.

The critical intervals are relatively easy to obtain. They are defined as

$$I(x_{\text{ref}}) := [f(x_{\text{ref}}), \min\{f(x_1), f(x_2)\}]$$

for refinement edges whose endpoints x_1 and x_2 have positive signs, and as

$$I(x_{\text{ref}}) := [\max\{f(x_1), f(x_2)\}, f(x_{\text{ref}})]$$

for negative signs. Note that refinement edges whose endpoints have different signs never lead to critical points.

In order to avoid missing critical points during the tree traversal, critical intervals have to be saturated similarly to the error indicator values. Each tetrahedron is assigned a critical interval which is defined as the critical interval of its refinement vertex. In a level–wise bottom–up traversal of the tetrahedral bisection tree, critical intervals are merged taking the upper and lower bounds of the intervals of the current tetrahedron and its two children.

4.5 Examples

In order to show the performance of the algorithm let us consider the zero set of the algebraic function

$$f(x, y, z) = \sqrt{x^2 + y^2} - (\frac{1}{2}x + \frac{1}{2}y - z + 0.01)^2$$

defined on $[-1, 1]^3$ sampled on a uniform 65^3 grid. This function has a strong diagonal singularity near the origin as can be seen in Figure 4. In Table 1 we compare the number of triangles for various geometric error threshold values ε (L_∞–norm). Topology preservation will retain the original topology at a low triangle cost since the tetrahedral grid refinement can strongly adapt towards the singularity. Without topology preservation, this delicate structure will become lost for large error thresholds.

As a second example we will consider the geometrically more complicated buckyball data set (courtesy of AVS). Figure 5 shows isosurfaces for different geometric error threshold values and for an isovalue where the data set inhibits a very complex topological structure. The values of ε and the corresponding triangle counts are listed in Table 2. Without topology preservation the isosurface does not maintain the correct connectivity over different LODs (third image from the left in top row), or even completely falls apart (fourth image in top row). With topology preservation the original topology is exactly retained. In the latter case, however, the amount of geometric simplification is limited due to the high complexity of the topology which the isosurface exhibits nearly everywhere in space. After a certain LOD, increasing the geometric error threshold will not result in lower triangle counts any more. The lower bound will be reached for a tetrahedral mesh that consists of all hierarchically critical points.

ε	without topology preservation	with topology preservation
0	59290	59290
1/64	25456	25464
1/16	7124	7528
1/4	1936	3503
1	374	3095

Table 1: Triangle counts for the algebraic function for varying geometric error threshold values ε.

ε	without topology preservation	with topology preservation
0.0	395798	395798
0.01	153539	163818
0.05	29594	110101
0.1	12552	109981

Table 2: Triangle counts for the buckyball data set for varying geometric error threshold values ε.

4.6 Performance

In this algorithm, the extraction speed per triangle is almost as fast as the rendering speed for a triangle on modern graphics workstations. In [6] a combined extraction and drawing rate of up to 300.000 triangles per second for a single processor and up to 800.000 triangles per second for a multiprocessor machine have been reported. This speed can also be measured for our topology–preserving version since only one additional check (namely if the isovalue is contained in the current critical interval) is necessary for each tetrahedron during the tree traversal. Typical mesh simplification algorithms have lower simplification rates of at least 2–3 orders of magnitude [19].

Let us also take a look at the memory overhead. In addition to the data the error indicator values, the minmax bounds, and the critical intervals have to be stored for each refinement vertex up to the second finest resolution. In some cases, the minmax bounds can be derived from the error indicator and therefore don't need to be stored [7]. However, with conservative quantization all additional information can be stored with roughly the same memory requirements as the data itself (see [6]).

The preprocessing step for the computation of this information requires a single level–wise bottom–up traversal of the whole data set which takes at most a few seconds for the considered examples. If the data set is stored in traversal order, the necessary computations can even be done during the data reading phase.

5 Controlled Topology Simplification

As we have seen in the previous example, topology preservation will usually infer a limit on the minimum number of triangles in the isosurface mesh. Depending on the application, it might be desirable to further simplify the resulting triangle meshes leading to controlled changes of genus of the isosurface components.

While the original — non–topology preserving — multiresolution algorithm simplifies topology naturally, it does so in an uncontrolled fashion. In principle it would be possible to employ controlled topology simplifying triangle mesh reduction schemes such as filtering and resampling [9], or α–hulls [5] in a postprocessing step of the isosurface extraction. However, these algorithms work in a bottom–up fashion on the extracted triangle mesh at the highest resolution and would therefore not meet our real–time performance constraints.

5.1 Choices for Simplification

The topology preserving algorithm described in the previous section can easily be transformed into a controlled topology simplifying one. To this end, let us define a *simplification weight* $w(x_{\text{ref}})$ on all hierarchical critical points which indicates the importance of that particular point. The topology simplifying algorithm will then be able to discard those critical points which are not important, i.e. $w(x_{\text{ref}}) < \delta$ where δ is a user–defined threshold.

This way, the user has complete control over topology simplification by the specification of a suited weight function w on the critical points together with a threshold δ. The actual values of the weight w at the hierarchical critical points can be precomputed together with the error indicator and the critical intervals.

An appropriate choice for the weight w strongly depends on the type of application and structure of the data set. Let us consider a few possibilities for the simplification weight:

- An obvious choice for w would be the size of the critical interval. This requires no extra computational effort in the presented framework. By a multiplication with the maximum gradient of all adjacent tetrahedra, one has an indicator for the amount of topological changes in the isosurface structure introduced by refinement of the tetrahedra surrounding the critical point.

- Another possibility would be the size of the corresponding local isosurface, i.e. the number of tetrahedra the family of isosurfaces within the critical interval would intersect with. This provides a control mechanism over the local geometric complexity.

- One could also choose bounds for the distance between isosurface components that are separated by the critical point, which of course makes only sense in the case of saddle points. This allows control of the geometric separation between surface components, i.e. disconnected components could be merged and simplified if the distance between them is small enough.

5.2 Example

An extensive comparison of simplification techniques for different applications would certainly exceed the limits of this paper. Therefore, we will here just consider a simple but characteristic example. Let us look at a typical bio–medical data set, the famous CT scan of a lobster (courtesy of AVS). In Figure 6 we see that original isosurface (1089004 triangles) is very noisy. The topology preserving algorithm can reduce the number of triangles to 609893, but the noise in the data set will of course remain. Let us therefore define the weight function dependent on the size of the critical interval,

$$w(x_{\text{ref}}) := |I(x_{\text{ref}})|.$$

Topology simplification will then reduce these artifacts nicely and maintain the overall topological structure of the animal. The simplified isosurfaces provide visually better representations of the data set with only 367271 triangles for $\delta = 0.2$ and 262357 triangles for $\delta = 0.7$, respectively.

6 Concluding Remarks

We have shown how topology preservation and controlled topology simplification can be achieved in multiresolution isosurface extraction on hierarchical tetrahedral meshes generated by recursive bisection. Thereby, we did not focus on methods working on the isosurface triangle mesh but on the underlying 3D volume data set itself.

While we mainly considered tetrahedral meshes based on regular refinement in this paper, our topology considerations apply to more irregular tetrahedral meshes generated by bisection as well. Refinement methods for tetrahedral mesh generation can thereby consider the most critical points for insertion, whereas decimation methods will consider non–critical or less–critical points for removal. With appropriate modifications the methodology could be extended to handle other hierarchical tetrahedral meshes as well although this would probably require an identification of more cases and look–up tables.

Finally, the proposed methods may not only be important by themselves, but can also be used as a fast preprocessor to other algorithms which rely on a topologically consistent input mesh.

References

[1] C. Andujar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Sole. Automatic Generation of Multiresolution Boundary Representations. *Computer Graphics Forum*, 15(3):87–96, 1996.

[2] C. Bajaj and D. Schikore. Topology Preserving Data Simplification with Error Bounds. *Computers & Graphics*, 22(1):3–12, 1998.

[3] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. Reconstruction of Topologically Correct and Adaptive Trilinear Isosurfaces. *Computers & Graphics*, 24(3):399–418, 2000.

[4] P. Cignoni, C. Montani, and R. Scopigno. MagicSphere: An Insight Tool for 3D Data Visualization. *Computer Graphics Forum*, 13(3):317–328, 1994.

[5] J. El-Sana and A. Varshney. Controlled Simplification of Genus for Polygonal Models. In *Proceedings IEEE Visualization '97*, pages 403–412. IEEE Press, 1997.

[6] T. Gerstner and M. Rumpf. Multiresolutional Parallel Isosurface Extraction based on Tetrahedral Bisection. In M. Chen, A. Kaufman, and R. Yagel, editors, *Volume Graphics*, pages 267–278. Springer, 2000.

[7] T. Gerstner, M. Rumpf, and U. Weikard. Error Indicators for Multilevel Visualization and Computing on Nested Grids. *Computers & Graphics*, 24(3):363–373, 2000.

[8] R. Grosso, C. Lürig, and T. Ertl. The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data. In *Proceedings IEEE Visualization '97*, pages 387–394. IEEE Computer Society Press, 1997.

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

[10] P. Heckbert and M. Garland. Survey of Surface Approximation Algorithms. In *SIGGRAPH '97 Course Notes*, 1997.

[11] T. Itoh and Y. Yamaguchi. Isosurface Generation using Extrema Graphs. In *Proceedings IEEE Visualization '94*, pages 77–83. IEEE Computer Society Press, 1994.

[12] Y. Livnat and C. Hansen. View Dependent Isosurface Extraction. In *Proceedings IEEE Visualization '98*, pages 175–180. IEEE Computer Society Press, 1998.

[13] Y. Livnat, H. Shen, and C. Johnson. A Near Optimal Isosurface Extraction Algorithm using the Span Space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–83, 1996.

[14] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[15] J. Maubach. Local Bisection Refinement for n-simplicial Grids generated by Reflection. *SIAM J. Sci. Comp.*, 16:210–227, 1995.

[16] B. Natarajan. On Generating Topologically Consistent Isosurfaces from Uniform Samples. *The Visual Computer*, 11(1):52–62, 1994.

[17] M. Ohlberger and M. Rumpf. Adaptive Projection Methods in Multiresolutional Scientific Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 4(4):74–94, 1998.

[18] B. Payne and A. Toga. Surface Mapping Brain Function on 3D Models. *IEEE Computer Graphics and Applications*, 10(5):33–41, 1990.

[19] E. Puppo and R. Scopigno. Simplification, LOD and Multiresolution Principles and Applications. In *Eurographics '97 Tutorial Notes*, 1997.

[20] R. Shekhar, E. Fayyad, R. Yagel, and J. Cornhill. Octree–Based Decimation of Marching Cubes Surfaces. In *Proceedings IEEE Visualization '96*, pages 335–244. IEEE Computer Society Press, 1996.

[21] A. Van Gelder and J. Wilhelms. Topological Considerations in Isosurface Generation. *ACM Transactions on Graphics*, 13(4):337–375, 1994.

[22] J. Wilhelms and A. Van Gelder. Octrees for Faster Isosurface Generation. *ACM Transactions on Graphics*, 11(3):201–227, 1992.

[23] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution Tetrahedral Framework for Visualizing Volume Data. In *Proceedings IEEE Visualization '97*, pages 135–142. IEEE Computer Society Press, 1997.

[24] Y. Zhou, W. Chen, and Z. Tang. An Elaborate Ambiguity Detection Method for Constructing Isosurfaces within Tetrahedral Meshes. *Computers & Graphics*, 19(3):355–364, 1995.

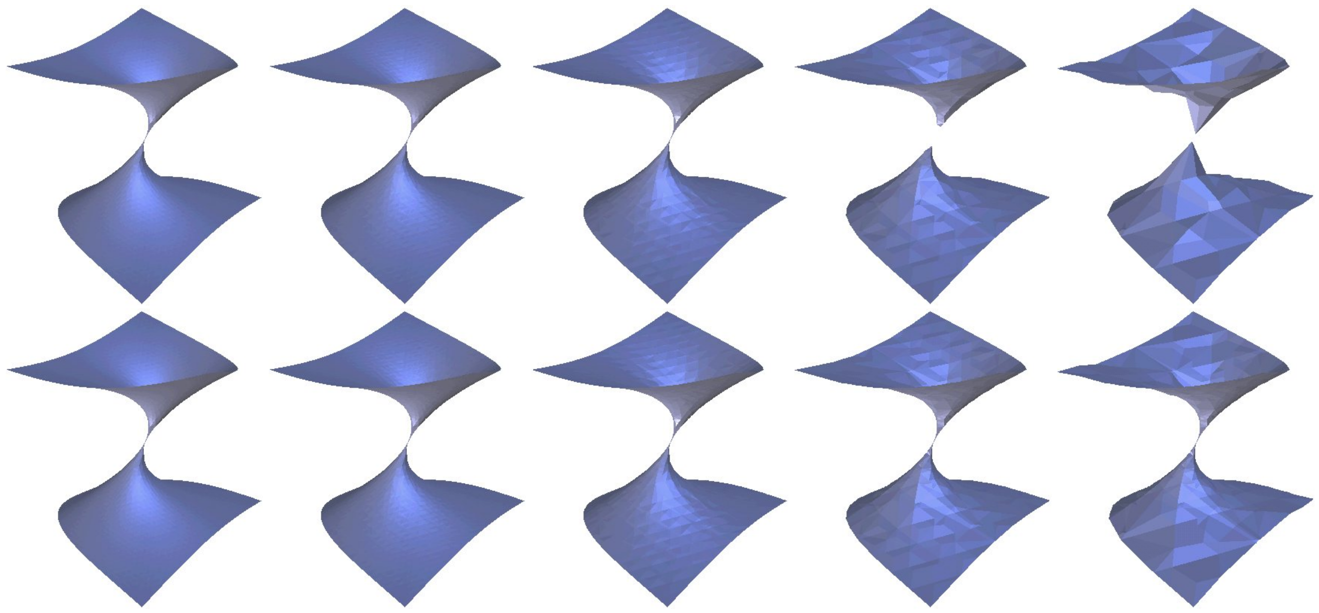
Figure 4: Isosurfaces of an algebraic function without (upper row) and with (lower row) topology preservation for varying error tolerances.

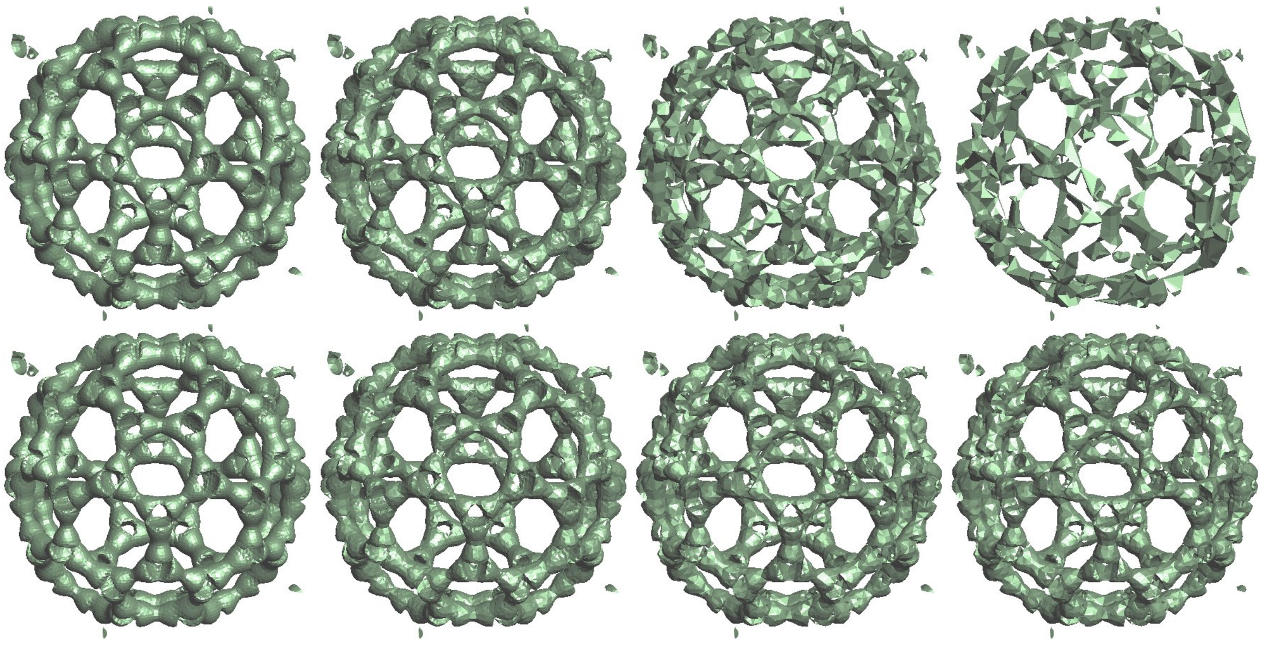
Figure 5: Isosurfaces of the buckyball data set without (upper row) and with (lower row) topology preservation for varying error tolerances.

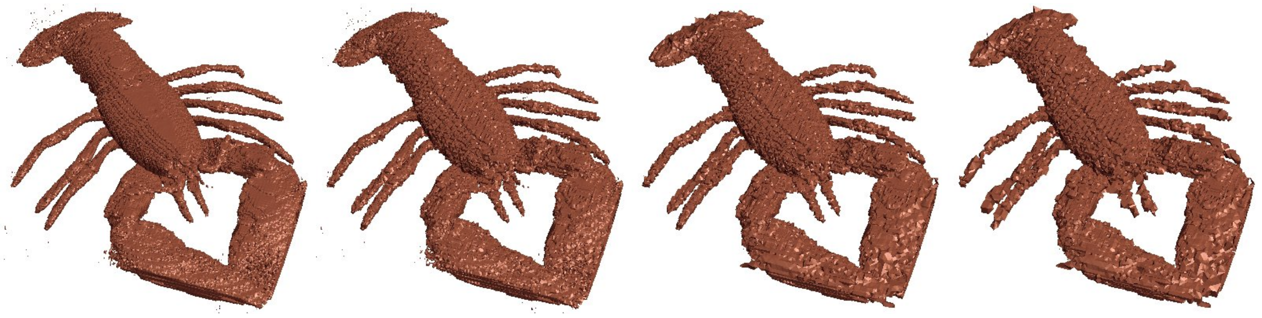
Figure 6: Isosurfaces of the lobster data set: original, topology preserving and controlled topology simplified versions (from left to right).