Adaptive Resolution Isosurface Construction in Three and Four Dimensions

Ramakrishnan Kazhiyur-Mannar Rephael Wenger Roger Crawfis Tamal K. Dey

Dept. of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210
E-mail:{ramakris,wenger,crawfis,tamaldey}@cis.ohio-state.edu

ABSTRACT

Marching Cubes and similar isosurface construction algorithms produce isosurfaces with tremendous numbers of triangles. Often, the isosurface need not be constructed at full resolution in all regions of the dataset. We present an algorithm for constructing an isosurface with varying levels of resolution from volumetric data given by a regular 3D or 4D grid.

Given a regular grid of scalar values and an isovalue, a grid vertex is labeled '+' if its scalar value is greater than the isovalue and '-', otherwise. A grid is monotonic in the direction of a given coordinate axis if all the '+' vertices precede the '-' vertices in that direction or vice versa. Our algorithm partitions the grid into rectangular regions of various sizes based on the monotonicity of these sign patterns. We show that by using monotonicity we can preserve isosurface topology. More significantly, using monotonicity "bands" we can control topological simplication, ignoring topological noise while retaining topological features.

The Marching Cubes algorithm applied to adjacent rectangular regions of different sizes may create cracks in the isosurface between the regions. We avoid this cracking by contracting some edges of the smaller regions, repositioning hanging vertices and faces so that adjacent regions intersect properly. The Marching Cubes algorithm applied to this mesh will create a crack-free isosurface. By adding a preprocessing splitting step, we can also guarantee that our edge contraction will not change the isosurface topology.

We present applications of our algorithm to 3D and 4D time-varying data and to 3D interval volumes.

Keywords: Volume visualization, isosurface, multiresolution, decimation

1 INTRODUCTION

Given a continuous scalar field, i.e., a scalar function of $\mathbb{R}^d$, an *isosurface* is a set of points with identical scalar values. Lorensen and Cline [11] gave a simple, efficient algorithm, called the Marching Cubes algorithm, for constructing a polyhedral approximation of an isosurface from a regular grid sampling of a scalar function in $\mathbb{R}^3$. The regular grid divides a volume into cubes whose vertices are the grid vertices. The Marching Cubes algorithm reconstructs the isosurface within each cube and then pastes together the resulting surface patches. Various modifications were proposed to avoid cracking problems in the original algorithm [12, 14].

Instead of cubes, the regular grid can be partitioned into tetrahedra and the isosurface can be reconstructed within each tetrahedron [10, 23]. Isosurface construction within each tetrahedron is simpler but the initial partition into tetrahedra causes the creation of about twice as many isosurface triangles [10]. Weigle and Banks [21] generalized the tetrahedral based algorithm to a simplex based algorithm in higher dimensions. Subsequently, Bhaniramka, Wenger and Crawfis [3] gave a hypercube based Marching Cubes algorithm in higher dimensions. The higher dimensional algorithms reconstruct $(d-1)$-dimensional surface patches within simplices or hypercubes and the union of these surface patches forms the isosurface.

The three dimensional Marching Cubes algorithm and its higher dimensional analogs create a large number of simplices which form the polyhedral isosurface. Many of these simplices represent relatively flat portions of the isosurface. We would like to replace these flat regions by a fewer number of larger simplices.

One approach to reducing isosurface complexity in $\mathbb{R}^3$ is to construct the isosurface and then apply a mesh decimation algorithm. There are numerous such algorithms with various properties[5], including preservation of isosurface topology, guaranteed error bounds, introduction of new vertices and preservation of feature edges.

An alternative method of reducing isosurface complexity is to generate flat portions of the isosurface from larger mesh elements, essentially ignoring some of the grid vertices in those regions. Techniques using this adaptive resolution approach in $\mathbb{R}^3$ are divided into those based on a regular grid of cube or hexahedral elements[13, 18], and those based on a tetrahedral grid[9, 22]. All these adaptive resolution approaches can guarantee that topology of the low resolution isosurface is the same as the topology of the full resolution isosurface.

This paper contains two contributions to the problem of constructing isosurfaces of varying resolution. First, we give a new method based on '+' and '$-$' sign patterns for determining the isosurface resolution in a given region. A vertex is labeled '+' if its scalar value is greater than the isovalue and '$-$' otherwise. A regular grid is *monotonic* in the direction of a given coordinate axis if all the '+' vertices precede the '-' vertices in that direction or vice versa. If a grid is monotonic along some axis, then the "bending" of the isosurface within that grid is extremely limited. Thus subgrids which are monotonic along one or more axes are good candidates for being replaced by larger mesh elements.

In Section 3, we formally define a monotonicity property based on the monotonicity of the full grid and of its faces. A grid with this monotonicity property defines an isosurface homeomorphic to a disk. Our algorithm identifies subgrids which have this monotonicity property and represents them by a single large grid cube. Because the isosurface in these subgrids are disks, replacing them by a single grid cube does not change the topology of the generated isosurface.

While monotonicity conditions work well on smooth, synthetic data, they have problems when applied to real data whose isosurfaces contain geometric and topological noise. To handle noise, we replace monotonicity conditions based on a single isovalue with monotonicity conditions based on a band of isovalues. By varying the size of this band, we can increase or decrease the resolution of

the resulting isosurface. We can also increase the size of this band to produce topological simplification in the isosurface, eliminating topological features which correspond to small changes in the scalar field.

Our second contribution is a method of avoiding isosurface cracks in meshes consisting of axis-parallel rectangular regions (hexahedra in 3D.) Isosurface patches which are generated in two adjacent regions of different sizes may have cracks along their common boundary [13, 18]. The reason is that some vertices in the smaller region may not correspond to any vertex in the larger one. To avoid these cracks, we identify "hanging" vertices in smaller regions which are not vertices on adjacent larger regions. We move these vertices to vertices on the larger region, essentially contracting some of the mesh edges. By contracting the edges consistently, we can guarantee that adjacent elements in the new mesh intersect properly on their faces.

The rectangular regions are hexahedra in three dimensions and topological hypercubes in four dimensions. Contracting edges will change the regions into other polyhedra such as pyramids or prisms. Such polyhedra can be viewed as degenerate hexahedra or hypercubes with degenerate edges of zero length which are not intersected by the isosurface. Thus, the isosurface lookup table for hexahedra and hypercubes can still be used to construct the isosurface in all the mesh elements.

Both our monotonocity conditions for determining isosurface resolution and the hexahedral edge contraction for avoiding cracks generalize directly to regular grids in any dimension. In fact, this ease of generalization was a major motivating factor in the choice and design of our algorithm. However, our methods only seem practical in four and possible five dimensions. We present some examples of our algorithm applied to four dimensional data.

The problem of determining when and where to simplify a polyhedral surface is a difficult one and we neither believe nor claim that the monotonicity should be the sole criterion for determining isosurface resolution. In particular, geometric conditions such as distance between low and high resolution surfaces and curvature of the high resolution surface should also play a role in choosing mesh element size. Similarly, while using monotonicity bands presents a novel approach to topological simplification and topological noise, traditional techniques such as Gaussian smoothing of the initial data, removal of small components, and removal of small features by erosion and dilation also are important tools. We offer monotonicity as one addition to a suite of tools in isosurface simplification.

In the next section, we describe some of the previous work on adaptive resolution isosurface construction. In Section 3, we define and discuss the monotonicity property. In Section 4, we discuss topological noise reduction and generalize isovalue monotonicity to monotonicity bands. In Section 5, we describe our hexahedral edge contraction for avoiding the cracking problem. In Section 6, we describe our algorithm for constructing the isosurface at adaptive levels of resolution. Finally, we present some details and results of our implementation and our plans for future work.

2 Previous Work

We refer the reader to [5] for a discussion and comparison of the major surface and volume decimation techniques, which are far beyond the scope of this paper. We note that while most decimation techniques preserve topology and some simplify topology, very few do both. Those that simplify topology do not usually produce manifolds as output.

While we know of no papers explicitly discussing decimation of surfaces lying in 4D, most of the algorithms simplifying 3D volumes or surfaces in 3D easily generalize. However, it is significantly more difficult to ensure that the topology of an isosurface in 4D does not change. Techniques by Dey et. al. [7] can be used to guarantee no change. For surfaces in dimensions greater than four, we know of no way of guaranteeing topology preservation.

Previous adaptive resolution techniques for isosurface generation divide into two categories, those based on a regular hexahedral grid and those based on a tetrahedral grid.

Müller and Stark [13] and Shekhar et. al. [18] generate adaptive resolution isosurfaces by merging unit cubes into rectangular regions of various sizes and generating isosurface patches with these larger regions. Both require that the isosurface in the larger regions be homeomorphic to a disk. Müller and Stark merges cubes if the isosurface generated after the merge intersects the same edges of the original grid as before the merge. Shekhar et. al. merge cubes if the isosurface is within some tolerance of a plane fitted to the isosurface. Both algorithms patch "cracks" between regions of different sizes by snapping vertices created in the smaller regions onto the isosurface from the larger ones. Both algorithms preserve isosurface topology and do not permit simplification of that topology. Neither the topology preservation nor the cracking resolution of either algorithm generalizes to higher dimensions.

The low resolution isosurface generated by Müller and Stark intersects the same edges of the original grid as the full resolution isosurface. This property is quite restrictive unless the isosurface is originally very smooth. In fact, Müller and Stark apply a smoothing operator to some of their data before applying their algorithm.

Both [13] and [18] preserve topology by constructing isocontours along the boundary of the cubes. If the isocontour is a single closed curve, then the isosurface bounded by the curve is a disk. The analogue in higher dimensions would be determining if the isosurface on the boundary forms a $(d-2)$-sphere, a difficult, if not impossible, task in higher dimensions.

Zhou et. al. [22] presented an alternate adaptive resolution approach based on meshes of tetrahedra. They detected and marked mesh vertices whose deletion would change the isosurface topology. Starting with six large tetrahedra, they repeatedly apply three cannonical subdivisions until all the marked vertices are part of the mesh. Cracking is avoided by splitting tetrahedra so that the mesh is a simplicial complex. Gerstner and Pajarola [9] used preprocessing and table lookup to identify the significant mesh vertices with greater accuracy, allowing the deletion of some vertices preserved in [22]. Both methods generalize to simplicial meshes in higher dimensions, although the number of subdivision types and table cases increases with the dimension.

The drawback to simplicial meshes is the necessity to split all hypercubes into simplices. The number of simplices required grows with the dimension. Simplicial decompositions also introduce their own sampling artifacts into the isosurface [4].

Instead of focusing on a single isosurface, one could reduce the mesh representing the scalar field while preserving scalar field topology. Bajaj and Schikore [1] gave an algorithm to do so in $\mathbb{R}^2$ based on computing critical points and integral curves in the scalar field. Edelsbrunner et. al. [8] showed how to compute the Morse complex on triangular meshes in $\mathbb{R}^2$. The Morse complex represents the scalar field topology and can be used to guide mesh reduction without affecting that topology. Computing the Morse complex on a tetrahedral mesh in $\mathbb{R}^3$ is more difficult and listed as an open problem in [8]. We know of no one addressing the issue of computing the Morse complex on a simplicial complex in $\mathbb{R}^4$ or whether that is even a tractable problem.

We refer readers to *The Handbook of Grid Generation*[19] for a reference to the vast literature on mesh generation and refinement, particularly Chapter 21-1 by R. Schneiders on quadrilateral and hexahedral element meshes. Following Schneiders definition, a mesh is *conforming* if the intersection of adjacent mesh elements is a face of each element. (This is not to be confused with boundary conforming meshes where the mesh conforms to some given

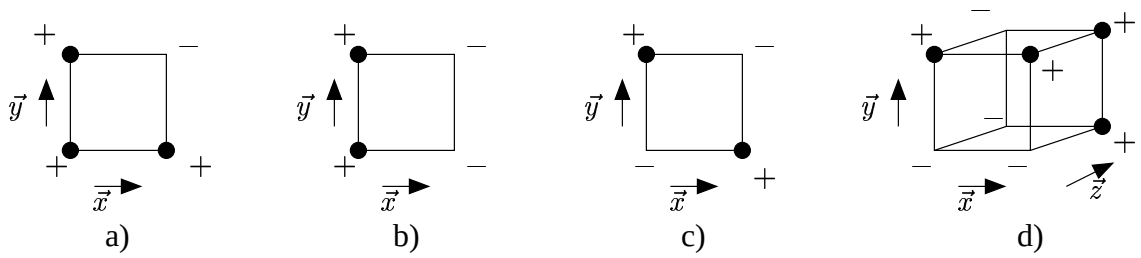

Figure 1: a) Monotonically decreasing in $\vec{x}$ and $\vec{y}$; b) Monotonically decreasing in $\vec{x}$ and decreasing and increasing in $\vec{y}$; c) Not monotonic along the x-axis or the y-axis; d) Monotonically increasing in $\vec{x}$ and $\vec{y}$, and not monotonic along the z-axis.

boundary.) Octree-based algorithms for adaptive refinement of hexahedral meshes are described in [17, 16, 15]. The problem is to subdivide an octtree to turn it into a conforming mesh. In [17], templates are used to subdivide octree elements so that they conform to their neighbors. There are difficulties with templates for certain configurations of octree elements. In [16], an entire surface is used to split octree elements in the region of desired refinement. The algorithms produce only convex hexahedral elements.

Weber et. al. in [20] use pyramids, prisms and hexahedra to fill the gap between regular hexahedral meshes of different resolutions. Careful case analysis is used to determine the various elements needed. The algorithm produces convex polyhedral mesh elements and does not subdivide the given meshes, only the region between meshes. On the other hand, it depends upon the regularity of the given meshes, and does not seem to handle the more general case of a general partition into hexahedra.

None of the papers mentioned above consider the problem of meshes in $\mathbb{R}^4$. It is not clear how hard generalizing them beyond $\mathbb{R}^3$ would be.

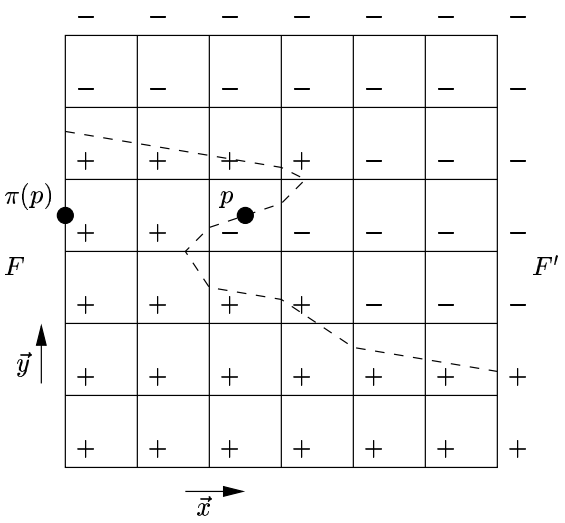

Figure 2: A 2-dimensional grid with the monotonicity property. The grid is monotonically decreasing in direction $\vec{x}$ and subgrids F and F' both have the monotonicity property. The grid is not monotonic along the y-axis. (Note the $+-+$ patterns in the third and fourth grid columns.)

3 MONTONICITY

Consider two parallel edges in a sign-labeled grid where one of the edges has label $(+,-)$, while the other has the reverse labeling $(-,+)$. If the edges lie on a single line L, then clearly an isosurface must bend to intersect L twice. However, even if the edges are not collinear, the reverse labeling of the edges force a connected isosurface to "bend" or "twist" so that both vertices labeled "+" are on the same side of the isosurface. We claim that if there are no such labeling reversals or if the labeling reversals are appropriately restricted, then the isosurface defined by the grid is homeomorphic to a disk.

A sign-labeled cube does not necessarily define a topologically unique isosurface passing through the cube. There may be two or more topologically distinct isosurfaces which are consistent with the given sign labeling. The isosurface S_G we choose is the one described by Bhaniramka et. al. in [3]. In three dimensions, this is topologically identical to the isosurface produced by the modified marching cubes algorithm[12] in almost all cases. The single exception is a cube with two opposite corners labeled '+' and all other vertices are labeled '-'. Actually, all our claims hold equally well for the isosurface defined by the modified marching cubes algorithm.

Bhaniramka et. al. construct an isosurface lookup table for each cube labeling by forming the convex hull of the '+' vertices and the midpoints of edges with '+' and '-' endpoints. The intersection of the boundary of this convex hull and the interior of the cube is the generic isosurface stored for this labeling. As in the original marching cubes algorithm, the isosurface S_G is constructed within each grid cube by looking up the generic isosurface for the given labeling and then interpolating the location of the isosurface vertex coordinates. (See [3] for more details.) The isosurface topology is entirely determined by the lookup table.

It is easier to define and illustrate monotonicity and the monotonicity property if we do so in arbitrary dimension. Let $\vec{x}$ be a unit vector parallel to a coordinate axis. A labeled d-dimensional regular grid is *monotonically decreasing* in direction $\vec{x}$ if no hypercube edge $(v, v+\vec{x})$ parallel to $\vec{x}$ has label $(-,+)$. (See Figure 1.) A labeled d-dimensional regular grid is *monotonically increasing* in direction $\vec{x}$ if no hypercube edge $(v, v+\vec{x})$ parallel to $\vec{x}$ has label $(+,-)$. Note that a grid can be both monotonically increasing and decreasing in a given direction (i.e., v has a '+' if and only if $v+\vec{x}$ does.) It can also be neither monotonically increasing nor decreasing in a given direction. A labeled d-dimensional regular grid is *monotonic* along a given axis, if it is monotonically decreasing in one of the two directions parallel to that axis (and hence monotonically increasing in the other direction.)

The *facets* of a regular d-dimensional grid G are the $2d$ regular $(d-1)$-dimensional subgrids bounding G. There are exactly two grid facets perpendicular to each axis. (In Figure 2, the grid facets perpendicular to the x-axis are labeled F and F'.) We say that a

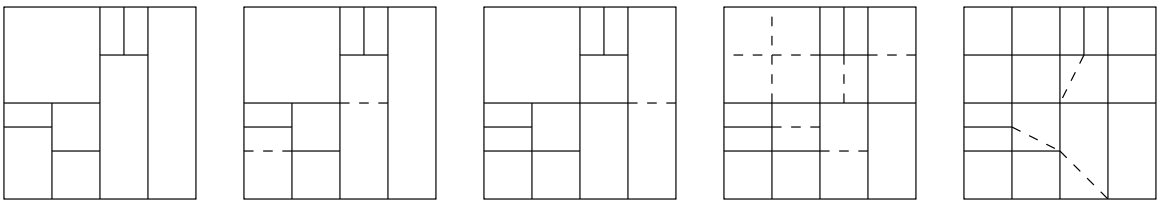

Figure 3: a) Partition into rectangles; b) Partition without partial overlaps; c) Balanced partition; d) Preprocessing splitting step; e) Mesh after edge contraction.

labeled grid G has the *monotonicity property* if either G has dimension zero or G is monotonic along a given axis and both facets of G perpendicular to that axis have the monotonicity property. (See Figure 2.) Thus a 1-dimensional grid has the monotonicity property if and only if it is monotonic. A 2-dimensional grid has the monotonicity property if and only if it is monotonic along a given axis and the two 1-dimensional grids perpendicular to that axis are monotonic. A 3-dimensional grid has the monotonicity property if and only if it is monotonic along a given axis and the two 2-dimensional grids perpendicular to that axis have the property. Note that if G is monotonic in all directions, then G has the monotonicity property.

We claim that if a three dimensional grid G has the monotonicity property, then the isosurface S_G is homeomorphic to a disk. More generally, if a d-dimensional grid G has the monotonicity property, then the isosurface S_G is homeomorphic to a $(d-1)$-dimensional ball. As an immediate corollary, if G is monotonic in all directions, then S_G is homeomorphic to a $(d-1)$-dimensional ball.

The general idea behind the proof for $\mathbb{R}^3$ is as follows. Assume a 3-dimensional grid G is monotonic in some direction $\vec{x}$ and both 2-dimensional subgrids in the faces F and F' perpendicular to $\vec{x}$ have the monotonicity property. Because G is monotonic in direction $\vec{x}$, the isosurface vertices can be perturbed so that π is a 1-1 projection of the isosurface into F. Perturbing the vertices does not affect the topology of G. The intersection of S_G and F and F' are curves on F and F', respectively. The isosurface is homeomorphic to the region on F between $S_G \cap F$ and $\pi(S_G \cap F')$.

By induction, $S_G \cap F$ is either the empty set or a single curve with endpoints on the boundary of F. The same holds for $S_G \cap F'$. Thus $S_G \cap F$ and $\pi(S_G \cap F')$ are non-intersecting curves with endpoints on the boundary of F. Since F is a topological disk, the region between $S_G \cap F$ and $\pi(S_G \cap F')$ is a topological disk. Thus the isosurface is homeomorphic to a disk.

A similar, although more intricate, argument can be made in higher dimensions. The detailed proof and its extension to higher dimensions are left for the full version of this paper.

4 Topology and Monotonicity

The monotonicity property described in the previous section preserves isosurface topology. While this is a desirable theoretical goal, it has significant drawbacks on real data which may have noise and/or small topological features. Noise may create small topological and geometric features on the isosurface causing subgrids containing such noise to fail the monotonicity property.

There are numerous methods for reducing the effects of noise on a data set, including smoothing the initial data set, removing small connected components and applying dilation and expansion operators to the region enclosed by the isosurface. We found it most useful to partition the data set into connected regions with values above or below the isovalue, and remove small connected components in that partition. Grid vertices in the connected components received the scalar value of their surrounding neighbors. Removing the small connected components in no way affects the remaining isosurface bounding the large components.

With the removal of small components, we were able to achieve significant levels of reduction in isosurface complexity even on noisy data sets. However, noise could still create topological and geometric features on a large isosurface causing subgrids containing such features to violate the monotonicity property. Small features on the isosurface would also cause such violations.

To further reduce isosurface complexity, we expanded monotonicity based on a single isovalue to *monotonicity bands* based on a minimum and maximum isovalue. A grid is monotonically increasing (decreasing) with respect to a given band if it is monotonically increasing (resp. decreasing) with respect to BOTH its minimum and maximum isovalue. A labeled grid G has the *monotonicity property* with respect to a given band if either G has dimension zero or G is monotonic along a given axis and both facets of G perpendicular to that axis have the monotonicity property with respect to that band. If a grid has the monotonicity property with respect to a given band, then the band can be replaced by an isosurface topologically equivalent to a disk, which separates the values above the maximum value with the values below the minimum value.

In Section 6, we describe how monotonicity bands around a given isovalue are used to determine the isosurface resolution. By varying the size of a monotonicity band, we can vary the resolution and complexity of the constructed isosurface. Note that while many decimation and simplification algorithms depend solely on the geometry and topology of the original isosurface, monotonicity bands depends both on isosurface geometry and on the surrounding scalar field. It is particularly well-suited to high frequency, low amplitude noise, which causes small changes in scalar field values.

5 Hexahedral Edge Contraction

Consider a regular d-dimensional grid partitioned into axis-parallel rectangular regions called *boxes*. Adjacent boxes may have different sizes and their intersection may be different from their faces. Our goal is to modify this partition into a mesh so that the intersection of adjacent mesh elements is a face of each. Applying the isosurface reconstruction algorithm to this mesh will produce a crack free surface.

We enforce two restrictions on our initial partition into boxes. First, we require that if any two boxes intersect, then the intersection is a face of at least one of the boxes. In other words, box faces cannot partially overlap each other. Our algorithm naturally enforces this condition by only creating boxes on certain boundaries but we could enforce it by a partitioning step which splits boxes violating this condition. (See Figure 3.)

The second condition is that if an edge of one box properly contains the edge of another, then the longer edge is exactly twice the corresponding dimension of the smaller one. This is a standard restriction in quadtree or octtree mesh generation algorithms and is enforced by splitting any large boxes violating this condition. Such splitting increases the size of the quadtree or octtrees by at most a constant factor[6, Chapter 14]. These two conditions guarantee that if one box face contains another, then each dimension of the larger box face is either equal to or twice the corresponding dimension of the smaller one.

A box vertex is called *hanging* if it lies on the boundary of some adjacent box but is not a vertex of that box. We wish to remove or relocate the hanging vertices.

We start with a preprocessing step which performs another splitting of boxes. For each box edge whose interior contains a hanging vertex, we split the box by a plane (or hyperplane) which passes through the hanging vertex and is perpendicular to the edge. After this split, none of the original vertices are hanging.

A hanging vertex lies in the interior of the face of some box. We identify the largest face whose interior contains the hanging vertex and move the hanging vertex to the face vertex with lowest coordinates. Note that since all faces are parallel to the coordinate axes, there is a unique face vertex whose every coordinate is less than or equal to the cooresponding coordinate of every other face vertex.

The preprocessing split serves two functions. First, since none of the original vertices are hanging, none of them will be moved. This guarantees that the isosurface topology will be preserved since all the original edges are preserved. Second, the vertices of a face containing a hanging vertex must be from the original set of vertices and so cannot be hanging. Thus a hanging vertex is never moved to a vertex which is itself hanging.

By always moving hanging vertices to the face vertex with lowest coordinates, we guarantee that the faces with hanging vertices will always contract onto faces of an adjacent box. Moreover, a box which loses its full dimensionality because of the repositioning of hanging vertices, will contract onto the face of an adjacent box. These conditions ensure that the intersection of adjacent face elements is a face of each in the resulting mesh. More formal proofs of these properties will appear in a full version of this paper.

Moving a hanging vertex may break the planarity of a box face. Topologically the face may remain the same, but its vertices may no longer lie in a single plane (or hyperplane.) Thus the mesh elements produced by our edge contraction are not convex polyhedra. This is not a problem for isosurface construction which really only uses the 1-skeleton (edges) of the mesh, although it does limit the utility of the algorithm for other remeshing applications.

6 ALGORITHM

The algorithm consists of six steps: topological cleanup by removal of small connected components, the initial partition into axis parallel boxes, splitting to balance box sizes, an additional splitting step, repositioning of hanging vertices and final isosurface extraction.

As described in Section 4, we form connected components of vertices with values above or below the isovalue and remove small connected components by redefining their isovalues based on neighboring isovalues. This step removes some of the topological noise associated with the isosurface.

For the initial partition into boxes, we could have used octtrees and their four dimensional equivalents but we did not want to restrict ourselves to cubes and hypercubes. Instead we use a bottom up approach, first partitioning the grid into subgrids of $2 \times 2 \times 2$ (or $2 \times 2 \times 2 \times 2$ in $\mathbb{R}^4$) voxels, and attempting to merge the voxels within each subgrid. If the $2 \times 2 \times 2$ subgrid satisfies our monotonicity condition, then we merge the entire subgrid into a single box. Otherwise we consider partitioning the $2 \times 2 \times 2$ subgrid into two $1 \times 2 \times 2$ or $2 \times 1 \times 2$ or $2 \times 2 \times 1$ subgrids. If none of those partitions succeeds, we consider the four $1 \times 1 \times 2$ or $1 \times 2 \times 1$ or $2 \times 1 \times 1$ subgrids. We repeat this for $4 \times 4 \times 4$ and $8 \times 8 \times 8$ and more generally $2^i \times 2^i \times 2^i$ subgrids. We restrict the maximum ratio between different dimensions of the box to four, although this parameter of the algorithm can be changed.

By only considering powers of two for subgrid dimensions, we guarantee that the intersection of adjacent boxes will always be a subset of the face of one of the boxes. Thus we avoid partial overlaps as required in Section 5.

Our monotonicity condition is either based on a single isovalue or on a band between a minimum and maximum values as described in Section 4. The smaller the band, the finer the resolution of the constructed isosurface. Note that the isosurface isovalue is independent of the minimum and maximum values defining the bands although it should lie between them. Of course, the use of monotonicity bands breaks the guarantee of preservation of isosurface topology.

To check the monotonicity condition for a subgrid, we could process all the edges of the subgrid and determine the monotonicity in each direction of the subgrid. However, we can also determine the monotonicity of a subgrid from the monotonicity of elements in a partition of that subgrid. For instance, if two subgrids are monotonically increasing in the x-direction, then the union of the two is monotonically increasing in the x-direction. If one subgrid is monotonically increasing and one decreasing in the x-direction, then the union is not monotonic in the x-direction. The same holds for partitions into more than two elements.

As we merge subgrids into boxes, we store the monotonicity in each direction for the boxes. When we attempt to merge boxes into larger boxes, we use the stored monotonicities to compute the monotonicities of the larger boxes. To compute monotonicity for bands, we must store the monotonicity with respect to both the minimum and maximum value of the band.

To check the monotonicity condition, we also need the monotonicity in each direction of the faces of each subgrid. However, storing that for each box would require storing the monotonicity in each of two directions for each of six faces in R^3. This is impractical for large data sets. Instead, we simply compute the monotonicity from the original grid edges.

The running time is dependent upon the time to check the monotonicity condition. If N is the total number of grid vertices, then there are $\Theta(N)$ grid edges, and there are $\Theta(\log(N))$ potential boxes which can contain a given edge (assuming that the ratio between the dimensions of a box can never be greater than four.) Thus the algorithm runs in worst case $\Theta(\log(N))$ time. This assumes that one considers all possible box dimensions of the form 2^i. In practice, we restricted the maximum box dimension to 8 voxels which compresses 512 voxels into a single box in $\mathbb{R}^3$.

The monotonicity is only one condition of many which could be used to determine box size. Geometric measures should also be included such as distance to the full resolution isosurface and curvature of the full resolution isosurface. Balmelli et. al. in [2] suggest counting total number of intersections between the isosurface and the grid edges. Since our contribution is the use of monotonicity, we do not incorporate these measures into our algorithm or the experimental results.

Splitting to balance box sizes, the additional splitting step and the repositioning of hanging vertices are discussed in Section 5. They run in time proportional to the total number of boxes which is $O(N)$.

For the isosurface extraction, each vertex is labeled "+" or "-" depending upon whether it's value is above or below the given isovalue. Each mesh element is a hexahedron (or hypercube in $\mathbb{R}^4$) with possibly some zero length contracted edges. The labeling of

Type	Dataset	IsoValue	Size at Full Resoln	Size with Decimation No Band	Size with Decimation and Band (with Width)	Small Component Size Chosen	Bound on Block Size	Average Time
3D	VisWoman Head	1100	3,256,272	1,597,152	307,354(100)	1000	4	515 s
3D	VisWoman Knee	1100	1,372,870	940,426	154,302(100)	1000	4	311 s
4D	Vortex	6	9,282,485	2,411,548	-	-	-	863 s
4D	Jetstream	0.0001	59,862,311	12,108,755	-	-	4	2138 s

Figure 4: Table of results

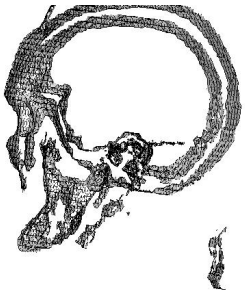

Figure 5: Cutaway of adaptive resolution isosurface representing skull. The internal structures in the skull are retained. Notice the high resolution in the inner ear area.

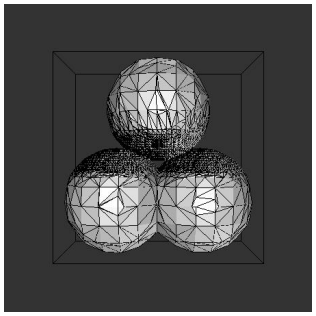

Figure 6: Three balls with full resolution in central region.

the mesh element vertices gives a labeling of the original hexahedron where a hanging vertex takes on the sign of the vertex to which it was moved. Using the Marching Cubes lookup table (or its 4D variant) we extract the isosurface patch for the mesh element. Note that both vertices of a contracted edge always have the same sign, so there is no problem of the isosurface intersecting the contracted edge.

7 RESULTS

We present results of our adaptive resolution algorithm on 3D and 4D data using various band widths. The performance measurements were done on a 1.5 GHz Pentium 4 system with 1 GB memory. All the time measurements shown are wall clock times. We used the Head and Knee portions of the Visible Woman dataset. For 4D data, we used 33 time steps of the Vortex dataset and the Jetstream Dataset. Isosurface size was measured in number of triangles for isosurfaces in 3D and number of tetrahedra for isosurfaces in 4D.

The Visible Woman dataset is made of 512 X 512 slices and each dataset we used had 257 such slices. This dataset has intreresting topology as well as noise especially near the bone region (isovalue approx. 1100.) The isosurface has internal structures that are not visible from the outside, such as parts of inner ear. These internal parts are also preserved by our algorithm, as is shown in Figure 5. Adaptive resolution with band width 100 reduced the number of isosurface triangles to approximately one-tenth of the number of triangles in the original full-resolution isosurface.

The Jet Stream and Vortex datasets are series of 3D scalar fields, which we represented as a 4D dataset. Each time step in the Jet dataset has dimensions 104 X 128 X 128, while Vortex is of size 128X128X128. We took the first 33 time steps of each of these datasets. Adaptive resolution reduced the number of tetrahedra in the Vortex data set to approximately one-fourth of the number of triangles in the original full-resolution isosurface. Adaptive resolution reduced the number of tetrahedra in the Jet Stream data set to approximately one-fifth of the number of original tetrahedra. No bands were used on either data set, meaning that isosurface topology was preserved in each.

For the Visible Woman MRI datasets, topological noise reduction was done by removing small components of less than 1000 grid vertices for the given isovalue. This process, however, does not remove the noise near the real isosurface, which is handled by our bands technique.

For the Visible Woman and Jet Stream data sets, we imposed a geometry criterion on the decimation by restricting the maximum size of the decimated block. Though this reduces the decimation achievable, in practice, the effect is not too drastic. The advantage of having a maximum bound is that it gives us regions of uniform block sizes. The surface patches in each such region have similar level of resolution, which is visually more pleasing. An additional advantage of having such uniform regions is that it reduces the number of hexahedral edge contractions that is required for fixing cracks.

With our algorithm, it is possible to retain high resolution in some region of interest, while allowing decimation in the rest. The resolution automatically reduces gracefully from the high resolution region. This effect could be noticed in the central band in Figure 6.

As described in [3], the interval volume between two isosurfaces in 3D can be constructed by lifting the 3D data into two layers in 4D. We lifted one frame of the Jet Stream data into 17 layers in 4D corresponding to 17 isovalues, and constructed an adaptive resolution isosurface in 4D representing the interval volume between the isosurfaces in 3D. We sliced the 4D isosurface to give an animated view of the different 3D isosurfaces. The adaptive resolution isosurface in 4D had 1,447,703 tetrahedra which was an improvement by a factor of six over 8,198,190 tetrahedra in the full resolution isosurface.

One advantage with the decimation algorithm described is that it partitions the dataset based on local conditions. Hence, in cases where the dataset is simply too big for the resources, we could consider portions of the dataset at a time and partition each. This is

especially useful in 4D datasets, which typically are of Gigabyte size.

Our implementation used very simple, but inefficient data structures, which contributed to an undesirably long running time. Significant amounts of time were spent in determining monotonicity and in detecting hanging vertices. We believe that more sophisticated data structures, data partitioning to avoid thrashing and preprocessing of empty voxels and blocks could substantially improve the running time.

8 ACKNOWLEDGEMENTS

We thank Zbigniew Fiedorowicz for helpful consultations on topology. We thank an anonymous referee who pointed out errors in our monotonicity conditions in a previous version of this paper. We thank Dr. Kwan Liu Ma for supplying the Vortex data set. The Jet Stream data set was downloaded from the University of Utah AVTC Center.

REFERENCES

[1] C. Bajaj and D. Schikore. Topology preserving data simplification with error bounds. *Computers & Graphics*, 22(1):3–12, 1998.

[2] L. Balmelli, C. Morris, G. Taubin, and F. Bernardini. Volume warping for adaptive isosurface extraction. In *Proceedings IEEE Visualization '02*, pages 467–474. IEEE Computer Society Press, 2002.

[3] Praveen Bhaniramka, Rephael Wenger, and Roger Crawfis. Isosurfacing in higher dimensions. In *Proceedings of Visualization 2000*, pages 267–273, 2000.

[4] Hamish Carr, Torsten Möller, and Jack Snoeyink. Simplicial subdivisions and sampling artifacts. In *Proceedings IEEE Visualization 2001*, pages 99–106, 2001.

[5] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Comput. & Graphics*, 22(1):37–54, 1998.

[6] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[7] Tamal Dey, Herbert Edelsbrunner, Sumanta Guha, and Dmitry Nekhayev. Topology preserving edge contraction. geometric combinatorics. *Publ. Inst. Math. (Beograd) (N.S.)*, 66:23–45, 1999.

[8] Herbert Edelsbrunner, John Harer, and Alfra Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 70–79, 2001.

[9] Thomas Gerstner and Renato Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proceedings of IEEE Visualization '00*, pages 259–266, 2000.

[10] Andre Gueziec and Robert Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Trans. Visualizat. Comput. Graph.*, 1(4):328–342, 1995.

[11] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Comput. Graph.*, 21(4):163–170, 1987.

[12] Claudio Montani, Riccardo Scateni, and Roberto Scopigno. A modified look–up table for implicit disambiguation of Marching Cubes. *Visual Comput.*, 10:353–355, 1994.

[13] Heinrich Müller and Michael Stark. Adaptive generation of surfaces in volume data. *The Visual Computer*, 9(4):182–199, January 1993.

[14] G.M. Nielson and B. Hamann. The Asymptotic Decider: Removing the ambiguity in Marching Cubes. In *Visualization '91*, pages 83–91, 1991.

[15] R. Schneiders. Algorithms for quadrilateral and hexahedral mesh generation. In *Proceedings of the VKI Lecture Series on Computational Fluid Cynamics*, 2000.

[16] R. Schneiders. Octree-based hexahedral mesh generation. *Internat. J. Comput. Geom. Appl.*, 10(4):383–398, 2000.

[17] R. Schneiders, R. Schindler, and F. Weiler. Octree-based generation of hexahedral element meshes. In *Proc. 5th International Meshing Roundtable*, pages 205–215, PO Box 5800, MS 0441, Albuquerque, NM, 87185-0441, 1996. Sandia National Laboratories. Also Sand. Report 96-2301.

[18] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Fredrick Cornhill. Octree-based decimation of marching cubes surfaces. In Roni Yagel and Gregory M. Nielson, editors, *Proceedings of the Conference on Visualization*, pages 335–344, Los Alamitos, 1996. IEEE.

[19] J.F. Thompson, B. Soni, and N. Weatherhill, editors. *Handbook of Grid Generation*. CRC Press, 1999.

[20] G.H. Weber, O. Kreylos, T.J. Ligocki, J.M. Shalf, H. Hagen, B. Hamann, and K.I. Joy. Extraction of crack-free isosurfaces from adaptive mesh refinement data. In G. Farin, B. Hamann, and H. Hagen, editors, *Hierarchical and Geometrical Methods in Scientific Visualization*, pages 19–40. Springer-Verlag, Heidelberg, Germany, 2003.

[21] C. Weigle and D. Banks. Complex-valued contour meshing. In *Proceedings of Visualization '96*. IEEE Computer Society Press, 1996.

[22] 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.

[23] 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.

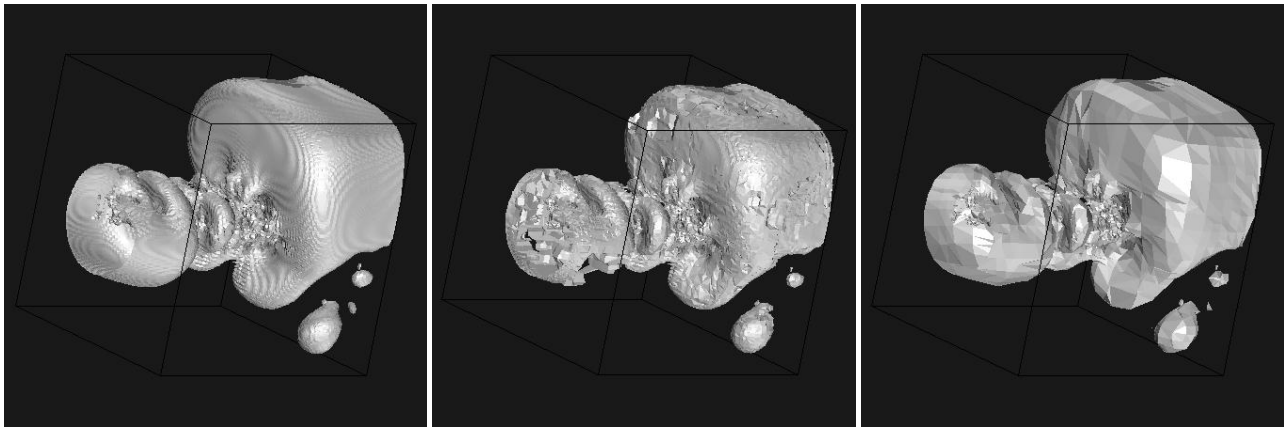

Figure 7: Jet Stream Slice: a) Full resolution isosurface generated from 3D data; b) Slice of adaptive resolution isosurface generated from 4D data; c) Adaptive resolution isosurface generated from 3D data

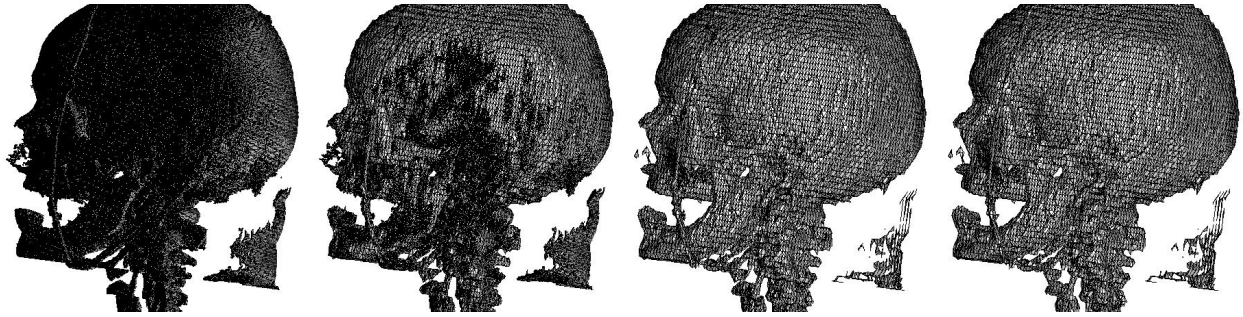

Figure 8: Visible Woman Head: a) Full resolution; b) Adaptive resolution with no band; c) Adaptive resolution with band width 100; d) Adaptive resolution with band width 200.

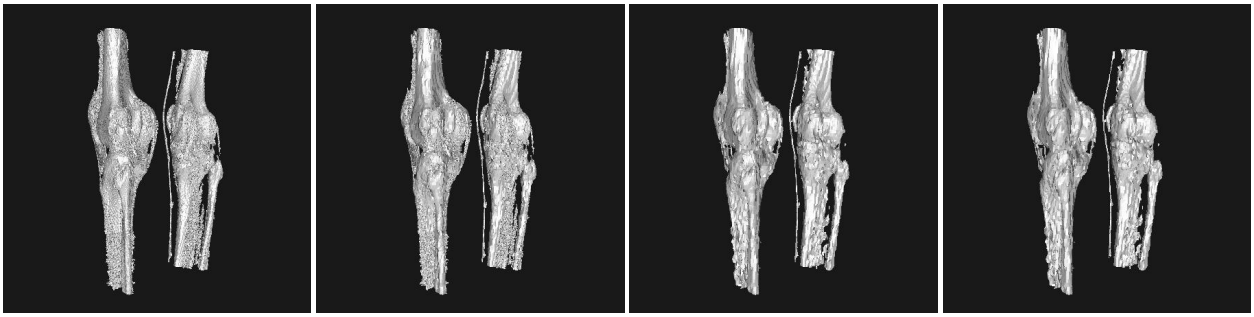

Figure 9: Visible Woman Knee: a) Full resolution; b) Adaptive resolution with no band; c) Adaptive resolution with band width 100; d) Adaptive resolution with band width 200.