Digital Morse Theory for scalar volume data

Jim Cox (cox@sci.brooklyn.cuny.edu)*

D. B. Karron (karron@casi.net)†

Nazma Ferdous (nferdous@sci.brooklyn.cuny.edu)‡

July 16, 2002

City University of New York

Computer Aided Surgery, Inc.

Abstract

We present a new method for preprocessing and organizing discrete scalar volume data of any dimension on external storage. We describe our implementation of a visual navigation system using our method. The techniques have important applications for out-of-core visualization of volume data sets and image understanding. The applications include extracting isosurfaces in a manner that helps reduce both I/O and disk seek time, a priori topologically correct isosurface simplification (prior to extraction), and producing a visual atlas of all topologically distinct objects in the data set. The preprocessing algorithm computes regions of space that we call topological zone components, so that any isosurface component (contour) is completely contained in a zone component and all contours contained in a zone component are topologically equivalent. The algorithm also constructs a criticality tree (independently developed [21],[20], [17],[10]) that is related to the contour tree of [2], [3], [32], [31]. However, unlike the contour tree, the zones and the criticality tree hierarchically organize the data set. We demonstrate that the techniques work on both irregularly and regularly gridded data, and can be extended to data sets with nonunique values, by the mathematical analysis we call Digital Morse Theory (DMT), so that perturbation of the data set is not required. We present the results of our initial experiments with three dimensional volume data (CT) and describe future extensions of our DMT organizing technology.

*Supported in part by ONR grant N00014-96-1-1057

†Supported by DARPA Contract DAAH01-98-C-R195 under DSO Dennis Healy and NIST ATP Grant 70NANB1H3050 Jayne Orthwein and B.J. Lide

‡Supported in part by ONR grant N00014-96-1-1057

1 Introduction

Many applications produce data in the form of a scalar function defined at discrete points in space (called volume data). Examples include X-ray crystallography, Computed Tomography (CT), Magnetic Resonance Imaging (MRI), and data from continuous field simulations such as computational fluid dynamics. The data is visualized using either volume rendering ([22],[1],[23],[25],[30]) or isosurface extraction (See [24],[12],[8], [19],[9]). In volume rendering the data is regarded as consisting of semi-transparent material and volume primitives are directly projected onto the screen using ray casting.

Isosurface based methods are primarily used for filtered volume data. In isosurface based methods the data is segmented by thresholding, that is, identifying objects defined by the sets of points in space where a suitable interpolating function f for the filtered data is greater than or equal to a real value τ, called the isovalue or threshold. The topological boundary of these objects will generally correspond to the level sets of f, that is, the points p where $f(p) = \tau$. The data is either regularly gridded, that is, represents a function δ defined at points of $\mathbf{Z^n}$ on the vertices of a hypercubic decomposition of space, or it is irregularly gridded and δ is typically defined on the vertices of a simplicial decomposition of $\mathbf{R^n}$. The simplices or (hyper)cubes are called cells. The isosurface extraction problem is to construct the level set of f, called the isosurface, for specific isovalue τ. Individual connected components of this set are typically called the contours, even when they are higher dimensional surfaces.

Many researchers have developed methods of organizing the data so that the "active cells", those cells that a given contour intersects, can be quickly accessed from disk for out-of-core rendering. The three main methods include geometric organization ([33], [34], [16], value based organization [11], [29], [37], [7], [4, 6, 5], and recently the topology based organization using augmented contour trees [2], [3], [32], [31]. The I/O optimal interval trees of [4, 6, 5] seem to produce the least disk I/O and seek time, and input the minimal number of blocks in active cell acquisition. Our organization is topology based.

Topological organization of the data uses Morse Theory. Morse theory studies the changes in the topology of the level sets of a Morse function f as the parameter τ is decreased. These changes occur at the values of critical points of f, where the derivative vanishes. A Morse function has the property that the critical points are isolated and the Hessian is nonsingular at each of these points. Typically the data readings are given on a tetrahedral mesh and are perturbed to insure that no two values are identical, and a simple (e.g. linear) interpolant f is selected, to insure that f is Morse. The contour tree tracks the topological changes of individual contours as the parameter τ is varied. The augmented contour trees of [32] are used to produce a seed cell set and active cells for a given contour are acquired by local propagation from the seed cells.

We define a variant of the contour tree and use it to organize the data into topological zones and topological zone components. We demonstrate that the zones have important properties not satisfied by the augmented contour tree, making them useful in a variety of visualization problems. For example,

the organization aids in active cell acquisition, and is superior to the seed cell method.

Prior work on contour trees requires that regularly gridded data be regridded onto tetrahedral cells. This has two disadvantages. It requires increasing the data complexity (e.g. dividing each cubic cell into 5 tetrahedra) and further requires an explicit representation of the tetrahedral cells and their adjacency, which increases storage requirements and the complexity of dealing with the data. Moreover, the data readings are then perturbed to insure that no two readings are identical. In this way one insures that the data can be extended to a Morse function. Our mathematical analysis, which we call Digital Morse Theory, demonstrates that this treatment of regularly gridded data is not needed. The contours produced by the most popular isosurface extraction methods (see for example [27]) are sufficiently well behaved so that the topological changes, as the isovalue is varied, are well defined and easily identifiable, without recourse to Morse theory. We feel that this is important because sampled density data associated with a physical phenomenon may contain regions of constant density, and regularly gridded data does not require an explicit storage of cell adjacency information. We give a combinatorial characterization of the topology changing criticalities and state a related open problem.

The paper is organized as follows: In section 2 we briefly discuss typical methods for isosurface extraction, so that we may identify the properties of the surfaces produced by these methods. We also give a very simple algorithm that computes contours in all dimensions for regularly gridded data. In section 3 we give our mathematical analysis. We formally define the properties satisfied by the above isosurface extraction methods, and demonstrate that these properties uniquely determine both the topology and the topological changes of the contours. Alternatively, one can use the standard tertrahedral regridding of regularly gridded data and perturbation of identical readings to insure that the function is Morse. In either case, the zone organization is an important contribution, and the rest of the paper may be understood independently of section 3, with the usual assumptions of a Morse function.

In section 4 give the formal definition of the topological zones and prove certain of their properties. In section 5 we show how to compute the data cells that intersect a topological zone component and disuss our implementation of the algorithm. In section 6 we describe our initial implementation of a visual navigation system for volume data, using the zone organization. We discuss our experimental results using the navigation tool, with particular attention to the I/O efficiency of isosurface extraction using our technique. We also discuss how we shall scale up the tool to handle very large data sets. In section 7 we discuss future applications of our technique, including topologically correct volume data simplification, and efficient management of level of detail. Finally in the appendix we give the proofs to our mathematical assertions of section 3.

2 Isosurface extraction

The isosurface extraction problem is often stated as follows. A scalar volume data set consists of tuples (x, g(x)), where x is a point in 3 (or higher) dimensional space, and g is a scalar function defined over a discrete set of these points. Given an isovalue q, the isosurface extraction problem is to compute the (hyper)surface consisting of the points $\{x : f(x) = q\}$, where f is a function (interpolant) which extends g to all of space. A more mathematically precise formulation is to say that we are computing the topological boundary of $\{x : f(x) \geq q\}$, as the set of points of constant value q may in fact include a 3D volume in the degenerative case, and we may have multiple surface components, which, by an abuse of terminology, are called contours in the literature. A further requirement is that the contours are oriented manifolds.

The data points are usually organized into cells, with the data readings given at the vertices of the cells. Regularly gridded data is given on the vertices of cubic cells and irregularly gridded data is typically given on the vertices of a tetrahedral decomposition of space. The algorithms discussed below construct the isosurface locally within each cell.

2.1 Marching Cubes

A classic algorithm, that is applied when the volume data is given on cubic cells, was proposed by Lorensen and Cline [24]. It is called marching cubes and it constructs a polygonal mesh to represent a 3D surface. In order to generate an isosurface, corresponding to a user specified isovalue, the algorithm visits all cubic cells created from eight adjacent pixels, four each from two adjacent slices, and for each cell determines whether any surface intersects the cell, and then moves to the adjacent cell. To determine surface intersection within a cube, the algorithm examines all its vertices and assigns a High, if the data value is higher than or equal to the isovalue, (i.e. these vertices are considered to be inside the object) or Low, if the data value falls below the isovalue (i.e. these vertices are outside the object).

A cube edge is intersected by the surface if and only if one of its two vertices is inside, and the other is outside the region of space bounded the surface (the isosurfaces are assumed to be oriented manifolds). The location of the intersection point can be approximated by linear interpolation. The topology of the surface within a cube can be determined by the pattern of its vertex values (High or Low). There are 256 ways to color 8 vertices with 2 colors, however, by taking into account rotational and complementary symmetry there are only 15 topologically distinct patterns. A look-up table is built that includes all 15 patterns and a corresponding surface approximation for each. The surface approximation uses triangles that connect the intersection points. All the distinct patterns of surface approximation are enumerated in figure 1. Each High is illustrated by a dark dot at a vertex.

It was later pointed out by Nielson [26], that the original *marching cubes* produced tiling errors between cells, and he proposed an asymptotic decider to

remedy this. The problem occurs in what Nielson terms ambiguous faces, those faces with 4 separate isosurface intersection points (which we term 4 hit faces). Figure 2 shows one such case where triangles produced by *marching cubes* do not produce a continuous surface.

For instance, if a cubic cell with configuration case 6 shares a face with a voxel having a configuration $\overline{3}$, (which is the complement of case 3), triangles produced by the *marching cubes* algorithm will create a hole in the resultant isosurface. In order to correct this problem, a different triangulation can be used. Figure 3 depicts two possible triangulations that will result in a topologically consistent isosurface. The choice of triangulation can be made using the proposed asymptotic decider which is based on bilinear interpolation of the value of an interior point in the face.

The goal is obviously to divide each cubic cell by one or several surface patches (up to 4 for 3 dimensional cells), each homeomorphic to a unit disk, so that the induced cell regions each contain only connected sets of High and Low vertices (above or below the isovalue resp.) Further each cell face contains 0, 1, or 2 curve segments that divide the connected High and Low vertices in the face. The 5 possible patterns are shown in figure 9. As we shall see this goal can be generalized to higher dimensional cubic cells.

2.2 Exploiting tetrahedral cells

Irregularly gridded data is typically organized into tetrahedral cells. The faces of the tetrahedral cells can only be classified (or colored) in three distinct ways, where the connectivity among the vertices can be resolved without any ambiguity. A look-up table is thus not needed to find the approximated surface. The three possible cases, described in figure 4, are the following:

- only v1 is outside the object, the rest are inside.
- v1 and v2 are outside the object, and v3 and v4 are inside.
- only v4 is inside the object, the rest are outside.

Decomposing cubic cells into five tetrahedral cells [12] can lead to an efficient surface extraction, primarily because of the fact that this method does not suffer from the well known inconsistency discussed above. However, Zhou et al [36] showed that tetrahedral decomposition and linear approximation along the introduced diagonals change the original function and may lead to incorrect, though consistent topology. They proposed trilinear approximation across the diagonal of the cells as a solution to this problem, rather than applying linear approximation along all edges. This method has a disadvantage of increasing five-fold the number of cells that have to be processed individually.

The surface construction for tetrahedral cells can be generalized to higher dimensions, where each cell is an n-dimensional simplex. In each case the vertices above and below the isovalue in a cell can be separated by a single hypersurface patch that is homeomorphic to an $n-1$-dimensional closed ball. For cubic cells

in n dimensions we should similarly require that surface patches within the cell are each homeomorphic to the $n-1$ dimensional unit ball, and similarly divide the cell into regions containing connected sets of High or Low vertices. The following simple algorithm achieves this goal.

2.3 Spider Web

The *Spider Web* algorithm [9], [19], [18], determines a cell intersecting the iso-surface and then determines hit points along the cell edges using linear interpolation. In the current implementation of the algorithm [10], the pair of hit points on a voxel face with 2 hits are considered adjacent and a decision of pairwise hit adjacency on a voxel face containing 4 hits is made, using a strategy similar to Nielson's asymptotic decider.

The adjacency defines one or several connected sets of hits within the cell. After that, the centroid point of all the hits within a connected set is calculated. This point, called the articulation point (AP), is used as the cell interior triangle vertex for all of the subsequently constructed triangles within that set of hits. Each triangle consists of the AP and a pair of adjacent hits on a cube face (see Figure 5). Hits on a cubic cell face are shared by the cell that shares this face, and thus the algorithm can continue surface construction in the adjoining cells.

Though this algorithm has the advantage of not requiring a case table, it produces more triangles than *marching cubes*. On the other hand, it is completely parallelizable, as each cell can be processed independently. The primary cases where it produces more triangles are when the hit set consists of 3 or 4 hits. In the former case one can remove the articulation point to produce one triangle. When there are 4 hits, *marching cubes* produces two triangles. However the inclusion of the interior AP and the subsequent 4 triangles produces a better approximation of the small scale curvature of the surface.

Also, unlike the original *marching cubes*, it is easily proven correct. We sketch the proof here since the ideas are used in the proof of theorem 1 (see Figure 6). To prove that the surfaces produced are manifolds one establishes two facts. Each traingle edge is shared by two triangles, and the set of triangles incident on any triangle vertex are homeomorphic to a unit disk. The proof uses the fact that each hit has a unique adjacent hit in each cell face on which it occurs.

The triangle edge between an AP of a cell C and a hit h is shared by two triangles: one triangle for each for each of the two faces of C that share this hit. Each triangle consists of the AP, h, and the unique adjacent hit in that face. The triangle edge connecting two adjacent hits in a cell face is shared by a triangle to an AP in the cell that shares this face.

The triangles incident on an AP form a triangulation of the circle, as starting from any hit h in the connected set and following hit adjacency, one returns to h and establishes that the hits form a cycle. Similarly, if h is any hit point, h is incident to 8 triangle edges: 4 to the unique adjacent hit in each of the 4 cell faces sharing the cubic cell edge on which h occurs, and 4 triangle edges, each

to an AP of each of the 4 cubic cells sharing the cell edge on which h occurs. Again these 8 triangles will form a cycle.

The algorithm can be extended to higher dimensions. For an $n > 3$ dimensional cell, recursively build the (hyper) surface patches in the $n-1$ dimensional boundary cells. Each of these patches will consist of $n-2$ dimensional simplexes and each patch will be homeomorphic to an $n-2$ dimensional ball. Now for each connected set of hits in the n dimensional cell select an AP. For each pair of adjacent hits in the set construct 2^{n-3} simplices, each consisting of the AP and an $n-2$ dimensional simplex that contains the pair in a boundary cell. Each such simplex will consist of the pair of hits, the AP, and $n-3$ lower dimensional articulation points.

The simplices thus formed with the AP for the set will form a (hyper) surface patch homeomorphic to an $n-1$ dimensional unit ball. For example, in a 4 dimensional cell, the lower dimensional patches will be the normal triangular meshes in the 3 dimensional boundary cubes. For a connected set of hits, these patches will form a closed surface within the 4 dimensional cell, topologically equivalent to a sphere (by the proof of correctness of the original Spider Web). Tetrahedra will be formed from an interior AP to each triangle on the closed surface, forming a 3 dimensional surface patch equivalent to a ball. This patch will intersect the patches in neighboring hypercubic cells by completely sharing a 2D surface patch in a 3D boundary cube, since this boundary cube will be completely shared by two neighboring 4 dimensional cells. In this way a collection of 3 dimensional manifolds is constructed.

Finally, analyzing this behavior of *Spider Web* (and *marching cubes*) led its inventors to develop the new techniques (Digital Morse theory) summarized below [21],[20], [17],[10].

3 Mathematical analysis: Digital Morse Theory explains why regridding and perturbation are unnecessary

The basic idea of the analysis is as follows. We characterize interpolants that produce contours topologically equivalent to a corrected *marching cubes* (or Spider Web for higher dimensions) or the standard techniques for tetrahedral cells. For any isovalue one labels each data reading as either High (greater than or equal to the isovalue) or Low (less than the isovalue). The topology of the isosurfaces produced by the aforementioned algorithms is uniquely defined by this binary image, up to the disambiguation required for 4 hit faces. The behavior is such that the surfaces produced induce connected components of space. We call the objects the components that are induced by (cell edge and cell face diagonal) connected sets of Highs and complementary objects are the components induced by connected sets of Lows (here connectivity is reminiscent of digital topology [15]). Thus as the isovalue is varied so that it passes through no data or disambiguation value the topology is invariant. As the isovalue is

decreased all that can happen is that at a data value, one or several Lows can become High, or at a disambiguation value, two diagonally opposite Highs in a cell face can become locally connected. Topological changes to the contours can only occur in the following ways.

A new component of Highs can be created (a new object is formed) when one or several Lows become High, or a component of Lows can vanish (complementary object destroyed). Two or more components of Highs can be merged when one or several Lows become High or connectivity of Highs changes at a disambiguation value, resulting in separate objects merging. Similarly, a component of Lows can be split, resulting in complementary objects splitting. Finally a change in the connectivity of a component of Highs (or Lows) can induce a change in the topological type of one or more of the contours bounding the object (or complementary object).

The important point is that one can infer these changes from the local pattern of the data readings by analyzing the behavior of the isosurface algorithms, without any recourse to Morse Theory. We can give a combinatorial characterization of the topology changes, without having to consider the analytic properties of the underlying interpolating function.

3.1 Definitions

Definition 1 *We will assume the volume data set is given by a real-valued function δ defined on a discrete set of points V in R^n. Regularly gridded data is given by a function defined on Z^n, where we form unit hypercubes in the natural way. Irregularly gridded data is given on the vertices of a simplicial decomposition of R^n and we assume that vertex adjacency information is represented in some form in external storage. We term both hypercubes and simplices, cells. Without any loss of generality, we shall assume δ is non-negative over the entire domain, and that $\delta > 0$ on a finite sub-set of points. If δ does not take the same value on any two points, we say that it satisfies data uniqueness.*

For simplicity, we will first describe our algorithm for volume data that satisfies data uniqueness. Later we will extend it for non-unique data, where identically valued spatially proximate data readings imply isovolumes.

Definition 2 *A continuous real valued function f interpolates δ if it extends δ to all of R^n and is nonzero only on the finite subset of cells for which δ is nonzero.*

We denote by $f^{-1}(\geq \tau)$, the set of $p \in R^n$ such that $f(p) \geq \tau$, for f that interpolates δ. The topologically connected components of this set are called objects. The components of the complement of this set are called complementary objects, and they of course share common boundary. Similarly, $\delta^{-1}(\geq \tau)$ denotes the set of $p \in V$ such that $\delta(p) \geq \tau$. Clearly if f interpolates δ then $\delta^{-1}(\geq \tau) \subseteq f^{-1}(\geq \tau)$. The points $p \in V$ whose function value is above or equal to (respectively below) the threshold are termed Highs (respectively Lows).

Definition 3 *The isosurface construction problem is to compute the contours of the boundary of the set $f^{-1}(\geq \tau)$, for specific real number τ. We use the notation $B(f^{-1}(\geq \tau))$ to denote the topological boundary of set $f^{-1}(\geq \tau)$, which encloses all the Highs of the data region for isovalue τ.*

The commonly used isosurface extraction methods discussed in the previous section construct $B(f^{-1}(\geq \tau))$ with the simplest topology, consistent with the data. These methods interpolate a single contour intersection point, called a hit, along a cell edge, if and only if the two end points are identified as High and Low, respectively. Objects are defined by connected set of Highs. For irregularly gridded data, connectivity between the Highs is defined solely by cell edge adjacency. However for regularly gridded data, cell edge adjacency is not sufficient, as observed above. The disambiguity occurs when for a range of thresholds a cube face F contains diagonally opposite Highs and diagonally opposite Lows. We call F a 4-hit face, because there are hits on each edge. We will assume that a consistent disambiguation method is chosen.

Definition 4 *The Disambiguation value c with respect to a function f is the maximum threshold for which the diagonally opposite Highs in F are locally connected through the interior of a cell sharing the face.*

Note that once the data points are path connected they will remain connected for all threshold less than c. We associate a disambiguation point with F, having the value c, which may actually be in the cube interior, and extend δ to this point. For isovalue $\tau > c$ we create a pseudo-edge connecting the two diagonally opposite Lows. For any isovalue $\tau \leq c$ a pseudo-edge connects the two diagonally opposite Highs. We extend the definitions of adjacency and connectivity of Highs and Lows to include the pseudo-edges.

3.2 Properties of admissible interpolating functions

We restrict the class of interpolants to those whose contours behave in a manner consistent with standard isosurface extraction methods discussed in the previous section. The following are the properties satisfied by admissible interpolants.

Let τ be an isovalue not in the range of the extended δ (not a data value or disambiguation value).

- For n dimensional data the contours of $B(f^{-1}(\geq \tau))$ form a finite collection of disjoint, compact and oriented $n-1$ dimensional manifolds embedded in R^n. These contours divide R^n into disjoint n dimensional connected components, each of which contains either a single non-empty connected component of Highs or Lows. An example for the 3 dimensional case is shown in figure 8, where the space is divided between disjoint 3 dimensional connected components of Highs and Lows, by their 2 dimensional boundary manifolds.

- The intersection of the contours with any d-dimensional cell C $(d \leq n)$, $B(f^{-1}(\geq \tau)) \cap C$, is a finite set of $(d-1)$ dimensional components called

surface patches. These surface patches are bicontinuously mappable to a $(d-1)$ dimensional unit disk (ball) and the intersection divides C into disjoint d dimensional contractible components, each of which contains precisely the vertices from a single non-empty connected component of Highs or Lows.

For τ in the range of δ we require that $B(f^{-1}(\geq \tau)) = B(\cap_{x<\tau} f^{-1}(\geq x))$. This insures that the boundary encloses all the data readings precisely equal to τ.

Definition 5 *A function f is admissible if and only if it interpolates δ and satisfies the properties listed above.*

Property 1 merely expresses the goal of isosurface construction from a global point of view, and property 2 expresses the type of local non-degeneracy assumed by isosurface construction algorithms. For example, Figure 9 illustrates all possible intersection of an isosurface with a cubic cell face and the associated division of the cell face. For a tetrahedral cell of any dimension there is always single patch that intersects the cell. For cubic cells, the number of distinct surface patches can go up to four. This situation occurs when there exists a particular threshold for which all cell faces are 4-hit faces and the given isovalue is above the disambiguation value of all faces. Each of the 4 High vertices will be part of a disjoint component of Highs.

3.3 Admissibility uniquely defines topology

We now show that these properties are sufficient to uniquely define the topology of irregularly gridded data. For regularly gridded data, the disambiguation values of 4-hit faces are also required to uniquely define the topology of the contours. For regularly gridded data the theorem is really just a tautology; isosurfaces topologically equivalent to SpiderWeb or Marching Cubes are topologically equivalent. More precisely,

Theorem 1 *If f and f' are admissible and additionally, for regularly gridded data, both have the same disambiguation values, then the boundary contours (manifolds) of $B(f^{-1}(\geq \tau))$ and $B(f'^{-1}(\geq \tau))$ are homeomorphic.*

Proof:

We prove this theorem by a series of lemmas (see appendix). Here is an outline of the proof. We first observe that no cell face can have a odd number of hits. The dual of data reading connectivity is the hit connectivity. We identify each hit by the cell edge on which it occurs. Hits are adjacent if they are connected by a curve segment of the contour in a cell face. Property 2 ensures that a pair of hits on a cell face with 2 hits are adjacent. For a 4-hit face the hits will be divided into 2 adjacent pairs (see figure 7). Since both f and

f' are admissible and have the same disambiguation value for all 4-Hit faces, they will have the same hit set and the same connectivity between the hits for any isovalue. Let M and M' be manifolds of $B(f^{-1}(\geq \tau))$ and $B(f'^{-1}(\geq \tau))$, respectively. We form a graph from the surface patches of a manifold where the nodes of the graph are patches and the edges connect intersecting patches. Since intersecting patches must share a common hit and the patches intersect by sharing lower dimensional patches, it follows from property 2 that we can iteratively construct a continuous bijection from M to M'. In other words, one can show that the graphs G and G', formed from M and M', are isomorphic.

□

3.4 Critical values for volume data satisfying data uniqueness

For the rest of the discussion we refer to components of $f^{-1}(\geq \tau)$ as objects and components of $f^{-1}(< \tau)$ as complementary objects. Topology changes occur when the isovalue becomes equal to a critical value. Let us first look at the possible critical values and the associated changes in topology before arguing how the admissibility uniquely defines topology changes.

Let us assume that δ satisfies data-uniqueness. A critical value c is a value at which the topology of the contours of f change as the threshold is decreased through c. Each such change has an associated critical point p for which $f(p) = c$. We use the term critical point in analogy to Morse theory, but we are not actually using any differential properties of the function. We use the admissibility properties of the function to identify the topology changes.

Let $p \in V$. Let N be the set of points in V adjacent to p by cell edge or pseudo edge. Let N_H (respectively N_L)]be the Highs in N (respectively Lows) with respect to the value of p. Compute the connected components of N_H and N_L within the cells that share p but excluding p itself. For example, two Highs are connected if they are connected within the cells that share p by a path that does not include p. Note that the fourth type of criticality only occurs in cubic cells.

1. If $N = N_L$, meaning p has a higher value than all its neighbors, then p is a local maximum critical point and $f(p)$ is a local maximum critical value. A new contour emerges at each of these critical points (see figure 10)

2. If $N = N_H$, meaning p has a lower value than all its neighbors, then p is a local minimum critical point and $f(p)$ is a local minimum critical value. An existing contour vanishes at each of these critical points (see figure 11)

3. If either N_L and N_H consists of more than one connected component then p is saddle critical point and $f(p)$ is a saddle critical value. Two or more contours may merge into one, contours may split into two or more pieces or a change to the topological type of one of the contours (genus change

in 3D) can occur at these critical points. Figure 12 gives an illustation of these topological changes at saddle critical points.

4. Let p be the disambiguation point of a cell face F with disambiguation value c. If the Highs of F (respectively Lows) are not part of the same same component within the cells that share F above (respectively below) the disambiguation value then p is saddle critical point and c is a saddle value.

Critical points of type 1, 2 and 3 can be encountered both for simplicial and regularly gridded data, whereas critical points of type 4 can occur only for the latter data type. Topology changes to the contours can only occur at critical values.

Theorem 2 *Let f be admissible and δ satisfy data uniqueness. If $[\tau_1, \tau_2]$ is a critical value free interval then $B(f^{-1}(\geq \tau_1))$ and $B(f^{-1}(\geq \tau_2))$ have the same topological type (homotopy type). The converse holds in 3-dimensions.*

See the appendix for the proof. Note that we have not proved the converse for dimensions $n > 3$, as it is beyond the scope of this paper.

3.5 Critical isosets when data does not satisfy uniqueness

If the data does not satisfy data uniqueness then the properties of the admissible functions imply that connected sets of data points (as Highs) of identical value c will be enclosed in isovolumes at thresholds $c - \epsilon$. As the threshold is decreased through c, an object can merge with the isovolume containing these points. Complex topology changes can occur at the value c of these sets, termed isosets. Let S be an isoset with value c and N, N_H and N_L be defined as previous section. For an admissible function f, in addition to the point criticalities described above there are set criticalities.

1. If $N = N_L$, then S is a maximum isoset and c is a maximum critical value. A new contour emerges at each of these critical isosets.

2. If $N = N_H$, then S is a minimum isoset and c is a minimum critical value. An existing contour vanishes at each of these critical isosets.

3. If either N_L or N_H consists of more than one connected component, then S is a saddle isoset, and c is a saddle critical value. Two or more contours may merge, contours may spit into two or more pieces, or a genus change may occur at these critical isosets

4. If N_H and N_L each consist of a single nonempty component then S is called a regular isoset. In some cases the addition of the isoset to an object (or removal from a complementary object) may cause a change in the topological type of a contour. We call this a critical regular isoset. In 3 dimensions this can cause a genus change in an object or complimentary

object O if, when it merges with O it adds or destroys a handle. This occurs if there exists a loop through N_L or N_H and a loop through S so that the loops are interlocked.

Note that a new object of arbitrary topological complexity with possibly multiple boundary manifolds is created at a maximum isoset. One or several manifolds vanish and multiple objects can merge at a minimum isoset. Extremely complex merges, joins, splits, and Betti number changes can occur at saddle isosets. At a regular isoset, in 3 dimensions for example, a genus change can occur. The conditions are that the structure of the isoset itself, when surrounded by isosurfaces, is topologically complex (not simply connected), and additionally it is joined with an object in an appropriate way. For example, consider a donut-shaped critical regular isoset with flat bottom and sides. Suppose this critical set merges with a solid cube object. If the merge is such that the donut rests flat on top of the cube, then no topological change occurs because boundary of the resultant object is still homeomorphic to a sphere. But if the merge is such that it is glued to the top of the cube resting on its side, then a new handle will be formed causing a genus change. Both of these cases are depicted in figure 13.

We leave it as an open problem to develop a efficient combinatorial algorithm for recognizing critical isosets in all dimensions. Note that since one can construct a simplicial complex representing the isosurfaces one can in fact determine algorithmically whether a regular isoset is critical (and indeed compute both the homotopy groups and the homology groups for our contours!) but this is far from an efficient way of determining criticality. This is not a practical problem as we will regard regular isosets above a specific size as critical. This will only serve to possibly increase the number of zones that we compute (see below) but will not change our results. The important point is that topological changes can only occur at our defined critical values and our initial experiments with CT data indicate large regular isosets are rare.

With the addition of the critical isosets we can drop the data uniqueness requirement and theorem 2 can be revised as follows (see appendix for proof):

Theorem 3 *Let f be admissible, if $[\tau_1, \tau_2]$ is a critical value free interval then $B(f^{-1}(\geq \tau_1))$ and $B(f^{-1}(\geq \tau_2))$ have the same topological type. The converse holds in three dimensions.*

Hence amissibility uniquely determines the topological changes that occur as the isovalue is varied.

4 Topological zones and the criticality tree

4.1 Criticality tree

The criticality tree is a search structure that hierarchically organizes the data cells into zones based on value and proximity, and traces the topological evolution of objects in the data as the isovalue is varied. Each node in the criticality

tree is a criticality. As the threshold is decreased from the value of a critical point p, if q is the first critical point encountered in the same object that contains p, then q is made the parent of p (see figure 14). More formally, let $O_p(\tau)$ be the object containing p at threshold τ. For any τ that is equal to a data value define $O_p(\tau)$ to be the closure of $\cap_{c<\tau} O_p(c)$.

Definition 6 *The nodes of the tree are the criticalities. Let p be a criticality and let q be the criticality of maximum value so that $f(q) < f(p)$ and $q \in O_p(f(q))$. Then q is the parent of p.*

If the data does not satisfy the data uniqueness, q may not be unique, and thus the object containing p may meet several criticalities at the same time. We just arbitrarily order the parentage of these to break ties.

4.2 Topological zones

The intuition behind the topological zones is quite simple. As the threshold is decreased from the value of a critical point p, the object containing the criticality grows and deforms with topologically invariant boundary until it encounters another criticality q, which in some manner changes the topology of the contours bounding the object. The topological zone of criticality p is the volume swept by the topological boundary of this object until it encounters q. Each connected component of this zone is the volume swept by an individual boundary manifold (contour) of the object. Figure 14 gives a visual representation of the zones.

We define the zones by a set difference so that no assumption that the contours vary continuously is required, because if the data contains an isoset the contours will not vary continuously. The topological zone for criticality p is defined as follows:

Definition 7 *Let p be a critical point with critical value x_p and parent q. The topological zone of p, denoted $\zeta(p)$, is the set difference $O_p(f(q)) - O_p(f(p))$. The topological zone components are the connected components of $\zeta(p)$.*

4.3 Properties of the topological zones

Theorem 4 *Let p be a criticality with parent q with $a = f(p)$ and $b = f(q)$.*

1. For each τ satisfying $a \geq \tau > b$, $B(O_p(\tau))$ is contained in $\zeta(p)$ and each component M of $B(O_p(\tau))$ is contained in a single component of $\zeta(p)$. The means for any isovalue between the value of a node that represents the critical point and that of its parent, the contour is completely contained with the zone for that criticality.

2. For an pair $[\tau_1, \tau_2]$, if $a \geq \tau_1 > \tau_2 > b$, $B(O_p(\tau_1))$ and $B(O_p(\tau_2))$ are topologically equivalent.

3. For all τ, if O is any component of $f^{-1}(\geq \tau)$ not containing p, then $B(O)$ and $\zeta(p)$ are disjoint.

4. For all τ, such that $\tau > a$ and $\tau \leq b$, $B(f^{-1}(\geq \tau))$ and $\zeta(p)$ are disjoint.

Proof:

A point $r \in \zeta(p)$ if and only if $f(q) \leq f(r) < f(p)$ and $r \in O_p(f(q))$. For any τ such that $f(p) > \tau \geq f(q)$, each point $r \in B(O_p(\tau))$ satisfies $f(r) = \tau$ by continuity of f. Also, because each such $r \in O_p(f(q))$, $B(O_p(\tau))$ is contained in $\zeta(p)$ by monotonicity of $f^{-1}(\geq \tau)$. The connectivity of the manifolds insures that it is contained in a single component of $\zeta(p)$, establishing claim 1. By definition, $\zeta(p)$ contains no critical point with a value between $f(p)$ and $f(q)$, which establishes the second claim. Claim 3 follows from the connectivity of the $O_p(\tau)$ and the fact that boundary manifolds are pair-wise disjoint by property 1 of admissible functions. Claim 4 follows immediately from the proof of claim 1.

□

The criticality tree and zones organize the data set in a hierarchical fashion, in the following sense:

Corollary 4.1 *The union of all zones in the subtree rooted at node p is completely contained in the class of objects corresponding to p. That is, for any isovalue τ in the range of $\zeta(p)$, $O_p(\tau)$ consists of a portion of $\zeta(p)$ and the union of all zones in the subtree rooted at p.*

4.4 Criticality tree vs. contour tree

There are some essential difference between the criticality tree and the contour tree [2], [3], [32], [31]. The two search structures are contrasted and compared below.

- The contour tree traces the evolution of individual contours while the criticality tree traces the evolution of the objects the contours bound.
- The contour tree does not record a change in the homotopy type (genus in 3D) of an individual contour, while the criticality tree records both a change in the topological type of any contour bounding an individual object, as well as a change in the number of contours bounding an individual object.
- Leaf nodes in the contour tree can be maxima or minima where contours are created or destroyed, as the threshold is decreased, while leaf nodes in the criticality tree are all maxima, where objects are created.
- In a contour tree a node with multiple children can occur when two or more distinct objects merge or when a contour of an object splits. In a criticality tree multiple children only occur when objects merge. Objects cannot split as the isovalue is decreased, though complementary objects can split, resulting in an increase in the number of contours bounding an object.

- As one descends the criticality tree from the root the isovalue is strictly increasing. The root of the criticality tree is the single object that encloses the entire data set. Following a path from the root to a leaf traces the evolution of a single object as the isovalue is increased. All the objects in the subtree of a node p are completely contained in the root object corresponding to p. There is no particular order when following a path in the contour tree.

- The seed cell method [32], which uses the contour tree for finding the active cells, extracts each contour by searching for a seed cell known to have intersected the isosurface in the corresponding super-arc first and then locally propagates from the seed cell. However, for an extremely large data set this local propagation can incur excessive disk seek time. The zone organization method uses a more sophisticated method to reduce I/O operations. Since a zone component file contains all the active cells for a contour, they can be extracted by reading the zone component sequentially, reducing disk seek time to finding the start of the component.

Let us illustrate the difference between the contour tree and the criticality tree through an example. Figure 15 shows the level set of a function f as the parameter τ is decreased, and figure 16 shows the corresponding contour tree and the criticality tree. Initially there are four objects created at maxima 7 through 10. These merge at saddles 5 and 6. Then the contours bounding each of the objects undergo two genus changes, changing from sphere to torus to sphere again at saddles 5' and 5" (at saddles 6' and 6" respectively).

These changes are not reflected in the contour tree. Then the two objects merge at saddle 4, the boundary of the this object becomes toroidal at saddle 4' and a inner boundary is created at saddle 3 by a split. At this point the object comes to enclose a hollow. Note that the contour tree has two edges coming out of node 3, which represents the outer and inner boundary manifolds of the object, respectively. Finally the bubble is closed at minima 2 and this node represents the single object containing the entire data set.

5 Zone organization algorithm

5.1 Computing zones

The input to the preprocessing algorithm is a volume data set. For regularly gridded data the data set is organized as a stack of two-dimensional slices. Each slice contains a two-dimensional array of real numbers. The output is a criticality tree and a direct access file containing a sorted list of the cells (voxels) intersecting each zone. The criticality tree indexes the zone file, with each node containing pointers to the components for the corresponding zone. The preprocessing of the volume data to compute the tree and the cells that intersect each zone component is performed in three phases, which are described below.

The zone computation algorithm proceeds by assigning a label to each data point, which corresponds to the zone that includes that particular point.

- Phase 1:

  In the first phase the data is scanned and critical points (and critical isosets) are identified by computing the connected components of Highs and Lows in the neighborhood of each point and identify isosets by depth-first search. One assigns the label $l(p) = p$, where p is a critical point (or representative point of a critical isoset). A zone file is created for each criticality p The critical points are then placed on max-heap ordered by value, with priority given to criticalities to break ties.

- Phase 2: One then iterates the following step until the heap is empty. Remove the maximally valued point r from the heap with label $l(r) = p$. Point r can either be a critical or non-critical point. We elaborate the actions taken in both the cases below.

  - Non-Critical Point: For a non-critical point $r \neq p$, examine the neighbors of r. If a neighbor q of r is unlabelled then assign a tentative label to q from the label of r (i.e. $l(q) = l(r) = p$) and place q on the heap. The label of r is now finalized and it is placed in the file for criticality p.
  - Critical Point: For a critical point $r = p$, examine the neighbors of r. If there is any unlabelled neighbor assign the tentative label as before. If there is a neighbor q that has a finalized label different from the label of r, or in other words $l(q) = a$, such that $a \neq p$, then make the node that corresponds to the criticality a the parent of the node that corresponds to the criticality r. The flooding of the zone for criticality r is now over. The points that are still waiting in the heap with a tentative label p are relabeled tentatively by a, and each one of them are also placed in the file for criticality r (see discussion below for efficient relabeling). Critical disambiguation points are regarded as adjacent to all the four vertices of the ambiguous cell face.

- Phase 3: One creates a single direct access file for all the zones. One next iterates through the original zone files dividing each zone file into its connected components using a standard depth first search. After the zone components are computed for a zone, a cell list is computed for each component and output to the direct access file. Each zone component stores the cell information (e.g. coordinate, values at all vertices, local maximum and local minimum) of all the cells that intersect the component. The cell information is sorted by the local maximum first followed by the local minimum values in order to able to scan fewer cells during isosurface extraction.

Observe that the algorithm labels the entire data set by highest value first order. That ensures that a zone is expanded as much as possible, by expanding

the zone boundary in the shallowest descent direction, before it is terminated. Also, at any point in time, all points greater than or equal to a particular τ have been labeled. These are precisely that points that comprise the components of $f^{-1}(\geq \tau)$. Whenever a zone labeling for a point p terminates, the points remaining in the heap with a tentative label p are the ones that lie in the cells that contain the boundary of this zone, and the data points with the finalized label of p are precisely the points contained in $\zeta(p)$. Cells that straddle one or several zone boundaries are placed in each such zone.

5.2 Implementing the zone organization algorithm

Our primary focus is limiting the I/O and maintaining as little in-core data as possible throughout the execution. The most expensive step is the relabeling of the boundaries of a zone from phase 2. Instead of immediately relabeling the points we use a lazy approach that results in correct but efficient relabeling. The points that have to be relabeled are left on the heap. When a point is removed from the heap, we trace the parent pointer of the node that corresponds to its current label, up to the root node in the partially built criticality tree, and give a finalized label that corresponds to the root node. While following the parent pointers, the point is also placed in all the zone files corresponding to the nodes on the path. The cell in which this point lies straddles the boundary of each such zone. The algorithm thus runs in time $O(kn \log n)$, where n is the number of data cells and k is the longest path traversed in the tree during relabeling (our experiments with CT data have shown this number to be 3 or less in practice).

During phase 1 the maximum number of slides that are read into main memory at any point in time is restricted to three. Suppose that currently $(i-1), i, (i+1)$ are read into memory. All critical points in the i^{th} slide are detected and then the next slide, $(i+2)$, is read into the same space as $(i-1)$. If the data includes an isoset then additional data needs to be read from arbitrary slides. In that case only the necessary data points are brought into memory, not the whole slide. Execution of any phase can be carried out independently given that the previous phases have been completed. We have implemented the zone preprocessing on a 14 processor Sun Enterprise UltraSparc, and it has been parallelized. The implementation is done in an out-of-core fashion, so that it may be scaled up to larger volume data sets. During Phase 1, only portions of at most 3 slices are kept in memory at one time. During Phase 2 only the data readings on the heap are kept in memory. The amount of in-core data is proportionate to the contour size. We also have a mode where we label only one zone at a time for further space efficiency. If we find that when we go to gigabyte scale datasets, the memory requirements for the heap grow too large (which we don't anticipate, see below) we can use an external memory implementation of the heap. Phase 3 requires marking of visited vertices during depth first search and marking of cells during cell list construction. We presently use an in-core boolean array. However further space efficiency (at the cost of time) could be achieved by using an external array or sparse matrix technology for this marking. The sorting of the cells within each zone is done by an efficient

sort. The sorting time dominates the phase and thus the total time for Phase 3 is bounded by $O(kn \log kn)$, where k is (as above) the maximum number of zones any point appears in.

Since we do not have a exact procedure to identify critical regular sets we assume all regular isosets above an arbitrary size to be critical. However in that case, some zones may be divided by regular isosets that are not actually criticalities. This will not change the properties of the zones that we have defined, but may segment the zone unnecessarily and make it smaller. Again we have found large regular isosets to be rare, even in unfiltered CT data.

6 Applications of the zone organization: A visual navigation tool

We have implemented a visual navigation tool using the criticality tree as a search data structure, with pointers to the zones and the zone components on disk. It allows us to visualize volume data, varying both the isovalue and the spatial viewpoint in real time. This tool is used to construct the contours at any given isovalue τ. The tool first uses the criticality tree to find the active zone components. The program scans through the cell list for each component. Because the cell lists are sorted, as soon as a cell with a max value $max(C) < \tau$ is encountered, the search for active cells is terminated for that zone component. The SpiderWeb algorithm contructs the surface patches within each active cell independently, and thus the construction can be parallelized.

The zone organization allows us to produce a visual atlas of all topologically distinct objects in the data set together with the range of isovalues that reveals each object. A common criticism of the isosurface extraction methods is that they obscure the underlying structure of the scalar field. The zones reveal it. The visualization tool also allows us to increment or decrement the isovalue by any value entered by the user. The tool reconstructs the isosurface from the loaded zone, and only loads a new zone when critical values are passed. The tool can also be used to animate the evolution of the objects as the isovalue is decreased from the global maximum to global minimum by a small increment. Figure 17 shows few snapshots of such an animation done on a simulated data set that has a similar structure to the data set shown in figure 15. The system also allows the user to rotate the objects generated at any isovalue, thus the user can interactively search for the isovalue that reveals a structure of interest. The toolkit loads and visits only a small percentage of the total cells. This is done on a low-end SGI workstation.

6.1 Experimental results

We have experimented on simulated data, and both filtered and unfiltered CT data taken from the Visible Human data set. Unfiltered data produces many more zones. We need to make modifications to reduce the number of zones for unfiltered data. If the zone is small enough (e.g. the size of the zone is less than

	Exp. 1	Exp. 2	Exp. 3
Data description	Simulated Data	Filtered Medical Data	Filtered Medical Data
Size (Row x Col x Slide)	40 x 40 x 40	20 x 20 x 20	100 x 100 x 100
Zone Size (as %)	13.95	16.27	3.09
% of active cells	28.83	11.66	2.21
% of cells loaded	42.6	15.49	2.94
Load utility (%)	67.67	75.28	75.17
Travel utility (%)	82.07	94.87	96.08

Table 1: Experimental Results

or equal to one disk block) we can read this single zone into memory using one disk seek.

Experiment 1 was conducted on simulated data that modeled the objects in the screen snapshots shown in figure 17. In this way we could verify by hand the correctness of the zone computation. Experiment 2 was conducted on a small portion (20 X 20 X 20) of unfiltered medical data, followed by a filtered (standard Gaussian) version of the same data set. We iterated through the isovalue range by a variety of small increments, and verified that all active cells for each contour (for each isovalue) were contained in a zone component. Experiment 3 was conducted with a larger portion of the filtered CT data set (100 X 100 X 100). We found that the unfiltered data produced too many zones. We did not count the I/O used to read the criticality tree file into core, as this is done once at the start of the program. Additionally, as we iterated through the isovalues, in many cases no I/O was performed, as the necessary zone was already loaded. Nevertheless, for experimental purposes we counted the I/O that would be performed in loading the zone. For these preliminary experiments we recorded the average zone component size as a percentage of the data set size, the average number active cells (over the range of isovalues) as a percentage of the data set, and the average percentage of cells that were actually brought into memory. Note that for the simulated data the average number of active cells exceeds the zone size, as at most thresholds several zones were needed to render the objects (a zone contains a single object).

Finally the two most important statistics, in analyzing the performance of our active cell acquisition, are the *load utility* and the *travel utility*. The *load utility* records the average percentage of active cells among those brought into memory. In other words, it records the percentage of loaded cells that were actually utilized to construct the isosurface. The *travel utility* records the percentage of active cells among those actually examined. This is perhaps the most important measure of efficiency. Since the zones are sorted, we stopped examining cells in a zone once the next cell was out of range for the isovalue. Since our goal was to reduce disk seek, we read the entire zone component. However the travel utility indicates the percentage of active cells among those actually examined; we can load into memory only these examined cells if we choose. As

we can see both utility factors turned out to be quite high. It will be interesting to compare these statistics with the other search structures, particularly the interval trees of [4].

The average percentage of in-core data is around 20% for small portions of data, which is quite reasonable, because on average 15% of the total cells are active cells that we actually need to produce the isosurface. When we increase the sample size to 100 by 100 by 100 (for filtered data) the percentage drops far below 10For filtered CT data the percentage of the cells actually examined that were active (part of the surface) averaged between 94.87 and 96.08 percent. As we increase the data set size we expect that the zone size will grow at about $n^{2/3}$. When we increased the data set from about 8000 cells to about 1,000,000 we found that the percentage of data readings in a zone dropped more than five-fold.

For both filtered and unfiltered medical data set, many of zones included fewer than 0.1 percent of the total data points, Such zones do not really reflect any significant change in the topology and can be culled without significant loss of information. A good heuristic for culling would be the number of new data points included in a zone, as opposed to the total zone size, which can include many points on the boundary of several zones. It will be interesting to experiment with using the zone algorithm to reduce noise, as opposed to Gaussian filtering, which may obscure essential features (see Figure 18).

6.2 Scaling up the visualization tool

The topology of the boundary manifold of a zone component remains invariant for any isovalue that lies within the range of that component. This means that as long as we retain the criticalities we can coalesce cells within a zone component up to the limits given by the zone structure without changing the topology of the resulting contours. In this manner we can reduce the tiling complexity either prior to or during extraction, based on the users' demand. Thus, we can have finely rendered surfaces in the area of interest, and coarsely rendered surfaces in other areas. This is different from simplification after construction, see for example [28],[14]). The prospect of dynamic multiresolution (as opposed to preprocessed multiresolution [13]), for cubic cells is particularly exciting for the following reason. Once one has read all the cells from a zone into memory, refining the resolution can be done without further I/O penalty. For example one can form larger cubic cells (say 64 by 64 by 64) from the original cells. To refine the rendering detail recursively, one only needs to include an interior data reading in the larger cell, dividing it into 8 smaller cells. Since this reading is already in memory no I/O is required. For static multiresolution, one can store each zone component at several scales. If the zone component is small enough, all the scales can be brought into memory initially. If not, each scale can be read as needed. Figure 19 shows an example of the different scales. We have not yet employed these multiresolution methods, but they will be essential in producing a very large data set visualization tool with real time capabilities.

7 Future applications and conclusion

It will be very interesting to explore the potential use of topological zones in applications such as multi-modal image registration (see [13], [14], [35] for good surveys of these problems). Note that the criticality tree is invariant with respect to rotation, scaling, translation and uniform value translation of the data set. Two images with same contents rotated at different angle, scaled differently or translated will result in similar or partially similar criticality trees. The criticality trees can be compared to verify whether or not the images have the same content. The image registration problem can then be reduced to registration of relevant portions of the two trees, and the corresponding critical points.

The zone segmentation provides a very natural and efficient way of organizing the data set since the organization is based on the contents of the data set. The whole procedure of the preprocessing is automatic and does not need any user intervention. The zone segmentation preprocessing results in efficient I/O operation which provides a great starting point for overcoming the bottleneck that remains one of the unresolved challenges of visualization and animation of very large data sets. Since the zone segmentation is hierarchical and the criticality tree traces the evolution of objects we believe it can be used as a modelling tool for volume data. The nodes of the subtree rooted at any node p of the criticality tree contain all the cells in the interior of the object associated with p. Thus the tree itself provides a topologically complete model of all threshold defined objects in the data set.

We have demonstrated the feasibility, correctness and utility of the zone organization in visualizing volume data sets. We believe that we have provided a basis for exploiting this powerful technique that will ultimately result in the ability to visually navigate extremely large volume data sets in real time without special purpose hardware.

8 Appendix

Proof of Theorem 1:

Lemma 1 *Let e be a cell edge. Then e has a single intersection point (hit) with $B(f^{-1}(\geq \tau))$ if and only if the incident end points are High and Low, respectively.*

Proof:
This follows immediately from property 2.
□

Lemma 2 *Each 2 dimensional hypercube face F (square) has 0, 2, or 4 hit points. Each 2 dimensional simplex face F (triangle) has 0 or 2 hits.*

Proof:
This follows from a simple parity argument or by examining the 4 possible arrangements of Highs and Lows on the 4 vertices of a

cube face. 1, 2 or 3 edge adjacent Highs give 2 hits and a pair of diagonally opposite Highs gives 4 hits (See figure 9). □

Definition 8 *Two hits are adjacent in a face F if they are connected by a curve segment of $B(f^{-1}(\geq \tau)) \cap F$. Connectivity of sets of hits is defined by the transitive closure of adjacency. Connected components are defined accordingly.*

Lemma 3 *Each hit point h of face F is connected by a curve segment of $B(f^{-1}(\geq \tau)) \cap F$ to a unique hit point h' in F, that is adjacent to h.*

Proof:
The result follows from the previous lemma, the disambiguation rule and property 2 (See figure 9).
□

Definition 9 *We uniquely identify a hit point by the cell edge on which it occurs.*

Lemma 4 *$B(f^{-1}(\geq \tau))$ and $B(f'^{-1}(\geq \tau))$ have the same hit sets and hit connectivity if f and f' are both admissible.*

Proof:
Changing the interpolation function does not change whether a vertex is High or Low. For regularly gridded data, f and f' both have the same disambiguation values resulting in the same hit connectivity.
□

Definition 10 *Two surface patches are adjacent if they intersect.*

Lemma 5 *Two $d-1$ dimensional surface patches of M, $\sigma \in C$ and $\sigma' \in C'$ are adjacent if and only if their intersection is a set of $d-2$ dimensional surface patches of M in the $d-1$ dimensional boundary cell shared by C and C'. Further, if σ and σ' are adjacent then they share at least one hit point in each component of their intersection.*

Proof:
The fact that they must intersect in some lower dimensional patch follows from the fact that the $d-1$ dimensional boundary cell is completely shared by C and C' and property 2 of the admissible function. The fact that they share a hit point in each component of their intersection is due to the fact that their intersection must occur on at least one cell edge, and thus they share the hit along this cell edge.
□

Lemma 6 *Each connected component of hits in cell C is a member of a unique surface patch $\sigma \in B(f^{-1}(\geq \tau)) \cap C$. Conversely the hits contained in each surface patch $\sigma \in B(f^{-1}(\geq \tau)) \cap C$, form a unique connected component of hits in C.*

Proof:

By definition of hit adjacency and connectivity, each pair of hits in a connected component of the hits within C is connected by a path within $B(f^{-1}(\geq \tau)) \cap C$. By the path connectivity of the surface patches, this path must be a part of the same patch. Thus the connected set of hits all lie in a single patch σ of C.

For the other direction, we must show that the hit points of σ form a connected set within C. The proof is by induction on the dimension d. The result is clearly true for $d = 1$ and $d = 2$. For inductive hypothesis, we assume that for all $j \leq d-1$, the hits of each patch of a j dimensional cell C is a connected set within C. Let h and h' be any two hit points of σ. Since σ is homeomorphic to the unit ball, its boundary β is connected and contained in the $d-1$ dimensional boundary cells of C. Then there is a path $\pi \in \beta$ from h_1 to h_2 such that this path π passes through a sequence, $\sigma_1, \sigma_2, \ldots, \sigma_m$, of adjacent $d-2$ dimensional patches in the $d-1$ dimensional boundary cells of C. Let h_i be any hit point in σ_i and h_{i+1} be a hit point in σ_{i+1}, where $1 \leq i < m$. We claim that h_i and h_{i+1} are part of the same component of hits of C.

To prove the claim we first observe that since σ_i and σ_{i+1} are adjacent, by lemma 5 they share at least one hit point. By the induction hypothesis, the hits of σ_i and σ_{i+1} are each connected individually within their respective boundary cells. Thus the union of the hits of σ_i and σ_{i+1} must be part of the same connected set of hits in C. This also implies that h_i and h_{i+1} are part of the same connected component of hits in C, thus establishing the claim. Any two hit points in σ are part of the same connected set. In other words, the hit points contained in any surface patch σ form a unique connected component of hits within the cell C.

□

Lemma 7 *Each connected component of hits is contained in a unique component manifold M of $B(f^{-1}(\geq \tau))$ and conversely the hits contained in each such manifold M form a unique connected component of hits.*

Proof:

A connected component of hits is contained in a single manifold M by the definition of hit adjacency and hit connectivity. This manifold is unique, since by property 1 the manifolds are pair-wise disjoint.

For the other direction, suppose that M contains two hit points h_1 and h_2. There is clearly a path π in M from h_1 and h_2. This path passes through some sequence of patches of M, $\sigma_1, \sigma_2, \ldots, \sigma_m$ of cells $C_1, C_2, \ldots, C_m$, where $h_1 \in \sigma_1$ and $h_2 \in \sigma_m$. For each $1 \leq i < m$, σ_i contains a unique connected component of hits within C_i by lemma 6. For each such i , σ_i and σ_{i+1} intersect in a shared boundary cell of

C_i and C_{i+1} and share at least one hit point by lemma 5. Thus the hit points of σ_i and σ_{i+1} are part of the same connected component of hits, which means h_1 and h_2 are part of the same connected set of hits as well.

□

From the previous lemma we get

Lemma 8 *If f and f' are both admissible, $B(f^{-1}(\geq \tau))$ and $B(f'^{-1}(\geq \tau))$ consist of the same number of components.*

Let H be a connected component of hits. Let M and M' be the components of $B(f^{-1}(\geq \tau))$ and $B(f'^{-1}(\geq \tau))$ containing H.

Lemma 9 *Let M and M' consist of the patches $\Sigma = \sigma_1, \sigma_2, \ldots, \sigma_k$ and $\Sigma' = \sigma'_1, \sigma'_2, \ldots, \sigma'_k$, respectively. There exists a one-to-one mapping ψ of Σ onto Σ' so that for all $1 \leq i < j \leq k$, σ_i is adjacent to σ_j, if and only if $\psi(\sigma_i)$ is also adjacent to $\psi(\sigma_j)$. In addition their respective intersections have the same number of components.*

Proof:

The mapping ψ associates two patches if the share the same hit set. The mapping is one-to-one and onto by lemma 6. If σ_i is adjacent to σ_j, then from the above lemmas, they share at least one hit in common. Suppose they share a hit on cell edge e. Then $\psi(\sigma_i)$ and $\psi(\sigma_j)$ also share a common hit one cell edge e and are adjacent.

□

In other words if we represent the patches and their adjacencies by graphs in the natural way, then the graphs representing M and M' are isomorphic. Using the above fact we can prove

Lemma 10 *M and M' are homeomorphic.*

Proof:

We can construct a global homeomorphism from M and M' piece-wise via their patches. From property 2, both patches σ_i and $\psi(\sigma_i)$ are bicontinuously mappable to a unit disk. From lemma 9 we know that σ_i is adjacent to σ_j if and only if $\psi(\sigma_i)$ is also adjacent to $\psi(\sigma_j)$. Also by lemma 5 and property 2 we know that the intersections of the corresponding pair of the adjacent patches are bicontinuously mappable. We can thus bicontinuously map σ_i to $\psi(\sigma_i)$ and extend the mapping for each adjacent patch recursively. Thus we can define the homeomorphism iteratively on each σ_i and extend it to define a global homeomorphism from M to M', proving the lemma.

□

Theorem 1 now follows immediately from lemma 7 and 10.
Proof of Theorem 2:

Proof:

Now the contours that we consider are all manifolds and each can be given by a simplicial complex. For our simple case weak homotopic equivalence implies homotopic equivalence: it is sufficient to know that the homotopy groups of two contours are isomorphic to give topological equivalence. What this means is that we need consider two types of topological changes to our contours: change in the number of contours or change in one of the homotopy groups (and corresponding Betti number) of a contour. Observe that objects and sets of Highs are monotonic in the sense that, as the threshold is decreased, High data readings remain High, and points that were part of the object, remain a part of the object throughout. The Lows becomes High after certain time and new points are added to the objects. Further, from the proof of Theorem 1 we know that no topological changes (in the strong sense of homeomorphism) can occur as the isovalue is varied in an interval that does not contain a data or disambiguation value.

Topological changes to $B(f^{-1}(\geq \tau))$ can thus occur in several ways and of course, several changes can occur at the same time. First of all a new object can be created as the threshold τ is decreased through a value c, and hence a new contour is created. It is straightforward to see from property 1 of admissible functions that this can only happen if a new component of Highs is created, and thus c is a local maximum critical value. In this case there is no object in some region of space for all thresholds $\tau > c$, but a new object exists in the same region for $\tau < c$. By admissibility and data uniqueness, this new object must contain a single data point p and must be comprised of a single connected component of Highs. Thus p is a local maximum critical point. Similarly a contour can vanish at c, and by monotonicity this can happen only if c is a local minimum critical value, where the associated point p comprises a single connected component of Lows for $\tau = c + \epsilon$.

Two or more separate contours can merge at value c. From property 1 and monotonicity this means that there exist two separate components of Highs at threshold $\tau = c + \epsilon$, that are merged into one component of Highs at $\tau = c - \epsilon$. This can only happen if c is a saddle critical value of type 3 and a point p becomes High at c and unites the two components, or c is a critical saddle point of type 4 and the change of the connectivity of Highs in a 4-hit face unites the two separate component of Highs.

One or several contours can split at c. From admissibility property 1 and monotonicity this will happen only when a complementary object (the region containing a component of Lows) is separated by

a change in connectivity at c. This can happen only of c is a saddle critical value of type 3 or 4.

In 3 dimensions a contour can change in genus at c, when a class of incontractible loops is created or destroyed. In higher dimensions either a homotopy class of incontractible k-spheres ($k < n$) is created or destroyed. Monotonicity, data uniqueness and admissibility imply that an increase in a Betti number of a contour can only occur in the following way. As the threshold is decreased through a value c, separate portions of the same object that exist in a neighborhood of a point p at threshold $c+\epsilon$, are merged at threshold $c-\epsilon$, so that a new class of incontractible loops or k-spheres is created. The fact that the object is locally separated in the cells containing p at $c + \epsilon$ implies from admissibility that there must be at least two components of Highs in these cells, thus insuring that p is a saddle critical point. Symmetrically, from admissibility a decrease in a Betti number can occur only when a complementary object is locally separated. Again this means that a component of Lows is locally separated at a value c. From admissibility and data uniqueness this means that either a single Low becomes High or there is a change in connectivity in a 4-hit face. In the former case the data point that changed will be a saddle critical point, and in the latter case the disambiguation point of the 4-hit face will be a saddle.

We now demonstrate that topological changes do occur at critical values in 3 dimensions. By admissibility, a new object is created at a local maximum and a complementary object vanishes at local minimum. Multiple changes can occur at saddles, including any combination of contour merges, splits, and handles created or destroyed. We argue that at least one of these changes occur at a saddle value. First of all the saddle point p, with value c is part of the same global component of Lows at threshold $c + \epsilon$ and at threshold $c - \epsilon$, when p becomes High, the component is separated. By property 1 a boundary contour has split at c. Similarly if two globally separate components of Highs are merged at c, at least one pair of the corresponding boundary contours has merged.

To demonstrate that changes to topological type (genus of a contour) occur at saddle values when objects are not merged (respectively complementary objects are not split), note that the cells N sharing saddle point p (with value c) are homeomorphic to unit ball B, with p at the center and the other data points on the spherical boundary of the ball. Suppose that there are at least two components of Highs of N_H, N_1 and N_2, on the boundary of B at thresholds above c (the argument for N_L is symmetric). By assumption both N_1 and N_2 are part of the same object O. We claim that one can find a loop L on the boundary of B (on the cell faces) that separates N_1 and N_2 on the boundary of B so that L is exterior to O. The definitions of connectivity of Highs and Lows insure that we can

construct L as a loop consisting of cell edges and cell face diagonals connecting adjacent Lows. O will intersect the boundary of B in at least two components O_1 and O_2 containing N_1 and N_2 respectively. By connectivity of O and its boundary contours, one can find a path π in the boundary of O from O_1 to O_2 so that π never intersects the interior of the cells B, and so that at $c - \epsilon$, π can be extended to a loop L' by a path from O_1 to O_2 through the interior of B. Clearly L' is incontractible as it will be interlocked with the above loop L.

□

Proof of Theorem 3:

Proof:

As in the proof of Theorem 2, from admissibility a new object can only be created at the value of a maximal isoset or point and a complementary object can only be destroyed at the value of a minimal point or isoset. A change in a Betti number of a contour can now occur in two ways. In the first way two locally separate portions of an object can merge at a value (resp. a complementary object is locally separated at a value) in such a way that a new homotopy class of incontractible loops or k-spheres, $k < n$ is created (resp. destroyed) on its boundary. From admissibility this means that a globally connected component of Highs is merged at a point or isoset (respectively component of Lows is locally separated) and this implies that the isoset or point is a saddle criticality as we have defined. The second way that the change can occur is caused when an isoset is merged with another object (or removed from a complementary object) so that the structure of the isoset itself adds a homotopy class of incontractible loops (or spheres) to the boundary of the object (or destroys a class of incontractible loops or spheres in the boundary of the complementary object). The isoset in this case is a critical regular isoset.

We now argue that topological changes do indeed occur at critical values in 3 dimensions. A change in topological type occurs, by definition, at the value of a critical regular isoset. A new component of Highs is created at the value of a maximal isoset and hence, from admissibility, a new object is created with at least one new boundary contour. Similarly, a component of Lows vanishes at the value of a minimal isoset and thus a complementary object and at least one boundary contour is destroyed. If two separate components of Highs are merged at a saddle isoset then the corresponding objects are merged with a corresponding change in the number of contours. If a component of Lows is split by a saddle isoset then a new complementary object is created and hence the number of contours changes.

We now consider a saddle in which neither of these two things happens. So assume without loss of generality that N_H consists of

at least two connected sets of Highs (the case of Lows is symmetric) that are globally part of the same object O above the critical threshold c. As in the previous proof, let N be the cells containing the saddle set. Divide N into its connected components by cell edge adjacency. If the components of Highs, N_1 and N_2, occur within different components of N then two distinct objects are merged at c, a contradiction. So assume they are contained in one connected component of N. One can then find a loop L, consisting of cell edges and cell face diagonals connecting adjacent Lows on the boundary faces of N, that lies completely exterior to O. As in the proof of Theorem 2 this loop can be chosen so that a new loop L' formed when these two object portions merge at the threshold c, will be interlocked with the loop L. This shows that loop L is incontractible and thus a change in the fundamental group of one of the contours occurs at the saddle value.

□

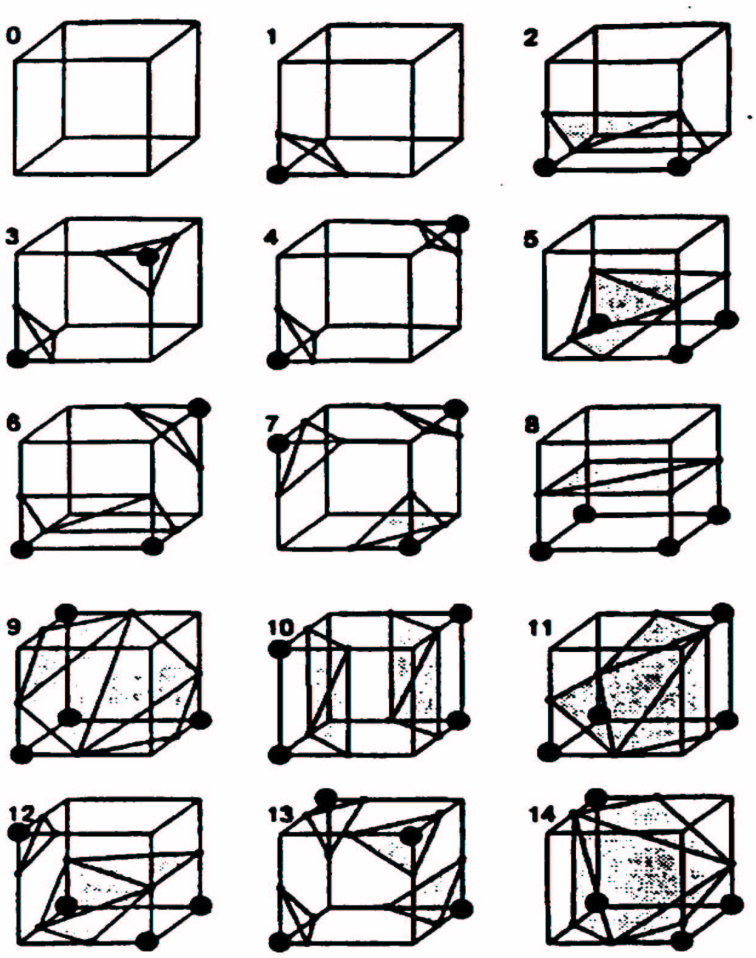

Figure 1: Distinct patterns of surface intersection

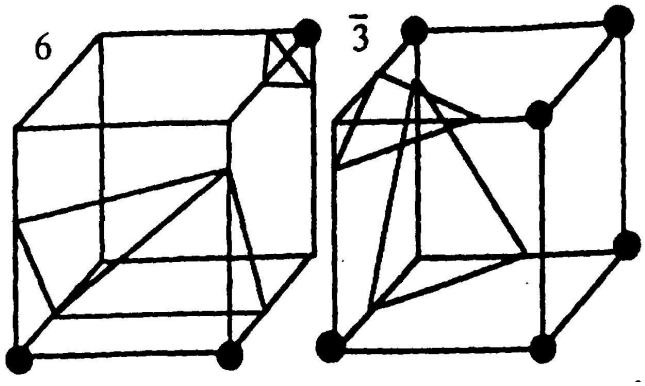

Figure 2: An example illustrating flaw in marching cubes

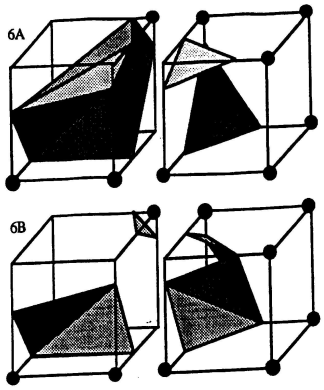

Figure 3: Two possible triangulations which produce correct isosurfaces

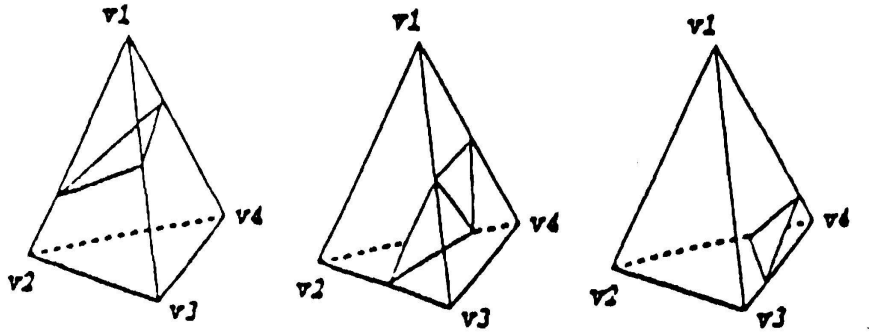

Figure 4: Surface intersection within tetrahedral cells

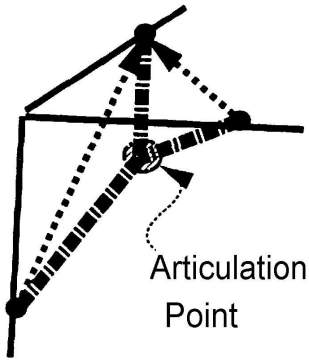

Figure 5: Constructing isosurfaces using Spider Web

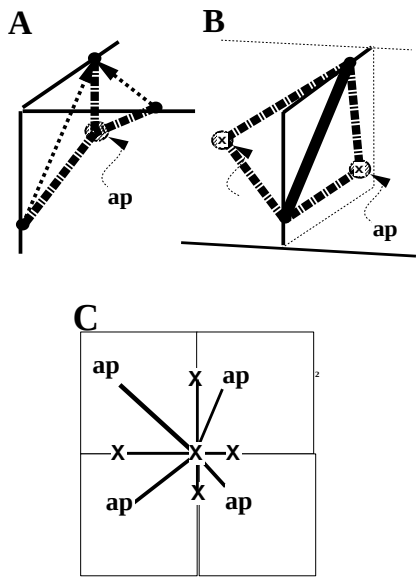

Figure 6: A- The triangle edge from an AP to a hit shared by 2 triangles. B- The edge across a face shared by 2 triangles. C- A hit is locally a triangulation of the circle.

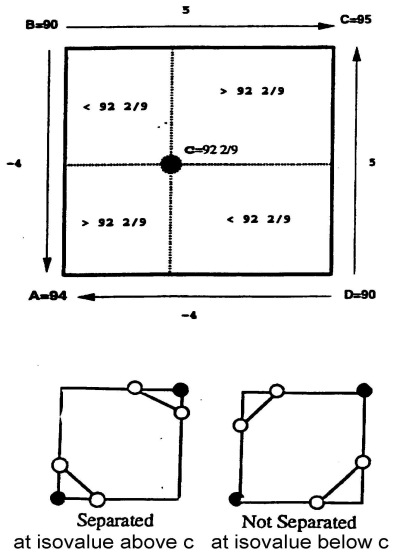

Figure 7: 4-hit face

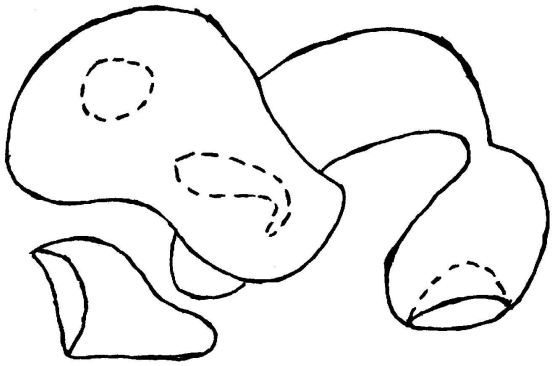

Figure 8: Connected components in 3D

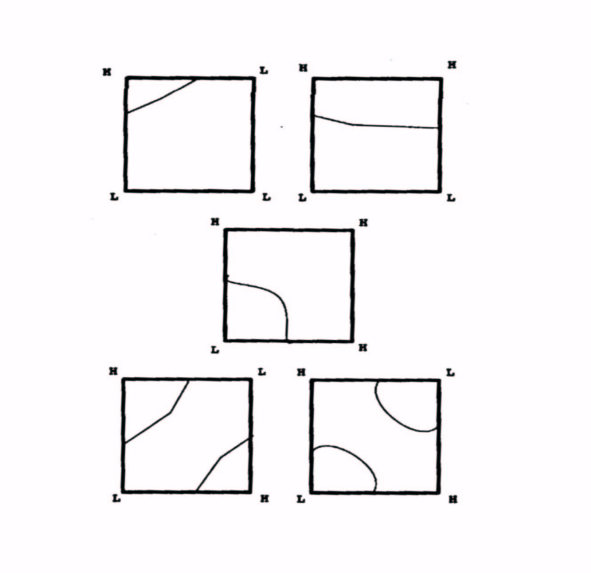

Figure 9: Possible isosurface intersections with cube face.

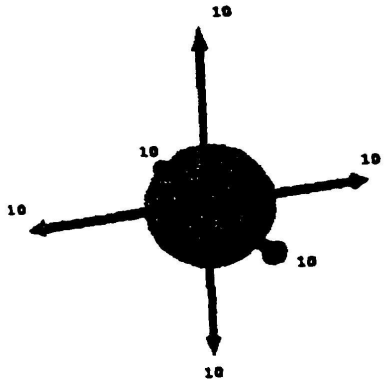

Figure 10: Local maximum critical point

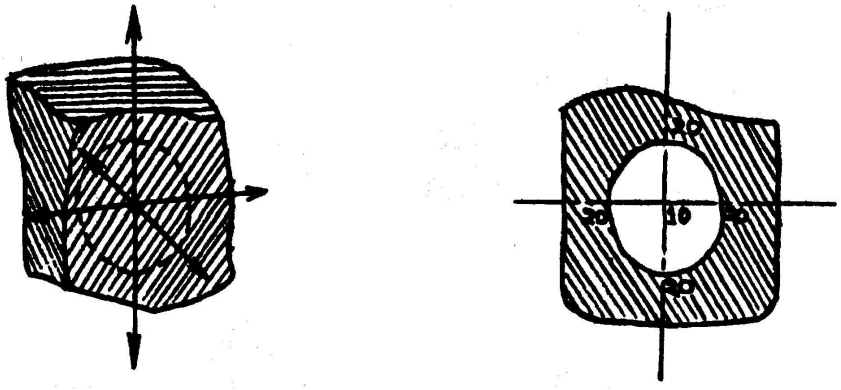

Figure 11: Local minimum critical point

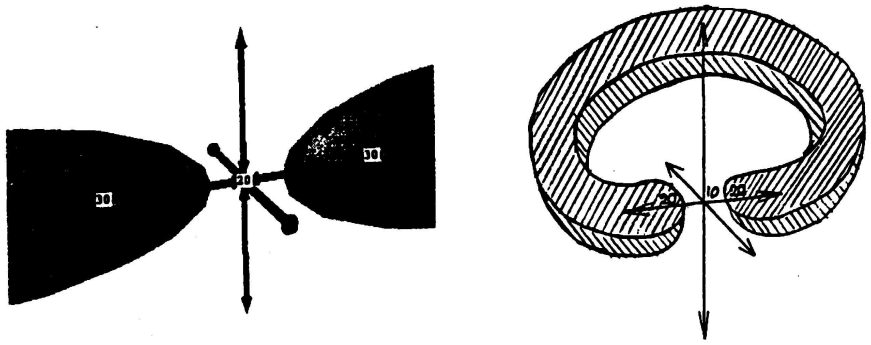

Figure 12: Saddle critical point

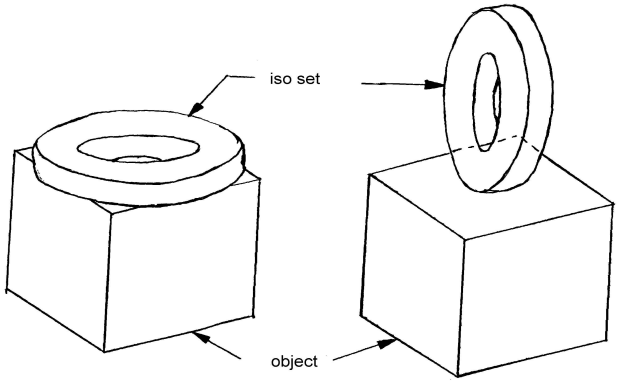

Figure 13: Non-critical and critical regular isoset

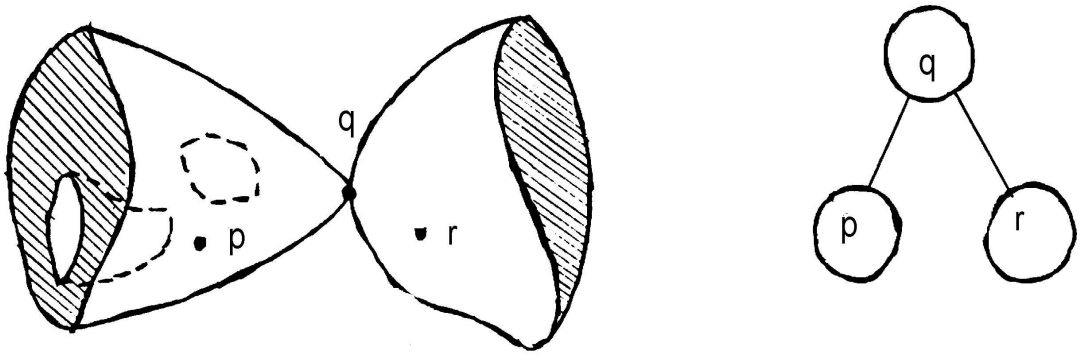

Figure 14: Topological Zones and corresponding Criticality Tree

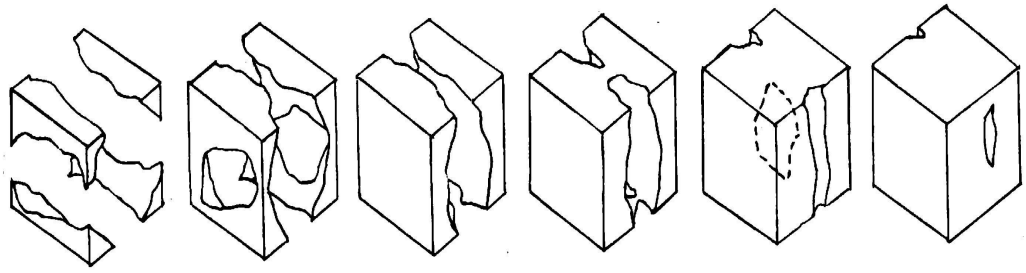

Figure 15: Level set of f as the parameter x decreases

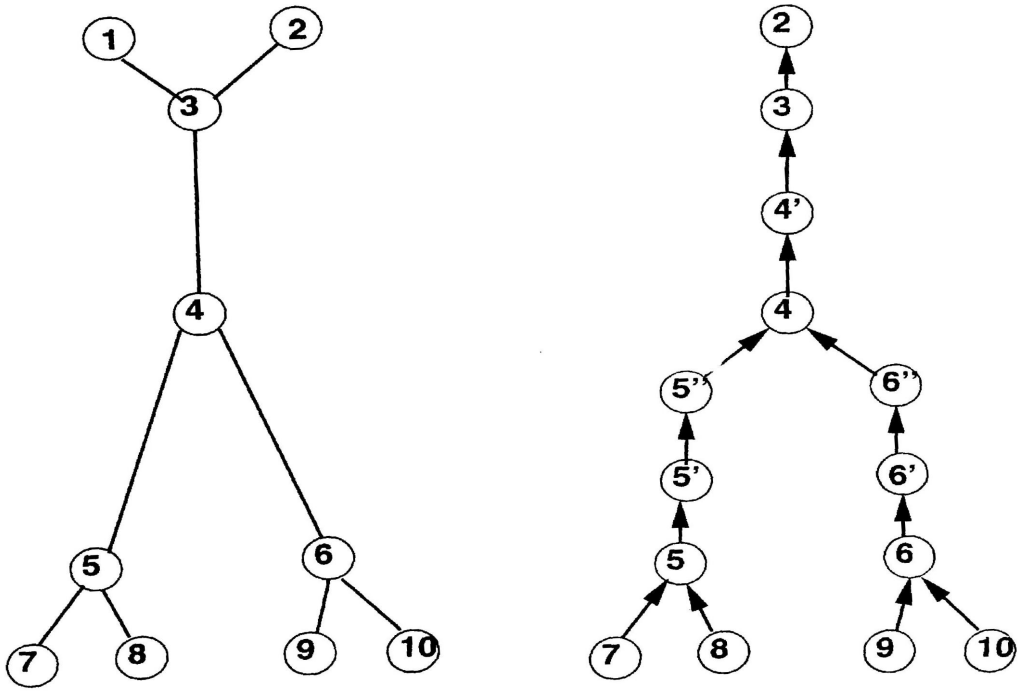

Figure 16: Contour and Criticality Tree

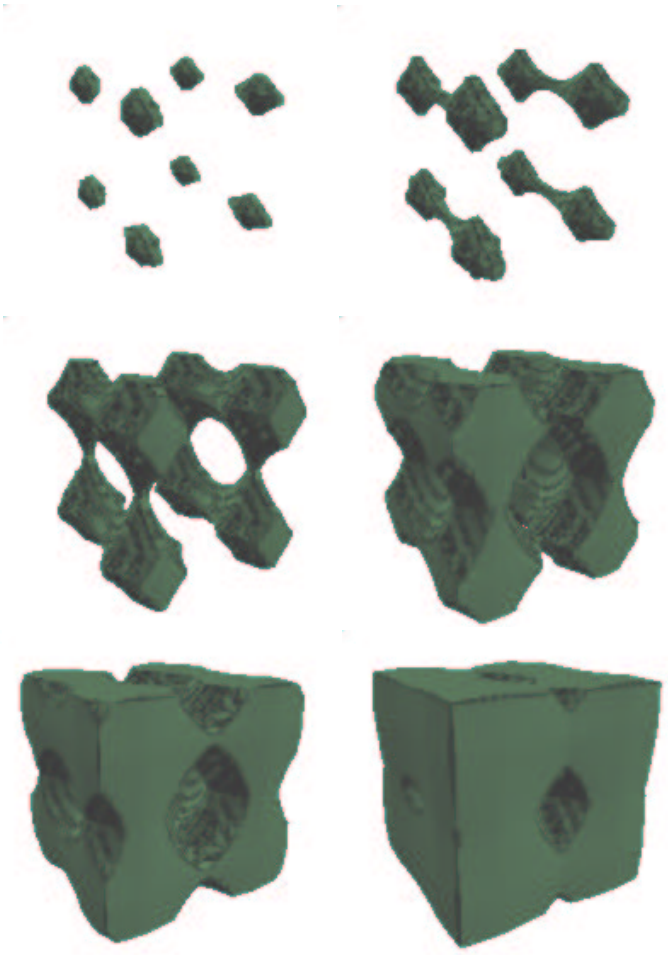

Figure 17: Snapshot of the level set of a function f as the parameter τ is decreased

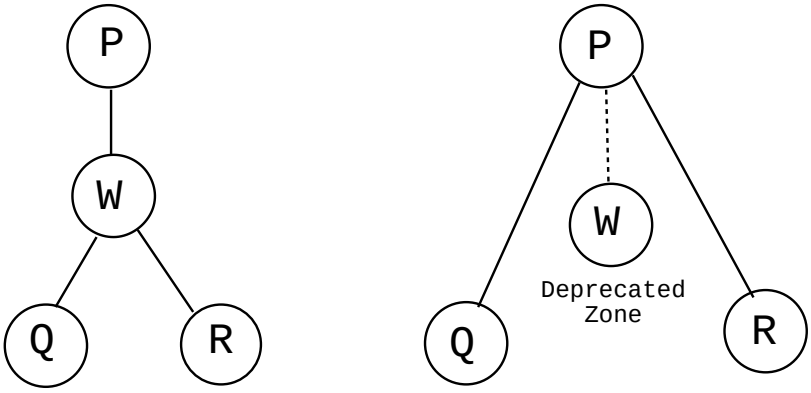

Figure 18: Rearrangement of the criticality tree when a zone is tagged deprecated

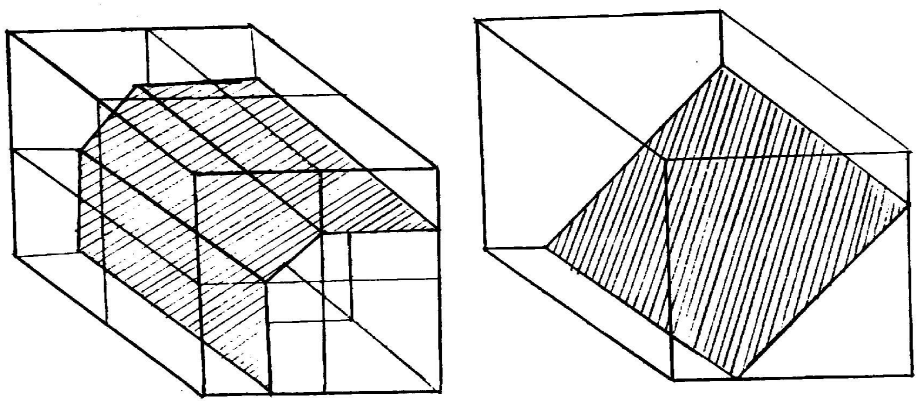

Figure 19: Coarsely rendered or finely rendered without changing the topology

References

[1] R. Avila, T. He, L. Hong, A. Kaufman, and et. al. Volvis: A diversified volume visualization system. In *Proceeings of the IEEE Conference on Visualization 1994 (Visualization '94)*, NJ, October 1994. IEEE, IEEE.

[2] C. L. Bajaj, V. Pascucci, and D. R. Schikore. Fast isocontouring for improved interactivity. In *Proc. 1996 IEEE Symposium on Volume Visualization*, pages 39–46, October 1996.

[3] Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing Contour Trees in All Dimensions. *Symposium on Discrete Algorithms 2000*, page to appear, 2000.

[4] Y.-J. Chiang and C. T. Silva. I/O Optimal Isosurface Extraction. *IEEE Visualization 1997*, pages 293–300, 1997.

[5] Y.-J. Chiang and C. T. Silva. External memory techniques for isosurface extraction in scientific visualization. *External Memory Algorithms (AMS-DIMACS Book Series)*, 50:247–277, 1999.

[6] Y.-J. Chiang, C. T. Silva, and W. J. Schroeder. Interactive out-of-core isosurface extraction. In *IEEE Visualization*, pages 167–174, 1998.

[7] P. Cignoni, P. Marino, C. Montani, E. Puppo, and R. Scopigno. Speeding up isosurface extraction using interval tree. *IEEE Transcations on Visualization and Computer Graphics*, 3(2), April-June 1997.

[8] Harvey. E. Cline, William. E. Lorensen, S. Ludke, and Bruce. C. Teeter C. R. Crawford. Two Algorithms for the Reconstruction of Surfaces from Tomograms. *Medical Physics*, 15(3):320–327, May 1988.

[9] James L. Cox, Daniel B. Karron, and B. Mishra. The SpiderWeb Algorithm for Surface Construction from Medical Volume Data: Geometric Properties of its Surface. *Innovations Et Technologie en Biologie et Medecine*, 14(6):634–656, November 1993.

[10] Nazma Ferdous. *Volume Data Set Visualization using Topological Zone Segmentation.* PhD thesis, City University of New York, 2001.

[11] R.S. Gallagher. Span filter: An optimization scheme for volume visualization of large finite element models. *Proceedings of Visualization '91*, 1991.

[12] A. Guezic and R. Hummel. Exploiting triangulated isosurface extraction using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1(4), December 1995.

[13] P. Heckbert and M. Garland. Multiresolution modeling for fast rendering. *Proceedings Graphics Interface 94*, 1994.

[14] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. *SIGGRAPH 97 course notes*, 1997.

[15] Gabor T. Herman. Oriented Surfaces in Digital Spaces. *CVGIP: Graphical Models and Image Processing*, 55(5):381–396, September 1993.

[16] T. Itoh and K. Koyamada. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. *IEEE Transactions on Visualization and Computer Graphics*, 1(4), December 1995.

[17] D. Karron and J. Cox. Digital morse theory for anatomic modeling. *Proceedings of IEEE BMES-EMBS first joint conference*, 10 1999.

[18] Daniel Karron and James Cox. The spiderweb algorithm for extracting 3D objects from volume data. In Nicholas Ayache, editor, *Computer Vision, Virtual Reality and Robotics in Medicine*, number 905 in Lecture Notes in Computer Science: Proceedings of the First International Conference, CVRMed'95, pages 481–486, Berlin, 1995. INRIA-Sophia Antipolis, Springer-Verlag.

[19] Daniel Karron, James Cox, and Bud Mishra. System and Method for Surface Rendering of Internal Structures within the Interior of a Solid Object. . United States Patent 5,898,793, April 1999.

[20] Daniel B. Karron and James Cox. Extracting 3D objects from volume data using digital morse theory. In N. Ayache, editor, *Computer Vision, Virtual Reality and Robotics in Medicine*, number 905 in Lecture Notes in Computer Science, pages 481–486, New York, NY, 1995. INRIA, INSERM, ECV net, Springer–Verlag.

[21] Daniel B. Karron, James Cox, and Bhubaneswar Mishra. New findings from the spiderweb algorithm : Toward a digital morse theory. In Richard A. Robb, editor, *Visualization in Biomedical Computing – '94*, number 2359 in SPIE Proceedings, pages 643–657. SPIE, SPIE, October 1994.

[22] Arie Kaufman, Reuven Bakalash, Daniel Cohen, and Roni Yagel. Introduction to Chapter 6: Architectures for Volume Rendering. In Arie Kaufman, editor, *Volume Visualization*, IEEE Computer Society Press Tutorial, pages 311–320. IEEE Computer Society Press, Los Alamitor, California, 1991.

[23] Marc Levoy. A Taxonomy of Volume Visualization Algorithms. SIGGRAPH 91 Volume Visualization Course Notes, May 1991.

[24] William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM Computer Graphics*, 21(4):163–169, July 1987.

[25] N. Max. Optical models for direct volume rendering. *IEEE Transcations on Visualization and Computer Graphics*, 1(2), June 1995.

[26] Gregory M. Nielson and Bernd Hamann. The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes. In Gregory M. Nielson and Larry Rosenblum, editors, *Proceedings of Visualization 91*, volume 2, pages 83–91, 1991.

[27] W. Schroeder, K. Martin, and W. Lorensen. *The Visualization Toolkit.* Prentice-Hall, 1996.

[28] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *ACM Computer Graphics*, 26(2):65–70, July 1992.

[29] H. Shen and C.R. Johnson. Sweeping simplices: A fast isosurface extraction algorithm for unstructured grid. *Proceedings on Visualization '95*, 1995.

[30] C. Silva and A. Kaufman. PVR: High Performance Volume Rendering. *IEEE Computational Science and Engineering 1996*, 1996.

[31] S. P. Tarasov and M. N. Vyalyi. Construction of contour trees in 3d in o(n log n) steps. *Proceedings 14th Ann ACM Symposium on Computational Geometry*, pages 68–75, 1998.

[32] M. vanKreveld, R. vanOostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour Trees and Small Seed Sets for Isosurface Traversal. *Proceedings 13th Ann ACM Symposium on Computational Geometry*, pages 212–220, 1996.

[33] K.Y. Whang, J.W. Song, J.W. Chang, J.Y. Kim, W.S. Cho, C.M. Park, and I.Y.Song. Octree-r: An adaptive octree for efficient ray tracing. *IEEE Transactions on Visualization and Computer Graphics*, 1(4), July 1995.

[34] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.

[35] G. Wolberg. 2d and 3d image registration and warping. *SIGGRAPH 99 Conference Course Notes*, 08 1999.

[36] B. Chen Y. Zhou and Z. Tang. An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes. *Computers and Graphics*, 19(3):355–364, 1995.

[37] Y.Livnat, H.W. Shen, and C.R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1), March 1996.

The authors wish to thank Nosson Yanofsky for many helpful discussions of algebraic topology and Bud Mishra for his invaluable assistance.