Isosurface Construction In Any Dimension Using Convex Hulls

Praveen Bhaniramka[1], Rephael Wenger[2], Roger Crawfis[3]

Abstract

We present an algorithm for constructing isosurfaces in any dimension. The input to the algorithm is a set of scalar values in a *d*-dimensional regular grid of (topological) hypercubes. The output is a set of (*d*-1)-dimensional simplices forming a piecewise linear approximation to the isosurface. The algorithm constructs the isosurface piecewise within each hypercube in the grid using the convex hull of an appropriate set of points. We prove that our algorithm correctly produces a triangulation of a (*d*-1)-manifold with boundary. In dimensions three and four, lookup tables with 2^8 and 2^{16} entries, respectively, can be used to speed the algorithm's running time. In three dimensions this gives the popular Marching Cubes algorithm. We discuss applications of four dimensional isosurface construction to time varying isosurfaces, interval volumes and morphing.

Index Terms

Scientific visualization, multi-dimensional visualization, isosurface, contour, interval volumes, morphing, time varying data.

---

[1] SGI Inc., Mountain View, California.
[2] Dept. of Computer and Information Science, The Ohio State University, Columbus, Ohio 43210.
[3] Dept. of Computer and Information Science, The Ohio State University, Columbus, Ohio 43210.

1. Introduction

Given a continuous scalar field, i.e., a scalar function on R^d, an *isosurface* is the set of points with identical scalar values. We wish to construct a piecewise linear approximation to such an isosurface from a regular grid sampling of the scalar field. In 1987, Lorensen and Cline [5] devised the Marching Cubes algorithm for constructing such isosurface approximations in three dimensions. The algorithm constructs an isosurface piecewise within each grid cube. Dürst [3] pointed out some problems with the original algorithm and various modifications were proposed. The Marching Cubes algorithm has gained widespread popularity and acceptance as a tool in visualization of volume data.

In 1996, Weigle and Banks [14,15] presented an algorithm for constructing piecewise linear approximations to isosurfaces in arbitrary dimensions, R^d. Such isosurface approximations consist of (d-1)-dimensional simplices which properly fit together to form a (d-1)-manifold (with boundary.) Weigle and Banks break cubes into tetrahedra or, more generally, hypercubes into d-simplices, and construct the isosurface piecewise within each d-simplex. However, a d-dimensional hypercube breaks into between $d!$ and $2^{d-1}d!$ simplices, depending upon the decomposition used [15]. This increases the time and space used by their algorithm and the complexity of the resulting isosurface by a corresponding factor.

In this paper, we present an algorithm that constructs an isosurface piecewise within each hypercube, thus avoiding the costly split into simplices. Our algorithm constructs the isosurface within a hypercube by finding the convex hull of an appropriate set of points and retaining the portion of its boundary that lies in the interior of the hypercube. We prove that these surface patches correctly join together to form a (d-1)-manifold with boundary.

The Marching Cubes algorithm labels each grid vertex positive, '+', or negative, '−', depending upon whether its value is greater than or less than the isosurface value, called the *isovalue*. (A small perturbation keeps any scalar grid value from equaling the isovalue.) The structure of the isosurface in a cube depends on the positive and negative labels of its eight vertices. Thus there are 2^8 ways in which the isosurface can intersect a cube. Most implementations of the Marching Cubes algorithm first build a table of these 2^8 cases and then use this table to determine the structure of the intersection of the surface within each cube. The actual location of the surface within the cube is determined by linear interpolation of the values at the cube vertices along the edges intersected by the isosurface.

By exploiting symmetry, Lorensen and Cline reduced the 2^8 three-dimensional cases to fourteen. Montani, Scateni and Scopigno [8] added more cases to resolve the ambiguities and inconsistencies discovered by Dürst [3]. A triangulated surface for each of these cases was constructed by hand. The dimension independence of our algorithm allows us to generate isosurfaces from arbitrary dimensional data. Furthermore, since the isosurface is generated piecewise within each hypercube, the algorithm can be used to automatically generate the lookup table for the given dimension and later used to construct isosurfaces, as in the case of Marching Cubes algorithm. The lookup table generated by our algorithm for three dimensions is identical to that proposed in the Modified Marching cubes algorithm [8]. We also use our algorithm to generate the lookup table for the four-dimensional case and use it to generate isosurfaces from four dimensional data.

As in the three dimensional case, the structure of the isosurface in a four dimensional hypercube depends on the positive and negative labels of its sixteen vertices. Thus there are 2^{16} ways in which an isosurface can intersect a four dimensional hypercube. Using symmetry we were able to reduce the 2^{16} possible vertex labelings in four dimensions to 222 cases. However, constructing a triangulated surface for even 222 four dimensional cases by hand would have been a tedious and error prone exercise. By forming the convex hulls of appropriate hypercube vertices and edge midpoints, we determine the structure of the isosurface for each of the 2^{16} cases. Once the table is constructed, the isosurface construction algorithm is based simply on table lookup and linear interpolations along the hypercube edges.

We demonstrate the usefulness of our algorithm by presenting some applications of isosurface construction in R^4. Weigle and Banks discussed one of these applications, visualization of time varying data, in a paper on applications of their isosurface construction algorithm [15]. We also apply techniques from that paper for volume morphing.

Using the four dimensional lookup table, our algorithm allows us to easily and quickly construct isosurfaces for four dimensional data. The most prevalent examples of such data are three dimensional time varying data. We visualize isosurfaces that lie in R^4 by slicing them along different axes. Slicing allows us to present alternative views of such data other than the traditional time animation. Additionally, animation of time-varying data can be smoothed by slicing and rendering the isosurface between coarse time steps.

We can also construct three dimensional interval volumes using isosurfaces in R^4. An *interval volume* is a tetrahedralization of the volume between two isosurfaces defined by two different isovalues. We construct the interval volume by defining a four dimensional scalar field, building an isosurface in that scalar field and then projecting the isosurface to three dimensions. The same technique can be used for any dimension.

Finally, isosurfaces in R^4 can be used to morph isosurfaces lying in R^3. The source and target isosurfaces are identified with parallel hyperplanes in four dimensions. We construct an isosurface in R^4 connecting the source and target isosurfaces and then slice the R^4 isosurface by parallel hyperplanes to animate the morphing from one isosurface to the other. Again this technique can be used in any dimension.

A preliminary version of this paper which did not contain proofs appeared in [1]. In this paper, we prove that our algorithm produces a higher dimensional surface with no "cracking" or "holes" which approximates the true isosurface. We also present additional results and applications of the algorithm.

2. Algorithm

2.1 Basic Algorithm

A d-dimensional regular grid divides a d-dimensional volume into hypercubes. The edges of the grid are the edges of these hypercubes. Our algorithm starts by computing the intersection of the isosurface with each of the grid edges. The isosurface intersects the grid edge if the scalar value at one endpoint of the edge is greater than the isovalue and the scalar value at the other endpoint is less than the isovalue. (By slightly perturbing the value at the cube vertices, we guarantee that the isosurface does not pass through the vertex.) We linearly interpolate between the scalar values at the endpoints to approximate the intersection point of the isosurface and the edge. Let U be the discrete set of all such interpolated points over all hypercubes. Set U will be the vertices of the piecewise linear approximation to the isosurface. Henceforth, we will refer to this approximation as the isosurface.

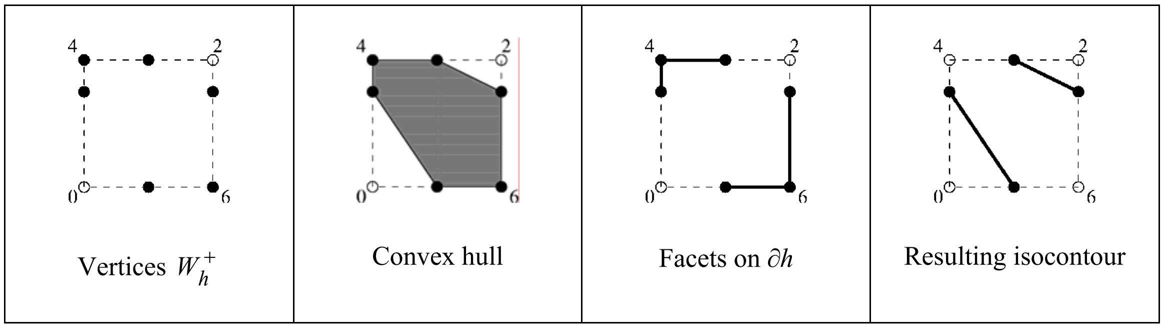


Figure 1. Example of two dimensional isosurface (isocontour) construction, isovalue 3.

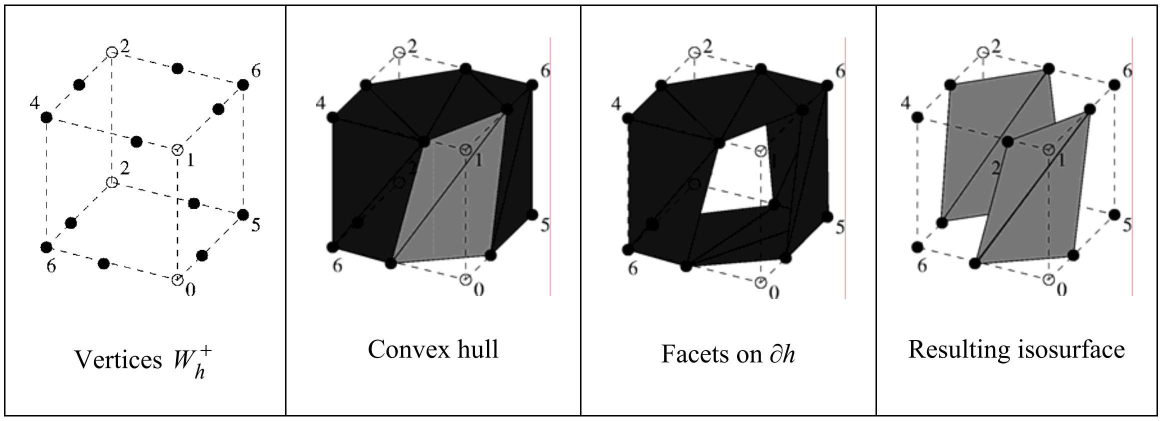


Figure 2. Example of three dimensional isosurface construction, isovalue 3.

Our next step is to reconstruct the isosurface within each hypercube. We do so by approximating the set of points in the hypercube with scalar values above the isovalue. The isosurface is the intersection of the boundary of this point set and the interior of the hypercube.

For each hypercube h, let U_h be the intersection of U and hypercube h, i.e., the points of U which lie on the edges of h. Let V_h^+ and V_h^- be the set of vertices of h whose scalar values are above or below, respectively, the isovalue. Let W_h^+ equal $U_h \cup V_h^+$, the set of interpolated points and the vertices of h with scalar value above the isovalue. (See Figures 1 and 2.) Similarly, let W_h^- equal $U_h \cup V_h^-$. Construct the convex hull of W_h^+.

```
Algorithm:

  1. Approximate the intersection of the isosurface and grid edges using linear
     interpolation;
  2. For each hypercube h do:
       a. $U_h$ ← Intersection point of hypercube edges and isosurface;
       b. $V_h^+$ ← Vertices of $h$ with scalar values above the isovalue;
       c. Construct canonically triangulated convex hull of $W_h^+ = U_h \bigcup V_h^+$.
       d. Remove simplices which lie on a facet of $h$.
```

Figure 3. Isosurface construction algorithm.

The convex hull of W_h^+ is a d-dimensional convex polytope lying in hypercube h and approximating the set of points in h whose scalar values are greater than or equal to the isovalue. Remove the (d-1)-dimensional facets of this polytope that lie on the boundary of the hypercube. The remaining (d-1)-dimensional facets comprise the isosurface in the hypercube. Repeat this process for every hypercube to construct the entire isosurface. (See Figures 1 and 2.)

If the points in U_h are affinely dependent, the (d-1)-dimensional facets comprising the isosurface may not be simplices. For instance, in three dimensions there may exist two dimensional polygons with more than three vertices. We wish to triangulate these facets so that the isosurface can be simply represented by a set of simplices. There are many ways of doing so. Note, however, that this triangulation should be consistent across adjacent facets, i.e., the intersection of any two simplices should be a face of each of the simplices. This is not a problem in three dimensions since two dimensional polygons intersect only on their edges. However, in four dimensions the reconstructed isosurface consists of three dimensional polyhedra. These polyhedra may meet in a two dimensional polygon other than a triangle, for instance a planar quadrilateral. If the triangulations of the two polyhedra contain different diagonals of this quadrilateral, then the resulting triangulations will not match.

Let $p_1,\ldots,p_n$ be points in convex position in R^d. We call a triangulation T' of conv($p_1,\ldots,p_n$) canonical if $T'-p_n$, the simplicial complex T' with all the simplices incident on p_n removed, is a canonical triangulation of conv($p_1,\ldots,p_{n-1}$). A single simplex is a canonical triangulation.

Intuitively, a canonical triangulation is built by incrementally adding points and their incident simplices in the specified order. Clarkson, et. al. in [2] describe an incremental convex hull algorithm which adds points to the convex hull one at a time. As each point is added, the boundary of the new convex hull is constructed from the boundary of the previous one. The algorithm constructs not only the boundary of the convex hull but a triangulation of the boundary. Since the points and simplices are added incrementally, the resulting triangulation of each facet is a canonical triangulation as defined above.

Clarkson, et. al. analyze the expected running time of their incremental convex hull algorithm when points are inserted in random order. To ensure the triangulations between two adjacent hypercubes is consistent, we insert the points in lexicographically sorted order. In the next section we prove that by inserting the points in lexicographic order the ‘canonical’ triangulations of the reconstructed isosurfaces in two adjacent hypercubes properly match at their boundaries. The incremental convex hull algorithm is not optimal, but provides a practical solution to our problem of generating consistent triangulations.

We note that the incremental convex hull algorithm is only one possible way to generate a consistent triangulation of the isosurface. We could have used any convex hull algorithm to construct the isosurface and then used an algorithm by Max [6] to triangulate each of its facets. The algorithm by Max produces a different triangulation than the canonical one described here although both are based on a lexicographical ordering of the triangulation vertices.

Instead of constructing the convex hull of W_h^+ containing the vertices V_h^+, we could have constructed the convex hull of W_h^- which contains the vertices V_h^-. Doing so gives a different, although equally valid, isosurface. However, using $W_{h_1}^+$ for hypercube h_1 and $W_{h_2}^-$ for an adjacent hypercube h_2 could result in a mismatch on the face common to h_1 and h_2. This was essentially the problem discovered by Dürst [3] in the original Marching Cubes algorithm.

2.2 Table Lookup

Constructing a convex hull for each hypercube can be quite time consuming. In dimensions four or less, we can use our algorithm to build lookup tables of surface patches for various configurations and then construct the isosurface within each hypercube using the lookup tables. For dimensions greater than four, the number of table entrees can be prohibitively expensive.

Each grid vertex can be assigned a positive label, '+', or a negative label, '−', depending upon whether its value is greater or less than the isovalue. There are 2^d vertices of a d-dimensional hypercube and so 2^{2^d} different vertex labelings. For each such labeling we construct one entry in the lookup table as follows.

Let h be the unit hypercube with a given labeling. Assign the positive vertices a scalar value of +1 and the negative vertices a scalar value of −1. Apply our algorithm to construct a triangulated isosurface in the hypercube with scalar value 0. Note that the vertices of this isosurface lie on the midpoints of the edges of h. Store this triangulated isosurface in the lookup table entry of the associated labeling. Instead of storing the vertices of the isosurface, we store information on the hypercube edges containing the vertices and the resulting triangulation. The actual vertex coordinates are irrelevant.

To construct the isosurface from a grid of scalar values, we label each of the grid vertices '+' or '−' depending on their scalar values. For each grid hypercube *h*, we find in the lookup table the set of simplices corresponding to the given labeling. Each simplex vertex is a midpoint of some edge of the unit hypercube. Using linear interpolation, we determine the location of the vertex on the corresponding edge of *h* and map the simplex vertex to this location. These mapped simplices form the isosurface patch in *h*.

Our basic algorithm first generates locations of the isosurface vertices and then constructs the triangulated isosurface. In contrast, the lookup table algorithm generates the isosurface triangulation and then adjusts the isosurface vertex locations. These algorithms do not have identical results and may generate different isosurfaces with different triangulations, although the topological structure of the underlying simplicial complexes will be the same. More problematic, we don't know whether moving the vertices in four or higher dimensions can introduce self intersections in the isosurface. This does not happen in three dimensions. We conjecture that this does not happen in four dimensions either, although we suspect that it might happen in sufficiently high dimensions.

Theoretically, isosurface lookup tables can be constructed in any dimension. However, a five dimensional hypercube has 32 vertices and thus 2^{32} possible labelings. This is far too large for a lookup table. One possibility would be to exploit symmetry to reduce the size of the lookup table. Another possibility might be to employ lazy evaluation so that only the most prevalent labelings appear in the lookup table.

3. Proof of Algorithm Correctness

Given a regular grid sampling of a scalar field and an isovalue, we claim that our algorithm constructs a surface that approximates the true isosurface for that isovalue. Our claim has two parts. First, we show that our algorithm does actually construct a surface, not simply a set of arbitrarily connected simplices. More specifically, it constructs a triangulated (d−1)-manifold with boundary in R^d. Note that a valid isosurface need not be a manifold but in many applications it is desirable that the surface be a manifold.

The second claim is that the surface constructed by our algorithm approximates the exact isosurface from the scalar function. Without conditions on the underlying scalar field, it is not possible to reconstruct the exact isosurface from a discrete sampling of the function or even to produce a surface that is close under some metric to the exact isosurface. The exact isosurface must intersect the grid edges with one positive and one negative endpoint, although it could intersect others. We claim and prove that our approximate surface intersects exactly this set of grid edges with one positive and one negative endpoint.

Even in three dimensions there are many topologically distinct surfaces that intersect the same set of grid edges. Without the underlying scalar function, it is impossible to know which surface is homotopic (continuously deformable) to the exact isosurface, although predictions can be made based on certain simplifying assumptions[9]. As with the Modified Marching Cubes algorithm[8], we make no claim that our isosurface is homotopic to the exact isosurface.

Our algorithm returns a set of simplices that we claim form a surface. To prove our claim, we first show that these simplices intersect properly at their faces, forming what is known as a *simplicial complex*. Then, using basic properties of simplicial complexes, we then show that this simplicial complex forms a surface.

To more precisely state our results, we proceed with some definitions. A set of points M in R^d is a (d−1)-dimensional *manifold with boundary* if the neighborhood of each point in M is homeomorphic to either R^{d-1} or a closed half-space of R^{d-1}. Intuitively, a manifold with boundary is a set of points that behave locally like a portion of (d−1)-dimensional Euclidean space or the boundary of a (d−1)-dimensional Euclidean half-space.

A set T of simplices defines a *simplicial complex* if the non-empty intersection of any two or more simplices of T is a face of each of these simplices. For instance, the non-empty intersection of any two tetrahedra is either a (triangular) face, or an edge or a vertex of the two tetrahedra and the non-empty intersection of any three tetrahedra is an edge or a vertex of all three.

More formally, a simplicial complex T is a finite collection of simplices in R^d such that:

- If t is a simplex in T and t' is a face of t, then t' is also a simplex in T;
- If t_1 and t_2 are simplices in T, then $t_1 \cap t_2$ is also a simplex in T.

Note that a simplex can have any dimension up to d and a face of a simplex is also a simplex (of lower dimension.) If T is a simplicial complex, then $|T|$ is the underlying point set, i.e., the union of all the simplices in T. The set T of simplices is called a triangulation of $|T|$.

Let T be the set of simplices returned by our algorithm. We prove:

1. The set T of simplices defines a simplicial complex;
2. $|T|$ is a $(d-1)$-dimensional manifold with boundary.

The first statement is used to prove the second.

We now more precisely define the output of our algorithm. Let G be a regular d-dimensional grid whose vertices are labeled positive, '+', or negative, '−'. Let V^+ and V^- be the vertices of G with positive and negative labels, respectively. Call an edge *bipolar* if one endpoint has a positive label and one has a negative label. Choose one point in the interior of each bipolar edge and let U be the set of all such chosen points. In Section 2.1 set U is a set of linear interpolants of scalar grid values but U could also be edge midpoints or higher order interpolants.

As before, let the sets U_h, V_h^+ and V_h^- be subsets of U, V^+ and V^- restricted to a single hypercube h. Let $W_h^+(U)$ (abbreviated W_h^+) equal $U_h \cup V_h^+$ and $W_h^-(U)$ (abbreviated W_h^-) equal $U_h \cup V_h^-$. Form $S_h^+(U)$ (abbreviated S_h^+) by taking the boundary of the convex hull of $W_h^+(U)$, removing any points on the boundary of h, and taking the closure of the remaining set. Formally, $S_h^+(U)$ equals $cl(\partial\text{conv}(W_h^+(U)) - \partial h)$, where cl, ∂ and $conv$ are the closure, boundary and convex operators, respectively. Similarly, $S_h^-(U)$ (abbreviated S_h^-) equals $cl(\partial\text{conv}(W_h^-(U)) - \partial h)$.

For instance, in Figure 1, set $S_h^+(U)$ contains the two open line segments in the interior of *h*. It also contains the endpoints of these line segments, which lie on the boundary of *h*. These endpoints are in the closure of the open line segments. Similarly, in Figure 2, set $S_h^+(U)$ contains four triangles in the interior of *h*, but it also contains the boundaries of these triangles, portions of which lie on the boundary of *h*.

Let $S^+(U)$ equal $\bigcup_{h \in G} S_h^+(U)$. Our algorithm returns $S^+(U)$. Actually, it returns a set of (*d*–1)-dimensional simplices whose underlying point set is $S^+(U)$.

Each *k*-dimensional face *f* of *h* is also a hypercube, albeit a *k*-dimensional one. Thus for *k* > 0 we can construct a (*k*–1)-dimensional isosurface in *f* in the same manner that we constructed a (*d*–1)-dimensional isosurface in *h*. Let the sets U_f, V_f^+ and V_f^- be subsets of *U*, V^+ and V^- restricted to *f*. Let $W_f^+(U)$ equal $U_f \cup V_f^+$ and $W_f^-(U)$ equal $U_f \cup V_f^-$. Let $S_f^+(U)$ equal $cl(\partial \text{conv}(W_f^+(U)) - \partial f)$ and $S_f^-(U)$ equal $cl(\partial \text{conv}(W_f^-(U)) - \partial f)$.

We could also construct a (*k*–1)-dimensional surface in *f* by simply taking the isosurface from *h* and intersecting it with *f*. We will show that these two constructions are equivalent, i.e., that $S_f^+(U)$ equals $S_h^+(U) \cap f$.

Our proof proceeds as follows. We start with a technical lemma showing that the convex hull of $W_f^+(U)$ is full dimensional, i.e., if *f* has dimension *k*, then so does conv($W_f^+(U)$) (Lemma 1.) This lemma is used for making arguments about the neighborhood of points on $S_h^+(U)$. Next, we prove that the intersection of $S_h^+(U)$ with a face *f* of *h* is $S_f^+(U)$, i.e., it can be constructed directly from the vertices and sample points on face *f* without any reference to the other vertices or sample points in the containing hypercube *h* (Lemma 2.) As a corollary, if sets $S_{h_1}^+$ and $S_{h_2}^+$ agree in the common face *f* between two adjacent hypercubes h_1 and h_2. In other words, $S_{h_1}^+ \cap f$ equals $S_{h_2}^+ \cap f$ (Corollary 1.) Note that this implies that there is no "cracking" between isosurface patches in adjacent hypercubes, the problem plaguing the original Marching Cubes Algorithm[3].

Using Lemma 2, we show that the set of simplices returned by our algorithm forms a simplicial complex (Theorem 1.) For completeness, we provide a simple proof that a simplicial complex is a manifold with boundary if it is a manifold with boundary in the neighborhood of each of its vertices (Lemma 3.) Finally, we apply Theorem 1 and Lemma 3 to show that $S^+(U) = \bigcup_{h \in G} S_h^+(U)$ is a manifold with boundary (Theorem 2.)

Lemma 1: If *f* is a *k*-dimensional face of a grid hypercube, then conv($W_f^+(U)$) is either the empty set or has dimension *k*.

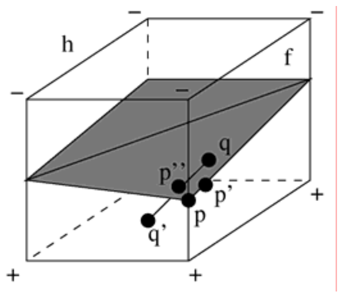
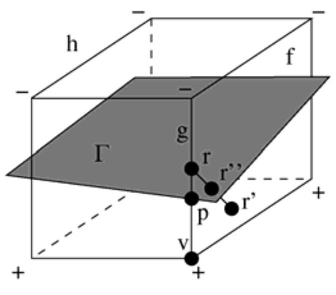
Proof: If $W_f^+ = W_f^+(U)$ is not the empty set, then *h* has some vertex $v \in V_f^+$. For every edge (*v,v'*) in h, either *v'* is in W_f^+ or some interior point of (*v,v'*) is in W_f^+. In either case, some interior point of (*v,v'*) lies in conv(W_f^+). Vertex *v* and interior points from each of its incident edges form a *k*-dimensional simplex contained in conv(W_f^+).

Q.E.D.

Note that *f* need not be a proper face and could be the full hypercube *h*.

By Lemma 1, the interior of $S_h^+(U)$ is identical to the boundary of a *d*-dimensional convex set and so clearly forms a surface. We next show that the intersection of $S_h^+(U)$ and any *k*-face *f* of *h* is completely determined by the vertices and sample points on *f*. This implies that the surfaces defined by $S_{h_1}^+(U)$ and $S_{h_2}^+(U)$ for two adjacent grid hypercubes h_1 and h_2 fit together properly at their boundaries.

We now show that the intersection of S_h^+ and a face *f* of *h* depends only on the vertices and sample points in *f*. The main difficult is that S_h^+ and S_f^+ are defined as the closure of open sets. To prove that a point *p* is in one of these sets, we must find a sequence of points in the interior of the set whose limit is *p*.

 **Figure 4**. Neighborhood of $p \in S_f^+$.	 **Figure 5.** Hyperplane Γ containing *p* and a facet of S_h^+.

Lemma 2: If *f* is a *k*-dimensional face of a grid hypercube *h*, then $S_h^+(U) \cap f = S_f^+(U)$.

Proof: As noted above, abbreviate $W_h^+(U)$ and $S_h^+(U)$ by W_h^+ and S_h^+, respectively. Similarly, abbreviate $W_f^+(U)$ and $S_f^+(U)$ by W_f^+ and S_f^+, respectively. The theorem is trivially true if *f* equals *h* so assume that *f* is a proper face of *h*.

Since *f* is a face of *h*,

$$\text{conv}(W_h^+) \cap f = \text{conv}(W_h^+ \cap f).$$

Since $W_h^+ \cap f = W_f^+$,

$$\text{conv}(W_h^+) \cap f = \text{conv}(W_f^+).$$

Since *f* does not intersect the interior of W_h^+,

$$\partial\text{conv}(W_h^+) \cap f = \text{conv}(W_h^+) \cap f = \text{conv}(W_f^+).$$

Consider a point $p \in S_f^+$. By definition, in every neighborhood of point *p* there is some point *p'* $\in \partial\text{conv}(W_f^+) - \partial f$. (See Figure 4.) The neighborhood of *p'* contains a point $q \in f$ which is not in conv(W_f^+). Since point *p'* lies in conv(W_f^+) $\subseteq$ conv(W_h^+) and, by Lemma 1, conv(W_h^+) has dimension *d*, the neighborhood of *p'* also contains a point *q'* which is in the interior of conv(W_h^+). Since conv(W_h^+) is a subset of *h*, point *q'* is also in the interior of *h*. Line segment (*q,q'*) intersects ∂conv(W_h^+) at a point *p''* which lies in the interior of *h*. Thus *p* is the limit of a set of points in ∂conv(W_h^+) – ∂h and so is in $cl(\partial\text{conv}(W_h^+) - \partial h) = S_h^+$. Since *p* is in *f*, point *p* lies in $S_h^+ \cap f$.

Now consider a point $p \in S_h^+ \cap f$. Some (d–1)-dimensional facet of S_h^+ contains p. Let Γ be the hyperplane containing this facet. (See Figure 5.) Since each facet of S_h^+ intersects the interior of h, hyperplane Γ also intersects the interior of h.

Let g be the smallest face of the hypercube h containing p. Since p lies on f, the face g is also a face of f. (Note that g could equal f.) By the argument given above,

$$\mathrm{conv}(W_h^+) \cap g = \mathrm{conv}(W_g^+).$$

Since p is in $\mathrm{conv}(W_h^+) \cap g$, set W_g^+ is not empty and must contain some vertex v.

We show that hyperplane Γ does not contain g. If hyperplane Γ contained g, then it would contain vertex $v \in g$. Since Γ intersects the interior of hypercube h, hyperplane Γ would separate conv(W_h^+) from some grid edge (v,v') incident on v. Edge (v,v') is a 1-dimensional face of hypercube h but conv($W_{(v,v')}^+$) equals v which is 0-dimensional, violating Lemma 1. Thus hyperplane Γ cannot contain g.

Since g is the smallest face containing p, point p lies in the interior of g. Since Γ does not contain g, but contains a point p in the interior of g, the subspace $\Gamma \cap g$ must separate $\mathrm{conv}(W_h^+) \cap g$ = conv(W_g^+) from some points in the interior of g. Thus, every neighborhood of p in g contains a point r which is not in conv(W_g^+). (See Figure 5.) By Lemma 1, every neighborhood of p in f contains a point r' which is in $\mathrm{conv}(W_f^+) - \partial f$. Line segment ($r,r'$) intersects ∂conv(W_f^+) at a point r'' which lies in the interior of f. This point is not on the boundary of f and so lies in

∂conv(W_f^+) – ∂f. Since p is the limit of points in ∂conv(W_f^+) – ∂f, point p is in $cl(\partial conv(W_f^+) - \partial f) = S_f^+$.

Q.E.D.

The following corollary follows immediately:

Corollary 1: If h_1 and h_2 are grid hypercubes and $f = h_1 \cap h_2$, then $S_{h_1}^+ \cap f = S_f^+ = S_{h_2}^+ \cap f$.

Recall that a triangulation T' of conv($p_1,\ldots,p_n$) is canonical if T'–p_n, the simplicial complex T' with all the simplices incident on p_n removed, is a canonical triangulation of conv($p_1,\ldots,p_{n-1}$). As discussed in the previous section, the algorithm in [2] returns a canonical triangulation of the boundary of the convex hull of a set of points. We prove that if the points are always sorted in lexicographic order, then the canonical triangulation of each facet of the isosurface produces a simplicial complex.

The definition of canonical triangulation still holds if conv($p_1,\ldots,p_n$) and its triangulation T' are not d dimensional. A canonical triangulation of conv($p_1,\ldots,p_n$) induces a canonical triangulation of every face of conv($p_1,\ldots,p_n$).

We call a triangulation of $S_h^+(U)$ canonical if it is a canonical triangulation of all the facets of $S_h^+(U)$. If f is a face of h, then $S_f^+(U)$ is a subset of $S_h^+(U)$ by Lemma 2 and the canonical triangulation of $S_h^+(U)$ induces a canonical triangulation of $S_f^+(U)$.

For each hypercube h, let $T_h^+(U)$ (abbreviated T_h^+) be the canonical triangulation of $S_h^+(U)$ where the vertices are sorted in lexicographic order. Our algorithm returns $T(U) = \bigcup_{h \in G} T_h^+(U)$.

For our next theorem, recall that an edge of G is *bipolar* if one endpoint has a positive label and one has a negative label.

Theorem 1: If G is a regular grid whose vertices are labeled positive or negative and U is a selection of points from the interior of each of the bipolar edges of G, exactly one point per edge, then set $T(U) = \bigcup_{h \in G} T_h^+(U)$ is a simplicial complex.

Proof: Let t_1 and t_2 be simplices in $T(U)$. We must show that $t_1 \cap t_2$ is also a simplex in $T(U)$. Note that t_1 and t_2 may have any dimension between 0 and d–1.

Simplex t_1 is an element of the canonical triangulation of $S_{h_1}^+ = S_{h_1}^+(U)$ for some hypercube h_1. Similarly, simplex t_2 is an element of the canonical triangulation of $S_{h_2}^+$ for some hypercube h_2. Let f equal $h_1 \cap h_2$. The canonical triangulation $T_{h_1}^+ = T_{h_1}^+(U)$ of $S_{h_1}^+$ induces the canonical triangulation T_f^+ of S_f^+. Since f is a face of h_1, set $t_1 \cap f$ is a simplex in T_f^+. Similarly, set $t_2 \cap f$ is a simplex in T_f^+. Since $t_1 \cap f$ and $t_2 \cap f$ are both simplices in T_f^+, the set $t_1 \cap t_2 = t_1 \cap t_2 \cap f$ is also a simplex in T_f^+. Since T_f^+ is a subset of $T_{h_1}^+$, the simplex $t_1 \cap t_2$ is an element of $T(U)$.

Q.E.D.

Let R_+^{d-1} denote the closed half-space of R^{d-1} given by the equation $x_d = 0$. A set M of points in R^d is a $(d-1)$-dimensional manifold with boundary if the neighborhood of each point in M is homeomorphic to either R^{d-1} or R_+^{d-1}. We claim that to show a simplicial complex is a manifold with boundary, we need only check its vertices.

Lemma 3: Let *T'* be a simplicial complex. If the neighborhood of each vertex of *T'* is homeomorphic to R^{d-1} or a closed half-space of R^{d-1}, then $|T'|$ is a manifold with boundary.

Proof: Let *t* be any simplex of *T'*, not necessarily full dimensional. The neighborhood of every point in the interior of *t* is topologically identical, so if the neighborhood of some point in the interior of *t* is homeomorphic to R^{d-1} or R_+^{d-1}, then every point has such a neighborhood. Simplex *t* has a vertex *v* whose neighborhood is homeomorphic to R^{d-1} or R_+^{d-1}. Some point *p* in the interior of *t* lies in this neighborhood and so *p* has a neighborhood homeomorphic to R^{d-1} or R_+^{d-1}. Since some point in the interior of *t* has such a neighborhood, every point in the interior of *t* has such a neighborhood. Since every point in $|T'|$ lies in the interior of some simplex of *T'*, every point has a neighborhood homeomorphic to R^{d-1} or R_+^{d-1}. Thus, $|T'|$ is a manifold.

Q.E.D.

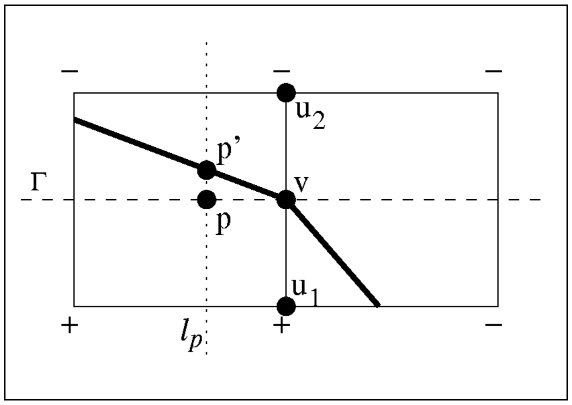


Figure 6. Neighborhood of vertex *v*.

Finally, we show that $S^{+}(U)=\bigcup_{h\in G} S_h^{+}(U)$ is a surface, i.e., a manifold with boundary.

Theorem 2: If *G* is a regular grid whose vertices are labeled positive or negative and *U* is a selection of points from the interior of each of the bipolar edges of *G*, exactly one point per edge, then the set $S^{+}(U)=\bigcup_{h\in G} S_h^{+}(U)$ is a manifold with boundary.

Proof: By Theorem 1, $S^{+}(U)$ has a triangulation *T*(*U*) = $\bigcup_{h\in G} T_h^{+}(U)$. By the previous lemma, we need only show that every vertex of *T*(*U*) has a neighborhood homeomorphic to R^{d-1} or R_{+}^{d-1}. Note that the vertices of *T*(*U*) are all in the interior of hypercube edges. While W_h^{+} contains hypercube vertices, $\partial(h)$ matches (∂conv(W_h^{+})) in the neighborhood of a hypercube vertex, and so hypercube vertices are never included in S_h^{+}.

Let *v* be a vertex of *T*(*U*) in the interior of the regular grid *G*. (See Figure 6.) Vertex *v* lies on some edge (u_1,u_2), where u_1 has a positive label and u_2 has a negative one. Let Γ be the hyperplane through *v* which is perpendicular to (u_1,u_2). For each point *p* on Γ, let l_p be the line $p + \alpha(u_1 - u_2)$ parameterized by α. Let *r'* be the minimum distance from *v* to any simplex in *T*(*U*) not containing *v*. Let r_1 and r_2 be the minimum distances from u_1 and u_2, respectively, to any simplex in *T*(*U*). Note that neither u_1 nor u_2 are on any simplex of *T*(*U*). Let *r* be the minimum of *r'*, r_1 and r_2. Let N_v be a ball in Γ of radius *r* around *v*.

Each line l_p for $p \in N_v$ intersects at least one hypercube *h* containing edge (u_1,u_2). Since line l_p is parallel to edge (u_1,u_2), line l_p intersects conv(W_h^{+}) in a line segment. One endpoint of this line segment is in the neighborhood of v while the other is in the neighborhood of u_1. The endpoint p' in the neighborhood of v is on $S_h^+ = S_h^+(U)$ while the other is not. Map p' to p.

Restricted to each hypercube h, this mapping from S_h^+ to N_v is 1-1 and onto. We must show that this mapping agrees on points in the intersection of two hypercubes. Let h_1 and h_2 be two hypercubes containing (u_1,u_2). Let f equal $h_1 \cap h_2$, a lower dimensional hypercube that is the intersection of h_1 and h_2. By Corollary 1, $S_{h_1}^+ \cap f = S_{h_2}^+ \cap f$. Thus, $l_p \cap S_{h_1}^+ = l_p \cap S_{h_2}^+$ for any point $p \in h \cap \Gamma$. Since the mapping agrees wherever two hypercubes intersect, it is 1-1 and onto. We can restrict any convergent sequence of points to a convergent sequence lying in a single hypercube. Since the map is continuous within each hypercube, it is continuous everywhere.

If v lies on an edge (u_1,u_2) on the boundary of G instead of in the interior, then we can map points of $|T(U)|$ in the neighborhood of v to $\Gamma \cap G$, which is homeomorphic to R_+^{d-1}. The argument is essentially the same as the previous one.

Since the neighborhood of each vertex of v is homeomorphic to R^{d-1} or R_+^{d-1}, set $|T(U)|$ is a manifold with boundary.

Q.E.D.

It remains to prove that $S^+(U)$ intersects the bipolar edges and only the bipolar edges of G. This follows directly from Lemma 2.

Theorem 3: If G is a regular grid whose vertices are labeled positive or negative and U is a selection of points from the interior of each of the bipolar edges of G, exactly one point per edge, then an edge of G is intersected by $S^{+}(U)$ if and only if one endpoint has a positive label and one endpoint has a negative label.

Proof: An edge e of G is a one dimensional face of a grid hypercube h. By Lemma 2, $S_{h}^{+}(U) \cap e = S_{e}^{+}(U)$ for any hypercube h containing e. If e is a bipolar edge, then $S_{e}^{+}(U)$ is a single point $U \cap e$ and $S^{+}(U)$ intersects e. If e is not a bipolar edge, then $S_{e}^{+}(U)$ is the empty set and $S^{+}(U)$ does not intersect e.

Q.E.D.

4. Implementation

Our implementation is composed of two separate applications. The first application generates the isosurface lookup table for the given dimension and stores it in a file to be used later. We use an implementation by Clarkson of the algorithm in [2] to compute the convex hull of the vertices in W_{h}^{+}. (See http://cm.bell-labs.com/who/clarkson.) Clarkson's hull algorithm uses fixed point arithmetic to avoid numerical inaccuracies in the computations. It also computes the convex hull incrementally, adding one vertex at a time, to ensure a canonical triangulation of the various cases.

The complete Marching-Cubes like lookup table for four dimensional contours has 2^{16} entries with a maximum of 26 tetrahedra in a given case. The average table entry contains approximately 13 tetrahedra. In practice, cases that require many tetrahedra are uncommon. For the Jet Shock Wave data set discussed in Section 5.1, the average number of tetrahedra per hypercube was only about 6. Our isosurface generation application took approximately 7 minutes

(3 minutes CPU time, 4 minutes system time) for generating the lookup table for four dimensions on a 1.5 gigahertz Pentium 4 processor running Linux. Approximately 70% of this time was spent in the convex hull computation.

The second application is used to compute and visualize four dimensional isosurfaces. We have used this application with several four dimensional data sets. We visualize the isosurface using three dimensional slices of the four dimensional space. The intersection of these three dimensional slices and the isosurface is a two dimensional surface in three dimensional space, which we rendered using standard graphics techniques. Our implementation allows slicing the isosurface with arbitrary hyperplanes to visualize the relationships among different data attributes (axes). Our isosurface generation algorithm performs topology checks to validate canonical triangulations across hypercube boundaries. It constructs adjacency lists for the simplicial mesh, which would be useful for further processing like mesh simplification.

Since the isosurface generation part of our algorithm is dimension independent, we were able to use it to construct isosurfaces in R^5. A five dimensional hypercube has 2^{32} possible sign patterns on its vertices. The isosurface lookup table would take too long to precompute and be the table would be too large to store. Instead, we used a lazy evaluation method with our algorithm to generate the entries as needed.

Source code for isosurface patch and table generation is available at http://www.cis.ohio-state.edu/graphics/isotable.

5. Applications

5.1 Time Varying Isosurfaces

A number of different techniques have been introduced for fast isosurface extraction and compressed representation of time-varying fields [11, 13, 14, 15]. Our algorithm provides another approach to compact representation of time varying isosurfaces, similar to that of Weigle and Banks [15]. The drawback of their method is that they decompose each 4-cell into at most 192 4-simplices and then recursively contour each of the simplices. This approach produces a larger number of tetrahedra than our method.

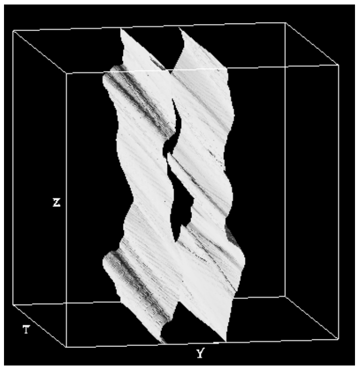
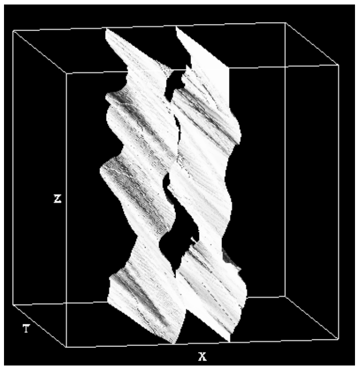
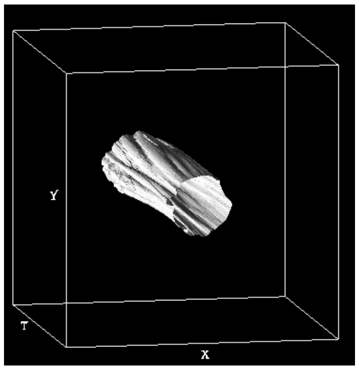
		
Slice along X-axis	Slice along Y-Axis	Slice along Z-axis

Figure 7. Slices of a time varying isosurface for the Jet Shock Wave Data Set along different axes. Isovalue = 37, Timesteps = 56-65.

We output our tetrahedral grid sorted in time, hence, we can easily access the 4D tetrahedra that intersect our time slice. This approach makes the slicing independent of the total number of time steps and speeds up the slicing considerably. On an SGI Octane, computing a time slice is interactive for a 4D isosurface generated from a 40x40x40x36 size data set. Constructing the isosurface in R^4 allows slicing at non-integral time steps, effectively merging the steps of interpolation and isosurface extraction into one, allowing us to generate smooth animations of the time-varying isosurface very efficiently.

For 10 time steps of the Jet Shockwave data set, an isovalue of 37 generated an isosurface with 8,021,739 tetrahedra having 1,394,104 vertices in just under 20 minutes. The isosurface intersected 1,317,975 hypercubes, giving an average of around 6 tetrahedra per hypercube. The total number of triangles generated for the same 10 time steps by Marching Cubes was 1,796,350.

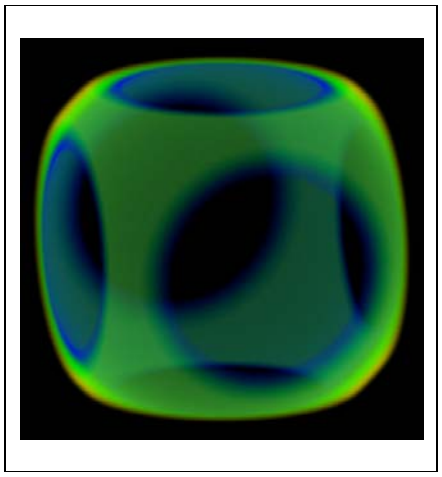


Figure 8. Interval volume for the sphere function.

5.2 Interval Volumes

For a trivariate function $f(x,y,z)$ sampled on a three dimensional rectilinear grid, the interval volume [4] is defined by $I_f(\alpha,\beta) = \{(x,y,z): \alpha \leq f(x,y,z) \leq \beta\}$. More generally, for a function $f : R^d \rightarrow R$ in any dimension, the interval volume is defined by $I_f(\alpha,\beta) = \{(x_1,\ldots,x_d): \alpha \leq f(x_1,\ldots,x_d) \leq \beta\}$. In three dimensions, Fujishuro et. al. [4] discuss applications of interval volumes and propose a solid fitting algorithm for tetrahedralizing the interval volume by extending Marching Cubes. Max et. al. [7] and Nielson et. al. [10] compute the tetrahedralization by decomposing each cube in the grid to five tetrahedra. Nielson then uses an efficient lookup table to compute the interval volume within each simplex and decompose it into tetrahedra. Since they do this for simplices rather than cubes, the number of tetrahedra generated is very large. We show how to compute interval volumes for any dimension d by computing an isosurface in R^{d+1} and then projecting this surface onto R^d.

For a d-dimensional function $f(x_1,\dots,x_d)$ and scalar values $\alpha < \beta$, consider the $(d+1)$-dimensional function $F(x_1,\dots,x_d,t)$ given by:

$$F(x_1,\dots,x_d,t) = f(x_1,\dots,x_d) - (\alpha(1-t) + \beta t).$$

Note that:

$$F(x_1,\dots,x_d,t) = \begin{cases} f(x_1,\dots,x_d) - \alpha & \text{for } t = 0 \\ f(x_1,\dots,x_d) - \beta & \text{for } t = 1 \end{cases}.$$

Let S be the isosurface given by $F(x_1,\dots,x_d,t) = 0$ for $0 \le t \le 1$. Note that the intersection of S and the hyperplane $t = 0$ is the isosurface given by $f(x_1,\dots,x_d) = \alpha$ while the intersection of S and the hyperplane $t = 1$ is the isosurface given by $f(x_1,\dots,x_d) = \beta$. Let π be the projection function mapping R^{d+1} to R^d given by $\pi(x_1,\dots,x_d,x_{d+1}) = (x_1,\dots,x_d)$. The mapping from S to $\pi(S)$ is a 1-1 mapping of S onto the interval volume $I_f(\alpha,\beta)$.

Our algorithm follows directly from the description of the continuous case. Given a regular d-dimensional grid sampling of f, the grid values for F are $f(x_1,\dots,x_d) - \alpha$ for $t = 0$ and $f(x_1,\dots,x_d) -$

β for $t = 1$. Construct the piecewise linear approximation to $F(x_1,\ldots,x_d,t) = 0$ using the algorithm in Section 2. Project this approximation to R^d by removing the last coordinate from each vertex. In the next section, we prove that this projection is the triangulation of the volume between the two piecewise linear surfaces approximating $f(x_1,\ldots,x_d) = \alpha$ and $f(x_1,\ldots,x_d) = \beta$.

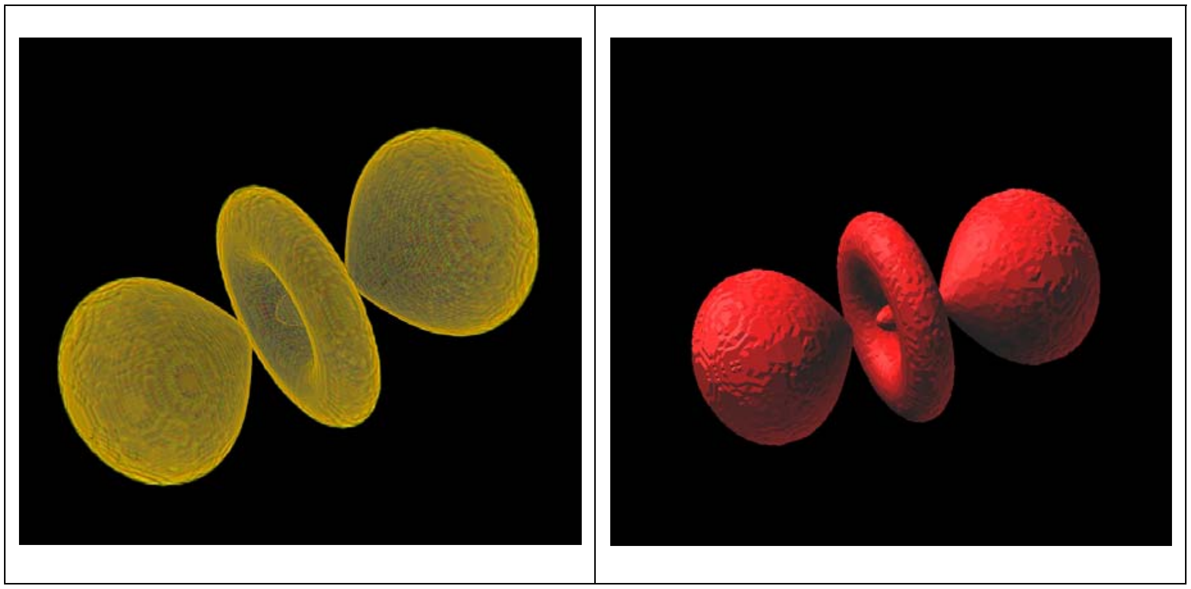


Figure 9. Interval volume and one of the isosurfaces bounding the interval volume for the hydrogen atom data set.

To compare our technique to that of Nielson's, we show the results obtained for the following function sampled on a 14x14x14 grid for $\alpha = 0.35$, $\beta = 0.37$.

$$f(x,y,z) = (x - 0.5)^2 + (y - 0.5)^2 + (z - 0.5)^2.$$

The resulting tetrahedralization consists of 4204 tetrahedra with 1496 vertices as compared to 8500 tetrahedra and 3224 vertices in [10]. Figure 8 shows the interval volume for the above function sampled at a resolution of 32x32x32 to give an interval volume consisting of 44,072 tetrahedra and 11,648 vertices. The tetrahedral mesh was rendered using the Shirley-Tuchman's tetrahedra projection algorithm [12]. Furthermore, the 3D isosurface can be animated to show the contour sensitivity or function gradient by slicing the 4D isosurface. Slicing allows the user to quickly move back and forth between different isovalues to see the contour sensitivity. Sections of the isosurface that are more sensitive to the isovalue can be seen to change more rapidly compared to other sections.

Figure 9 shows an interval for the hydrogen atom data set. The data set is 128x128x128 in size and the resulting interval volume has approximately 797,000 tetrahedra and 142,000 vertices. The first figure shows the interval volume rendered using [12] while the second figure is a slice of the same interval volume.

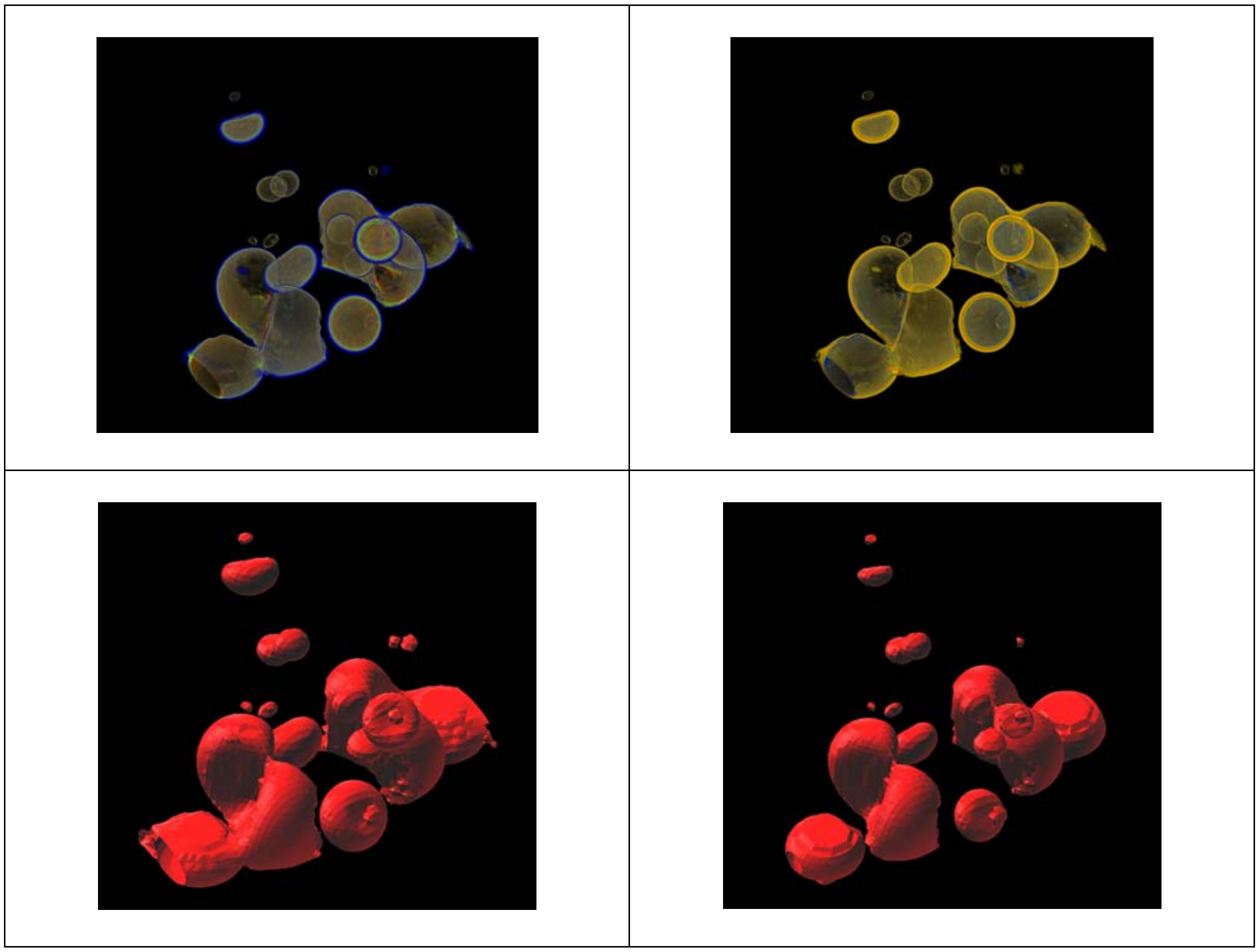


Figure 10. Interval volume using two different transfer functions and the two isosurfaces bounding the interval volume for the Neghip data set.

In order to visualize a larger range of isovalues ($\alpha_1, \alpha_2, \ldots, \alpha_n$) this idea can be extended to tetrahedralize the volume using n steps along the t axis. Thus, the function $F(x_1,\ldots,x_d,t)$ is given by

$$F(x_1,\ldots,x_d,t) = f(x_1,\ldots,x_d) - (\alpha_i\,(1-t) + \alpha_{i+1}\,t) \;\text{ for } 1 \le i < n.$$

Using 5 steps, we generated the interval volume for the neghip protein molecule data set. The data set is of size 64x64x64 and the resulting interval volume consisted of 222,000 tetrahedra and 44,600 vertices. We used the isovalues (110, 120, 130, 140, 150) for α_i in the interval volume. Figure 10 shows the volume rendered tetrahedral mesh using two different transfer functions. Notice the "bands" corresponding to each "interval" in the volume. The volume rendered data shows how the structure of isosurface changes with the isovalue. This level of detail would be difficult to visualize using conventional direct volume rendering of the volume data set. Two slices of the 4D isosurface are also shown.

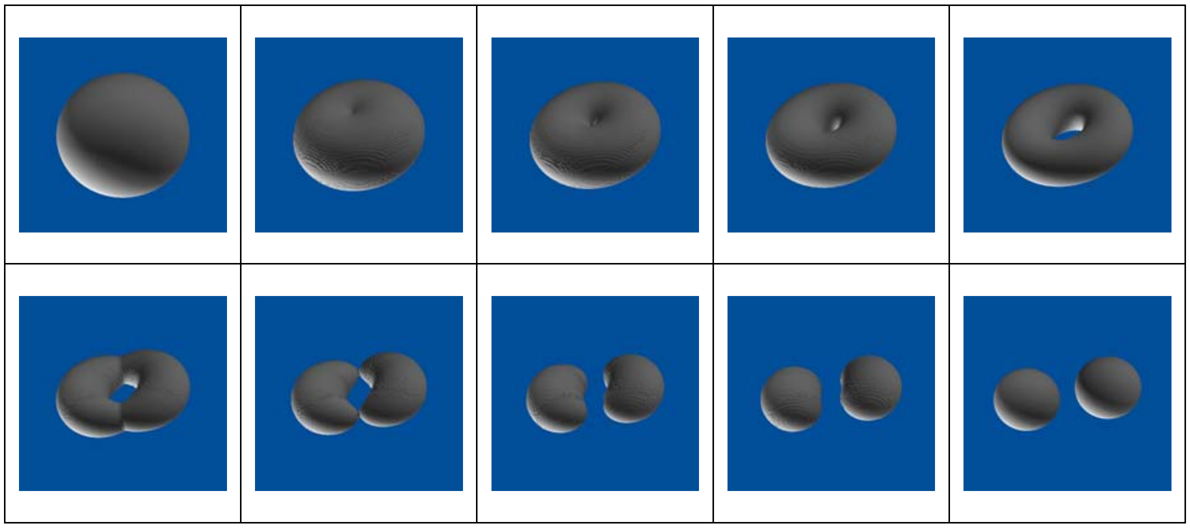


Figure 11. Morph from a sphere to a torus and then to two spheres.

5.3 Morphing

Time-varying isosurfaces can also be used as a compressed representation in volume morphing applications. We use the idea from Weigle and Banks [15] of representing a time-varying isosurface as a four dimensional isosurface and then slicing that isosurface. For two functions $f_0(x_1,\ldots,x_d)$ and $f_1(x_1,\ldots,x_d)$ and scalar values α_0 and α_1, define the (d+1)-dimensional function $F(x_1,\ldots,x_d,t)$ given by:

$$F(x_1,\ldots,x_d,t) = (f_0(x_1,\ldots,x_d) - \alpha_0)\,(1-t) + (f_1(x_1,\ldots,x_d) - \alpha_1)\,t.$$

Note that:

$$F(x_1,\ldots,x_d,t) = \begin{cases} f_0(x_1,\ldots,x_d) - \alpha_0 & \text{for } t = 0 \\ f_1(x_1,\ldots,x_d) - \alpha_1 & \text{for } t = 1 \end{cases}.$$

Construct the piecewise linear approximation to the R^{d+1} isosurface $F(x_1,\ldots,x_d,t) = 0$. At $t = 0$, this surface equals the isosurface $f_0(x_1,\ldots,x_d) = \alpha_0$, while at $t = 1$ it equals the isosurface $f_1(x_1,\ldots,x_d) = \alpha_1$. Intermediate surfaces can be constructed by cutting the R^{d+1} isosurface by the hyperplane $x_{d+1} = t$ for various values of t. Alternatively, one could compute and store a separate d-dimensional isosurface for each value of t. Computing a single isosurface in R^{d+1} is far more time and space efficient, however.

Figure 11 shows a sequence of frames generated using a time varying function. This function is radial at time 0, migrates to a toroidal function at time 1, and then to a union of two radial functions at time 2. This technique easily and effortlessly handles topology changes.

5.4 Isosurfaces in R^5

Applying the interval volume algorithm in Section 5.2 to time varying data, requires constructing an isosurface in five dimensions. An isosurface lookup table for a five dimensional hypercube would have 2^{32} entries, which would be prohibitive to pre-compute. Instead of using the table, we constructed the isosurface patches directly in each hypercube.

We constructed an interval volume for the Jet Shockwave between isovalues 27 and 37 for timesteps 56 and 57. This interval volume consisted of 3.3 million simplices and 2.2 million vertices for the 256x256x256x2x2 size data set. The average number of simplices per intersected 5-cube was approximately 24.

5.5 General convex polyhedra

The algorithm in this paper can be used to automatically generate isosurface patches in any convex polyhedra, not just hypercubes. In particular, it could be used to build isosurface lookup tables for higher dimensional simplices or pyramids. The polyhedra must be convex and all the given vertices and edges must be external. We've used our algorithm to build isosurface tables for various dimensional simplices up to and including dimension 10. Simplices in dimension 10 have 11 vertices, so the isosurface lookup table has 2^{11} entries. Computation of the lookup table took 3 minutes on a 1.5 gigahertz Pentium 4 processor running Linux.

6. Conclusions

In this paper, we presented a simple, efficient algorithm for constructing isosurfaces in d-dimensional grids. Our algorithm can either be used to generate a lookup table of isosurface patches or to directly generate the isosurface patches in each hypercube. Isosurfaces in R^4 of time varying data sets give an alternative to the traditional visualization of such data sets by animation through time. Isosurfaces in R^4 and R^5 can also be used to construct interval volumes of three and four dimensional data, respectively. Finally, isosurfaces in R^4 can be used to morph three dimensional volumes.

7. Acknowledgements

This work was supported by NSF career grant #ACI- 9876022. We would like to thank Zhi Yao, Donglin Liang and Hoseok Kang for contributing to the introductory ideas for this work. We would also like to thank Zbigniew Fiedorowicz for helpful consultations on topology. The Jet Shockwave data is part of the Advanced Visualization Technology Center's data repository and appears courtesy of the University of Chicago.

8. References

[1] P. Bhaniramka, R. Wenger and R. Crawfis, *Isosurfacing in higher dimensions*, in T. Ertl, B. Hamann and A. Varshney, eds., *Visualization 2000*, IEEE Computer Society Press, Salt Lake City, Utah, 2000, pp. 267-273.

[2] K. L. Clarkson, K. Mehlhorn and R. Seidel, *Four results on randomized incremental constructions*, Comput. Geom. Theory Appl., 3 (1993), pp. 185-212.

[3] Dürst, *Additional reference to marching cubes*, Computer Graphics, 22 (1988), pp. 72-73.

[4] I. Fujishiro, Y. Maeda, H. Sato and Y. Takeshima, *Volumetric data exploration using interval volume*, IEEE Transactions on Visualization and Computer Graphics, 2 (June 1996).

[5] W. E. Lorensen and H. E. Cline, *Marching cubes: A high resolution 3d surface construction algorithm*, in M. C. Stone, ed., *Computer graphics (Proceedings of SIGGRAPH 87)*, Anaheim, California, July 1987, pp. 163-169.

[6] N. Max, *Consistent subdivision of convex polyhedra into tetrahedra,* Journal of Graphics Tools, 6(3), 2002, pp. 29-36.

[7] N. Max, P. Hanrahan and R. Crawfis, *Area and volume coherence for efficient visualization of 3d scalar functions*, *Computer graphics (San Diego Workshop on Volume Visualization)*, November 1990, pp. 27-33.

[8] C. Montani, Scateni, R. and Scopigno, R., *A modified look-up table for implicit disambiguation of Marching Cubes*, Visual Computer, 10 (1994), pp. 353-355.

[9] G. M. Nielson and B. Hamann, *The Asymptotic Decider: Removing the ambiguity in Marching Cubes*, *Visualization '91*, 1991, pp. 83-91.

[10] G. M. Nielson and J. Sung, *Interval volume tetrahedrization*, in R. Y. a. H. Hagen, ed., *IEEE Visualization '97*, IEEE, November 1997, pp. 221-228.

[11] H.-W. Shen, *Isosurface extraction in time-varying fields using a temporal hierarchical index tree*, in D. E. a. H. H. a. H. Rushmeier, ed., *IEEE visualization '98*, IEEE, October 1998, pp. 159-166.

[12] P. Shirley, A. Tuchman. *A polygonal approximation to direct scalar volume rendering.* Volume Visualization Workshop, 1990, pp. 63-70.

[13] P. M. Sutton and C. D. Hansen, *Isosurface extraction in time-varying fields using a temporal branch-on-need tree (t-bon)*, in D. E. a. M. G. a. B. Hamann, ed., *IEEE Visualization '99*, IEEE, San Francisco, California, October 1999, pp. 147-154.

[14] C. Weigle and D. C. Banks, *Complex-valued contour meshing*, in R. Y. a. G. M. Nielson, ed., *IEEE Visualization '96*, IEEE, October 1996, pp. 173-180.

[15] C. Weigle and D. C. Banks, *Extracting iso-valued features in 4-dimensional scalar fields*, *1998 Volume Visualization Symposium*, IEEE, October 1998, pp. 103-110.

[16] J. Wilhelms and A. V. Gelder, *Multi-dimensional trees for controlled volume rendering and compression*, in A. K. a. W. Krueger, ed., *1994 Symposium on Volume Visualization*, ACM SIGGRAPH, October 1994, pp. 27-34.

[17] J. Wilhelms and A. V. Gelder, *Octrees for faster isosurface generation*, ACM Transactions on Graphics, 11 (July 1992), pp. 201-227.