Extracting Iso-valued Features in 4-dimensional Scalar Fields

*Chris Weigle and **David C. Banks

*University of North Carolina at Chapel Hill
**Mississippi State University

Abstract
Isosurfaces are an important tool for finding features in 3D scalar data. This paper describes how recursive contour meshing is applied to extract similar features in 4-dimensional space. In the case of time-varying isosurfaces f(x, y, ,z, t) = c, the technique constructs a solid mesh for the isosurface that sweeps a volume in space-time. An instance of an isosurface at a particular time results from applying a second constraint against this volume. The envelope defined by the time-varying isosurface can be captured in a similar way: when a time-varying isosurface f=c reaches is maximum extent, the function's partial derivative with respect to time must be zero. This second constraint and produces a surface containing the extrema of the isosurfaces. Multi-resolution models and inter-penetrating blobby objects and can also be extracted from 4-dimensional representations.

1 Introduction

Time-varying scalar fields arise in many engineering datasets. Pressure, temperature, electric field strength, magnetic field strength, velocity magnitude, and vorticity magnitude are examples of scalar quantities that may be analyzed on a 3D grid produced by a numerical simulation. An isosurface at time $t=t_0$, defined by the points satisfying $f(x, y, z, t_0) = C$, is a helpful visualization tool for analyzing the 3D data. In a simulation of a time-varying problem, the isosurface typically changes shape from one time step to the next. If the solutions have been calculated at widely-spaced values t_0, t_1, ..., t_k, an animation of the isosurfaces at those time-steps will be jerky rather than smooth.

*Department of Computer Science
weigle@cs.unc.edu
**Department of Computer Science
300 Butler Hall, Mail Stop 9637
Mississippi State, MS 39762
banks@cs.msstate.edu

One way to improve the animation is to re-compute the solution grids at a finer time-resolution and store these results. But re-computing solutions might be very slow for a complicated field simulation, and storing the resulting grids may present an excessive burden on system memory (and even on external storage). A second approach is to interpolate the solution values on the grid at closely-spaced values of t between time steps and then to extract an isosurface for each interpolated grid. A changing grid poses problems for this approach, since the correspondence between grid-points at successive time steps may not be one-to-one, and thus one value may be interpolated to many.

Instead of interpolating grid values point-wise between two time steps, one can treat the static or dynamic grid as a single 4-dimensional entity and extract isosurfaces from it. How can this be accomplished?

Many techniques exist that triangulate the isosurface $f=C$ that arises when a scalar-valued function f is evaluated at points on a grid [Allgower] [Bloomenthal] [Lorensen] [Wyvill]. These techniques do not extend easily to multiple constraints in large dimensions. A solution to the more general problem of triangulating the contour that satisfies k constraints in n-dimensions was presented by Weigle and Banks [Weigle]. The algorithm – recursive contour meshing – triangulates contours in arbitrary dimensions.

Recursive contour meshing consists of routines to contour an n-simplex into a set of (n-1)-simplexes. When applied recursively, these routines contour an n-simplex against constraint functions f_k to produce (n-k)-simplexes. Given a tiling of n-space by n-simplexes, the (n-k)-simplexes produced by recursive contour meshing yield the contour which satisfies the Boolean intersection of the functions

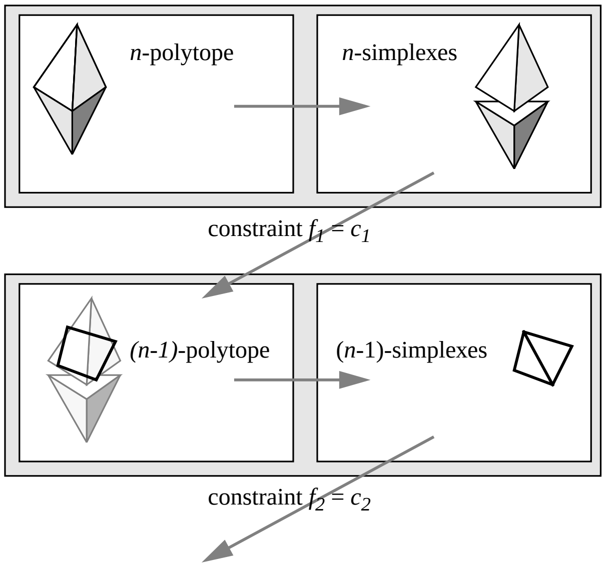


Figure 1. Recursive contour meshing applies a constraint to an n-simplex to produce an (n-1)-polytope. Splitting the polytope into simplexes permits the algorithm to apply another constraint.

f_i=C_i. The key to the recursive algorithm is that it applies a constraint to an *n*-simplex to produce a connected star-shaped polytope that is easily triangulated (with the addition of a midpoint) into (*n*-1)-simplexes. The resulting simplexes can be contoured by another constraint by applying the same scheme recursively (figure 1).

We can treat time-varying scalar fields as 4-dimensional (x, y, z, t)-space by considering time to describe the fourth, temporal axis. To find an isosurface at a particular time step in the data, we let f_1 be the scalar value at each point in space-time and f_2=t be the temporal value at each point. Since n=4 and k=2, the resulting (n-k)-simplexes will just be triangles. The first pass, contouring the space-time grid against f_1, produces the volume swept out by the isosurface over time. Contouring this volume against f_2=t in the second pass gives the desired time step. The volume resulting from the first pass can be stored as a set of 3-simplexes in 4-dimensional space. The second pass (using successive values of f_2) will yield an isosurface at each desired time step, thereby interpolating between isosurfaces.

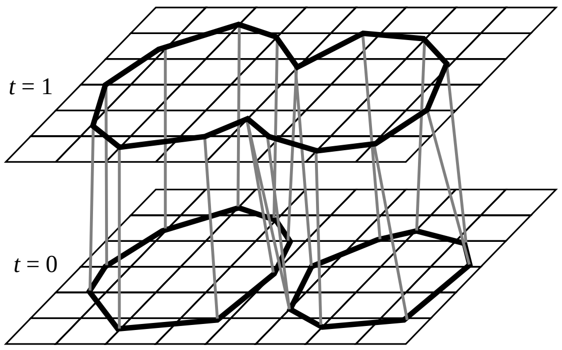


Figure 2. The contour curves at t=0 and t=1 are easy to compute, but difficult to connect together from disjoint planes. Using a 3D grid makes the connections simpler to determine. Two isosurfaces at different time values of 3D grids demonstrate similar characteristics.

Note that the isovolume defined by f_1=C is the swept volume of the isosurface over time; recursive contour meshing can be employed to find the envelope of the time-varying surface as well as individual isosurfaces contained within it. Section 4 describes how to do this.

2 Triangulating Isosurfaces in Isovolumes

2.1 Isosurfaces of 3D Scalar Data: ($\mathbb{R}^2$, t)

Consider the contouring of an isosurface within a 3D grid. Many techniques have been developed which begin by collecting contour-curves from 2D slices of the 3D data (perhaps from axially aligned slices where the *t* coordinate is constrained to a single value per slice) [Fuchs] [Johnstone]. These techniques join the curves by creating triangles between vertices from curves in adjacent slices. Such triangulations can be difficult to construct for a variety of reasons; changes in mesh structure, changes in topology (figure 2), and changes in geometry can lead to ambiguities when connecting vertices between slices [Fuchs].

The first case, change in structure, can be solved by “nearest-neighbor” techniques where the vertices in one slice are connected to the nearest vertices in the next. The next two cases, changes in topology and geometry, are much harder to solve. There are many possible triangulations, but there is no way to determine correctness without additional data.

Typically solutions to such problems involve picking the minimal surface triangulation [Johnstone]. Contouring individual slices of a 2-dimensional time-varying function may be an appropriate technique if the time steps are separated by long intervals.

Although research still continues into contour-slicing techniques for visualization [Johnstone], a shift has been made towards algorithms which look at the 3D data as a whole [Allgower] [Bloomenthal] [Lorensen] [Wyvill]. Treating the data as a volume allows these methods to find the connectivity along the third axis (the axis along which the slices span space) simply and efficiently. The triangulations computed via these 3D techniques are usually acceptable, if not always correct [Nielson] [Hill], if the data resolution is high enough.

What makes interpolation so easy between grid points, but so difficult between isocurves? As long as the grid points are regularly spaced, a reconstruction filter in the *t*-direction is simple to produce. For linear interpolation between grid points (*x, y, t*) and (*x, y, t*+1), a tent-filter suffices. The irregularly sampled points on the isocurves cannot be filtered with such a simple scheme.

2.2 Isovolumes of 4D Scalar Data: ($\mathbb{R}^3$, t)

Just as 2D contour-slicing-plus-reconstruction is not ideal for (*x, y, t*) data, applying a 3D contouring-plus-reconstruction to time-varying (*x, y, z, t*) data is an imperfect solution. The application of such a technique is possible when each time step is treated as a separate entity (just as each contour-slice was treated separately), but in doing so we forfeit connectivity information between time slices. The reconstruction of this lost information is a daunting task, but would allow for the inspection of contours at intermediate time steps or envelopes of swept surfaces. Typically such information is simply not available to the user without re-sampling the data at the desired time steps.

One way to recover the data is by interpolation at grid points between existing time steps. For example, to find the isosurface $f(x, y, z, 0.5) = 0$, one might average the function $f(x, y, z, 0)$ and $f(x, y, z, 1)$ on the underlying 3D grid and extract

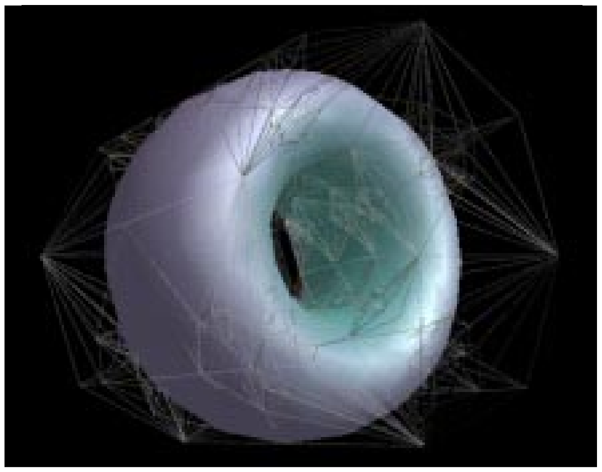
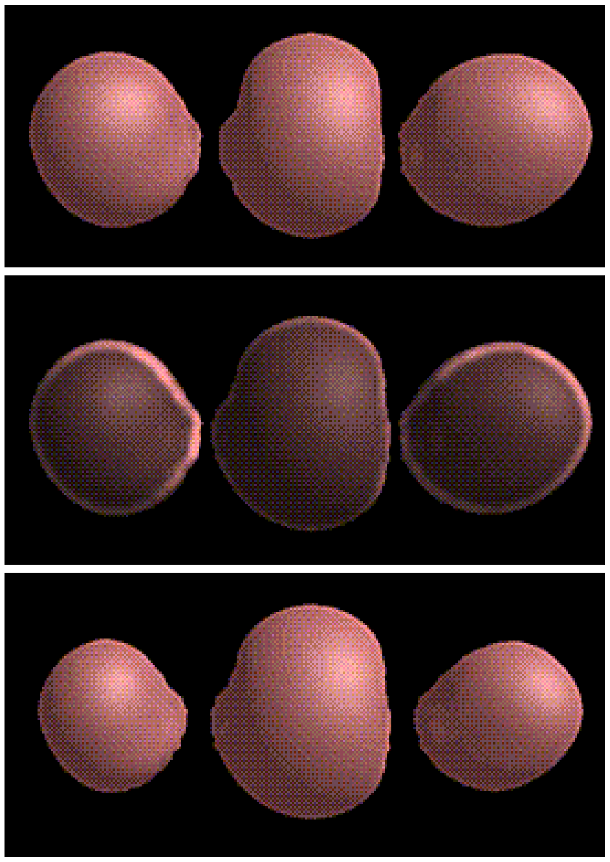


Figure 3. Above: surface satisfying 2 constraints in 4-dimensional space. 3-simplexes (wire frames) result from first constraint. The surface within them satisfies the second constraint also. Below: the volume swept by a time-varying isosurface (middle) fills the space between the isosurfaces (top and bottom) at successive time steps.



the isosurface from it. Although this allows the desired contour to be triangulated, it requires an expensive interpolation step. The interpolation might be restricted to only the cells that are flagged as containing the isosurface at either time step, but

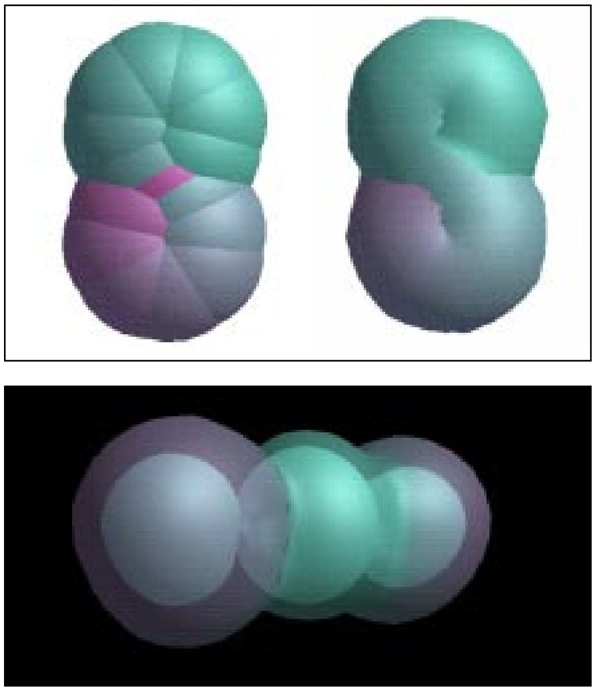


Figure 4. Top left: individual sphere-shaped isosurfaces in a time varying scalar field. Sampled at discrete time steps, these spheres approximate a figure-eight shape. Top right: contour-meshing the envelope in (*x, y, z, t*)-space yields a more complete surface and captures discontinuities, cusps, and self intersections that arise from the projection to (*x, y, z*)-space. Bottom: one frame of an animation of electric field strength of a dipole-antenna simulation (data courtesy of J. Beggs). The isosurface lies within a bounding envelope.

even this optimization retains a considerable amount of “fat” (entire grid cells) around the “meat” (the swept surface within the cells).

Promoting time-varying scalar data to 4D and applying recursive contour meshing is an effective way to solve the problem of locating isosurfaces of a time-varying function. By promoting the 3D time-varying data to 4-dimensional (*x, y, z, t*)-space, we can first contour the isovolume $f_1(x, y, z, t) = 0$ using 3-simplexes. To produce a surface for a particular time step c_t, we apply a second pass where $f_2(x, y, z, t) = t - c_t = 0$. Figure 3 shows an example of this contouring scheme, with a particular isosurface $f_1 = f_2 = 0$ rendered together with the cells of the isovolume $f_1 = 0$ that contain it. The lower portion of the figure shows the volumetric layer swept by an isosurface (in this case, a surface of constant electric field strength) over a short interval of time, together with isosurfaces at the two values that bound the time interval.

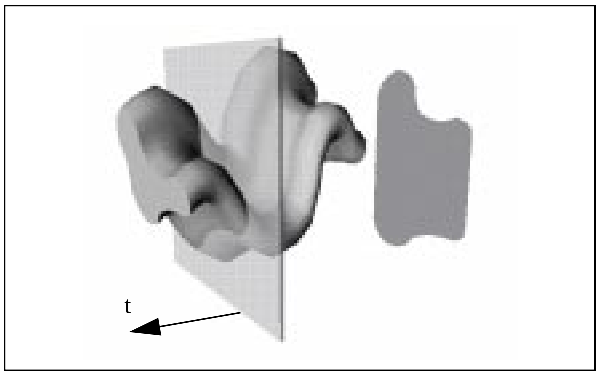


Figure 5. A time-varying 2D contour sweeps out surface in (x, y, t)-space. The silhouette of the surface (projected down the *t*-axis) forms the envelope of the swept curve.

3 Triangulating Envelopes

Consider a time-varying scalar function *f*(*x, y, z, t*). If the isosurfaces *f*(*x, y, z, t*) = 0 of a time-varying scalar function are all projected into (*x, y, z*)-space, they occupy a volume called a swept volume. The boundary of this volume is called its envelope. The envelope defines the minimum extent of space necessary to contain all of the time-varying isosurfaces. The envelope may be used to visualize, for example, the region of space where the temperature ever reaches a certain threshold, or where the magnitude of an electric field ever reaches a specified amount.

To produce an envelope of the swept isosurface *f*=0, one could extract an isosurface at each of the computed (perhaps integer-valued) time steps, then display them all simultaneously. This is an inefficient and inelegant solution to the problem. It is inefficient because most points on the isosurfaces will lie *within* (but not *on*) the envelope. It is inelegant because the in-between time steps are not smoothly interpolated; that is, the swept surface is “reconstructed” from the samples by convolving them with a Dirac delta function (rather than with a better filter) before projecting them to 3D space. See figure 4 (top).

Schroeder takes a different approach [Schroeder]. Rather than reconstructing the swept surface in space-time and projecting it to 3D (*x, y, z*)-space, he projects a distance function $d(x, y, z, t)$ onto 3D space and reconstructs the envelope near d=0. The distance function results from iteratively moving an implicit model through a spatial grid. The distance function is non-negative, so extracting the envelope at d=0 presents problems for simple root-finding schemes.

Wang describes how critical curves on moving geometry sweep out the envelope between the end-points $t_{initial}$ and t_{final} of the time interval [Wang]. Their technique exploits an explicit coordinate transformation function that describes the motion of an object.They observe that a point on the envelope satisfies

$$\frac{\partial f}{\partial t} = 0$$

This observation invites a straight-forward application of recursive contour meshing to locate an envelope. To see why the equation captures envelopes, consider a time-varying 2D function $f(x, y, t) = 0$. At a given slice $t = t_0$, the level set of $f(x, y, t_0) = 0$ forms a curve. The curves sweep out a surface in (x, y, t)-space. When viewed down the t-axis, the surface's silhouette corresponds to the envelope of the time-varying level sets. The view direction (down the t-axis) grazes a point on the silhouette, meaning the surface normal is perpendicular to the t-axis. But the surface normal is ∇f, so we see that

$$\nabla f \cdot (0, 0, 1) = \partial f/\partial t = 0$$

in agreement with the previous equation. Figure 5 illustrates the situation in (x, y, t)-space.

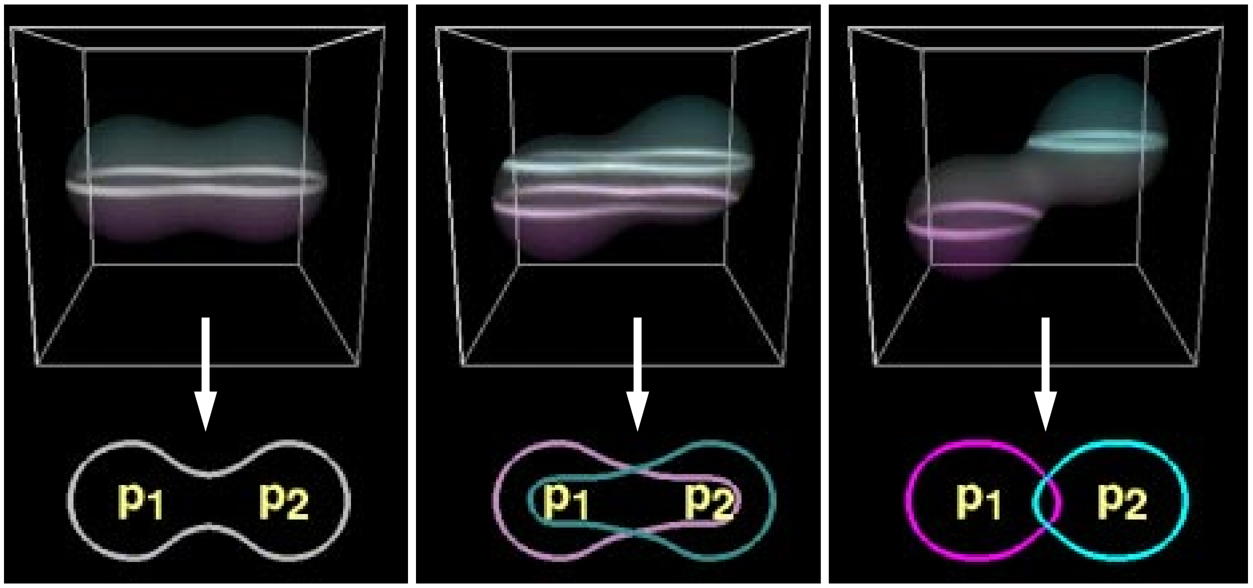
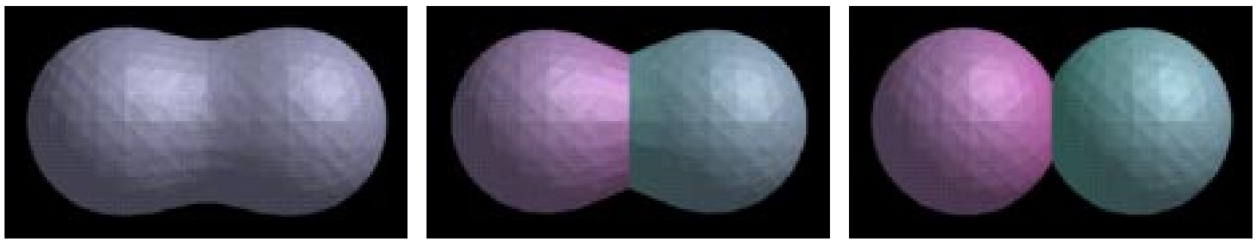


Figure 6. Blobby objects blend together when they are close. This behavior can be avoided by making them move apart in a higher dimensional space, but then projecting them in the direction of their separating motion. Above: three successive images of blobby curves in 2-space arise from slicing their parent blobby surfaces in 3-space. The slices become disjoint when the blobs move apart in the vertical direction, and the isocurves begin to overlap transversely. Below: analagous situation one dimension higher. Three successive blobby surfaces in 3-space arise from separating their parent blobby volumes in 4-space, slicing the iso-surfaces, then projecting to 3D.



The envelope of a 3D time-varying isosurface $f_1=0$ can be found in the same way. We use a second constraint, $f_2(x, y, z, t) = \frac{\partial f_1}{\partial t} = 0$ to locate the points on the isosurfaces that also lie on the envelope. The key advantages of this method are that (1) the envelope and all the isosurfaces at different time steps are latent in a single 4-dimensional data structure (the 3-simplexes that define the isovolume $f_1=0$), and (2) that the time steps are filtered in space-time before projecting to 3D instead of projected then filtered. Figure 4 shows the result of applying this technique to time-varying data from a computational field simulation of electromagnetic waves.

4 Intersecting Blobs; Mesh Reduction

An isosurface can be promoted to a higher dimension to produce an interesting effect. Consider a pair of point-density functions in three dimensions. An isosurface of constant density produces two disconnected components until the seed points become sufficiently close, causing the surfaces to blend together.

But if the seed points are separated in a fourth coordinate direction, the isovolume can be sliced and projected to three dimensions, producing intersecting (rather than merged) blobs. This process is illustrated in figure 6, first for a 2D projection of intersecting curves plane that result from slices of an isosurface as the seed points move apart in the vertical (z) direction, and secondly for a 3D projection of intersecting surfaces that result from slices of an isovolume as the seed points move apart in the w-direction. We are investigating ways to use such a technique to model implicitly defined objects with controlled blending.

Multidimensional isosurface extraction permits a simple way to specify shapes with different resolutions. Suppose the scalar field $f(x, y, z, 1)$ is shrunk to produce another scalar field

$$f(x, y, z, 2) = f(2x, 2y, 2z, 1).$$

The isovolume $f(x, y, z, w) = 0$ can be sliced at w=1 or at w=2 to produce isosurfaces at two different levels of detail on the same grid. The smaller one (at w=2) has fewer polygons than the larger. If all of the vertices in the smaller one are multiplied by the same factor of 2, the isosurfaces nearly coincide but possess different levels of geometric detail. Figure 7 illustrates the result of this process, where the underlying function satisfies

$$f(x, y, z, n) = f(2^n x, 2^n y, 2^n z, 1),$$

and the vertices of an isosurface defined by f=0, $w=w_0$ are scaled by a factor of 2^n.

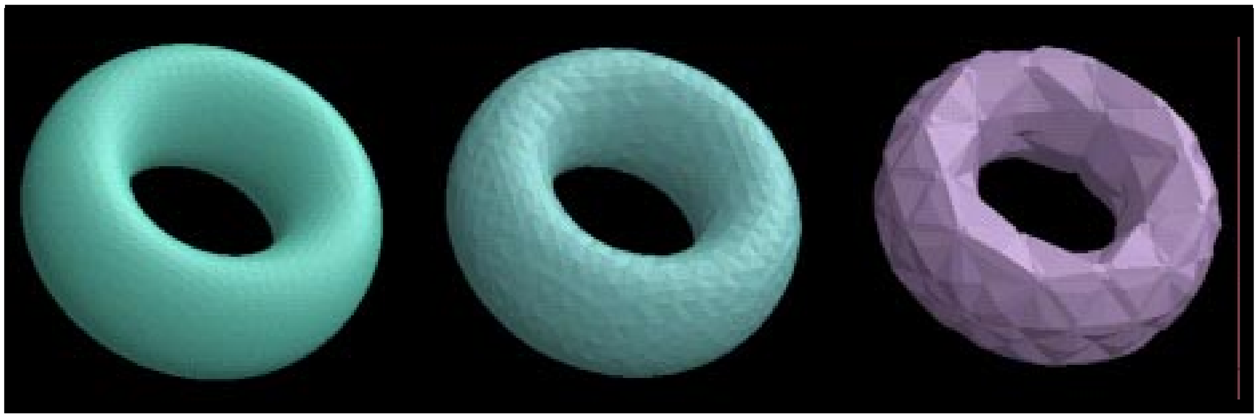


Figure 7. A family of isosurfaces at different resolutions constitutes a single volumetric representation of a torus. Different slices of the isovolume $f(x, y, z, w)$=0 in 4-space are projected to 3-dimensional (x, y, z) coordinates and scaled to produce polygonal meshes with varying complexities. The slices are taken from low values of w (on left) to high values (on right).

5 Performance

We implemented the recursive contour-meshing technique on time-varying data from an electromagnetic field simulation [Beggs]. We computed isosurfaces from a grid with 40×40×40×36 = 2,304,000 samples (36 time steps). The isosurfaces were computed using a fixed constraint $f_1=C$ in 4-dimensional space-time, and then repeatedly applying a second constraint $f_2 = t$. We super-sampled the time axis by a factor of ten to produce a smooth animation. Each 4-cell in the grid yielded between 3 and 4 triangles on average. The isosurfaces at interpolated values of t yield more triangles that the isosurfaces generated at t-values on the 4D grid. The reason is that we decompose the interior of each 4-cell into 192 4-simplexes, several of which may contain the isosurface.

On a Silicon Graphics Onyx with RealityEngine graphics with a MIPS R10000 195Mhz processor, we were able to generated a single triangular mesh in approximately 5 minutes for the 40×40×40×36 electric field. However, once the isovolume was constructed, finding an isosurface within that volume at a subsequent time step required less time, on the order of two minutes. There is still room for performance improvement: simplicial decomposition can be improved by a factor of 8 (using other subdivision schemes) and the exhaustive traversal of all the 4-cells can be improved by using a hierarchical data structure [Bloomenthal], perhaps by a factor of 4.

6 Conclusions

Time-varying isosurfaces provide a useful tool for visualizing the behavior of an unsteady 3D scalar field. We have shown how recursive contour meshing can interpolate and triangulate such time-varying surfaces by extracting them from the isovolume $f(x, y, z, t) = C$, rather than by (1) interpolating grid values pointwise between successive time steps and then (2) re-triangulating new isosurfaces at interpolated time steps. This allows us to efficiently construct smooth animations.

The envelope of the time-varying isosurface is the boundary of the projection of the isovolume in the t-direction. Using a partial derivative as a constraint on this isovolume allows recursive contour meshing to triangulate the envelope of the moving isosurface. This envelope reveals the global behavior of the isosurface over time.

Implicitly-defined models can be promoted to a higher dimension, sliced, and project to control the extent to which they merge together, and implicit functions can be promoted to a higher dimension and scaled in order to construct isosurfaces of varying geometric complexity.

These techniques are not always very practical in terms of their computational performance (because multidimensional grids are often quite large), but the general approach of extracting isovolumes and the isosurfaces they contain permits us to address a wide variety of different problems in graphics and visualization, and to do so in a simple and consistent way.

7 References

[Allgower] Allgower, Eugene and S. Gnutzmann, “Simplicial pivoting for mesh generation of implicitly defined surfaces,” *Computer Aided Geometric Design* (Netherlands), vol. 8, no. 4, Oct 1991, pp. 305-325.

[Beggs] J. H. Beggs, R. J. Luebbers, K. S. Yee and K. S. Kunz, “Finite-Difference Time-Domain Implementation of Surface Impedance Boundary Conditions,” *IEEE Trans. Antennas Propagat*., vol. 40, Jan. 1992, pp. 49-56.

[Bloomenthal] Bloomenthal, Jules, “Polygonization of implicit surfaces,” *Computer Aided Geometric Design*, vol. 5, no. 4, Nov. 1988, pp. 341-355.

[Fuchs] Fuchs, Henry, Z. Kedem, and Sam Uselton. “Optimal surface reconstruction from planar contours,” *Communications of the ACM*, (October) 1977, pp. 693-702.

[Hill] Hill, Steve and Jonathan C. Roberts, “Surface models and the resolution of *n*-dimensional cell ambiguity,” *Graphics Gems V*. Boston: Academic Press, 1995, pp. 98-106.

[Johnstone] Johnstone, John and Kenneth Sloan, “Tensor product surfaces guided by minimal surface area triangulations,” *Proceedings of Visualization ‘95*. IEEE Computer Society Press, (October) 1995.

[Lorensen] Lorensen, William and H. E. Cline, “Marching cubes: a high resolution 3-D surface construction algorithm.” Proceedings of SIGGRAPH ‘87, in *Computer Graphics*, (July) 1987, pp 163-169.

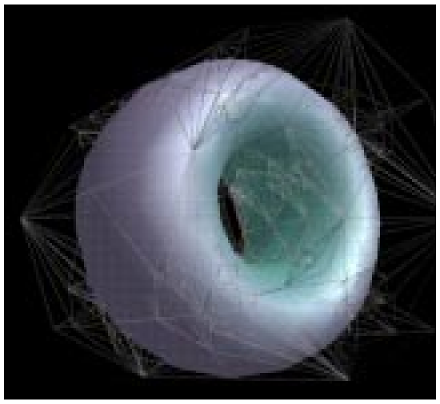
[Nielson] Nielson, G. M. and Bernd Hamann, “The asymptotic decider - resolving the ambiguity in marching cubes,” *Proceedings of Visualization ‘91*, IEEE Computer Society Press, (October) 1991, pp. 83-91.

[Schroeder] Schroeder, William, William Lorensen, and Steve Linthicum, “Implicit modeling of swept surfaces and volumes,” *Proceedings of Visualization ‘94*, IEEE Computer Society Press, (October) 1994, pp. 40-45.

[Wang] Wang, W., and K. Wang, “Geometric modeling for swept volume of moving solids,” *IEEE Computer Graphics and Applications*, (December) 1986, pp., 8-17.

[Weigle] Weigle, Chris, and David C. Banks. Complex-valued contour meshing. Proceedings of Visualization ‘96. IEEE Computer Society Press, (October) 1996, pp. 173-180.

[Wyvill] Wyvill, G., C. McPheeters, and B. Wyvill. “Data structures for soft objects,” Visual Computer, (August) 1986, pp. 227-234.



Weigle and Banks, “Extracting Iso-valued Features in 4-dimensional Scalar Fields“

CCW from upper-left. **Figure 3**. portion of an isovolume (wire frame) with an isosurface inside; volume swept by time-varying isosurface. **Figure 4**. Sphere-shaped isosurfaces sweeping a volume; the volume’s envelope; dipole-antenna’s envelope of iso field-strength. **Figure 6**. Blobby objects in higher dimensions projecting as interpenetrating objects. **Figure 7**. A family of isosurfaces at different resolutions within a single volumetric representation of a torus.