EUROGRAPHICS 2010 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (*2010*), *Number 2*

Isosurfaces Over Simplicial Partitions of Multiresolution Grids

Josiah Manson and Scott Schaefer

Texas A&M University

Abstract

We provide a simple method that extracts an isosurface that is manifold and intersection-free from a function over an arbitrary octree. Our method samples the function dual to minimal edges, faces, and cells, and we show how to position those samples to reconstruct sharp and thin features of the surface. Moreover, we describe an error metric designed to guide octree expansion such that flat regions of the function are tiled with fewer polygons than curved regions to create an adaptive polygonalization of the isosurface. We then show how to improve the quality of the triangulation by moving dual vertices to the isosurface and provide a topological test that guarantees we maintain the topology of the surface. While we describe our algorithm in terms of extracting surfaces from volumetric functions, we also show that our algorithm extends to generating manifold level sets of co-dimension 1 of functions of arbitrary dimension.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

Volumetric data comes from a variety of sources such as CT scanners or Magnetic Resonance Imagers (MRI). Volumetric data is also synthesized in numerical simulations such as computing electron densities around a molecule, pressure in a fluid dynamics simulation, or neutron densities in a nuclear reactor. In graphics, even surfaces are represented volumetrically through their implicit forms, which is useful when performing Constructive Solid Geometry (CSG) operations. However, exploring volumetric data can be difficult.

One way to visualize volumetric information is by computing a level set of the volumetric function $F : \mathbb{R}^3 \to \mathbb{R}$ at some value c. The set of points $\left\{p \in \mathbb{R}^3 | F(p) = c\right\}$ is called an *isosurface*. Notice that, without loss of generality, we can assume the isosurface is given by $F(p) = 0$ since the value of c can be subtracted from the function. Isosurfaces are natural ways to visualize the data with clearly defined borders between different volumetric regions. A skeleton in a CT scan or a shock wave in a fluid simulation, for example, have clear boundaries. Many surface reconstruction methods also produce volumetric functions that clearly define the interior and exterior of the reconstructed shape, and creating an isosurface to generate polygons is typically the last step in this process. Even without clear boundaries in the data, isosurfaces are often helpful visual aids as they are faster to draw and take less memory to store than volumetric data.

In particular, we examine the case where a function is known for all points in a bounded region. Because we only assume that the function is piecewise smooth and does not have to be a distance function, generating accurate isosurfaces can be challenging. The level set of $F(p)$ may contain very thin regions or sharp features that are commonly produced during CSG operations. Moreover, uniform sampling of $F(p)$ is an inefficient strategy for calculating isosurfaces, because the majority of samples are far from the surface. Therefore, we desire a method that creates a surface from an adaptive sampling of the volumetric function. Finally, the isosurface produced should be usable for other post-processing applications such as finite element analysis. These applications require that the isosurface is manifold to avoid anomalies not present in real-world objects, meaning that the surface must be locally equivalent to a disk in terms of its topology and have vertices positioned to avoid self-intersection.

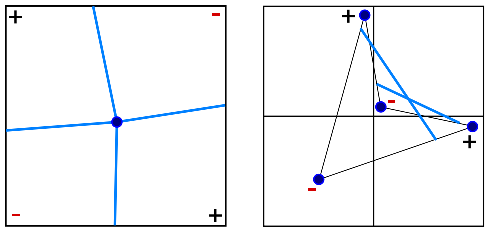


Figure 1: *Dual Contouring (left) creates a single vertex per cell and connects the vertices to adjacent cells sharing an edge with a sign change resulting in non-manifold geometry whenever an ambiguous sign configuration is encountered. Dual Marching Cube (right) can create self-intersecting surfaces whenever the dual cells are non-convex. In both pictures, the surface generated is depicted by a thick blue line.*

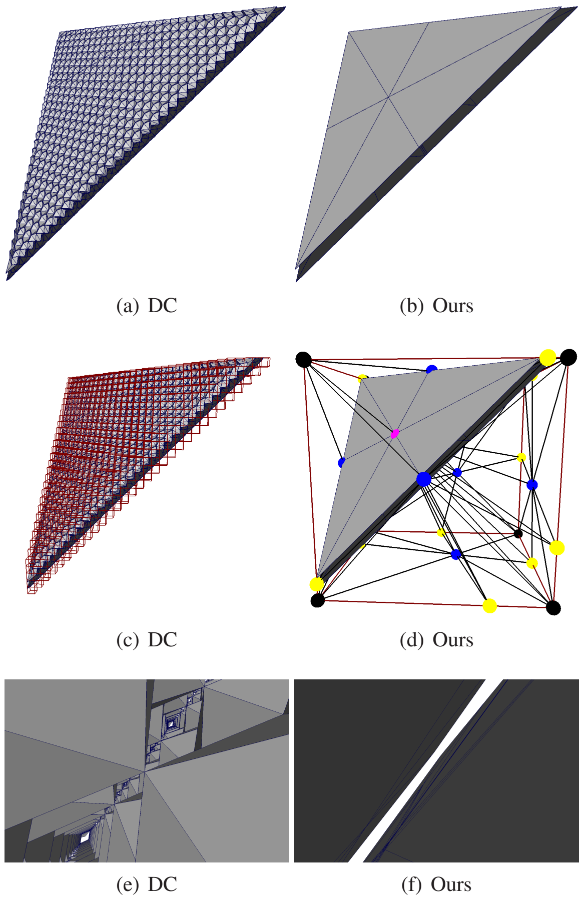


Figure 2: *Two thin sheets contoured with DC and our method (top), but our method uses a single cell whereas DC uses 32 cells on a side. In the middle, the sampling resolution and the partitions of space are shown. Our method adapts its partitioning of the cell to reproduce the thin sheet exactly. The bottom shows a view in between the two sheets to illustrate the non-manifold geometry created by DC.*

Contributions

Our method extracts an isosurface using an octree partitioning of space. In particular, we provide

- a surface extraction method that produces a manifold surface from an arbitrary octree,
- a strategy for placing dual vertices to reproduce sharp features of the isosurface or thin features beyond the sampling resolution of the octree,
- a method for reducing the number of polygons in the isosurface and improving the quality of the resulting surface by placing dual vertices directly on the isosurface,
- a simple topological safety test that guarantees that we still produce a manifold surface and maintain the topology of the original shape after modifying the dual vertices to reduce the polygon count.

Finally, though our description concentrates on 3D isosurfaces, our method extends to arbitrary dimension.

2. Related Work

Early contouring algorithms like Marching Cubes (MC) [LC87] sample $F(p)$ over a regular grid that partitions space. For each cube in the grid, MC determines the vertex positions using linear interpolation along its edges. The topology of the surface is indexed from a table solely by the sign configuration (inside/outside) of the corner samples. However, the sign configuration does not uniquely determine the topology inside of a cell. This ambiguity can be overcome by reconstructing the topology induced by trilinear interpolation [Nie03]. Marching Tetrahedra (MT) [DK91] also removes topological ambiguities by uniformly partitioning space with tetrahedra rather than cubes, which has the additional benefit of using a smaller table of sign configurations than MC. Semi-regular grids of tetrahedra have also been used to find surfaces for data where each sample is either inside, outside, or unknown [Nie08, NL09].

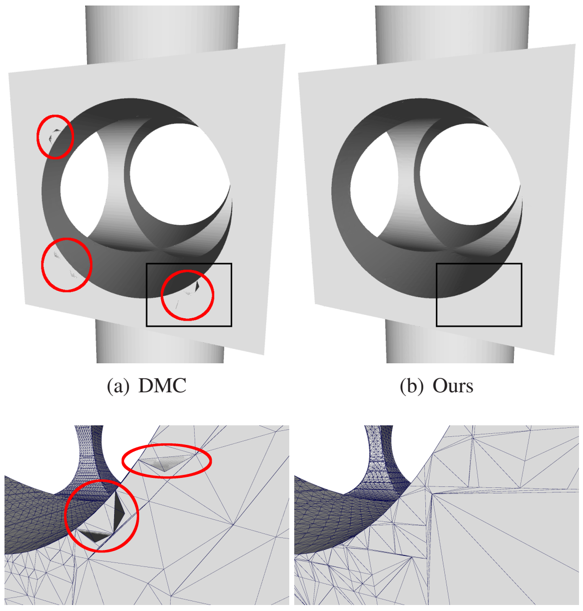
Although sampling over a regular grid is simple and produces good surfaces, uniform sampling is inefficient because surfaces usually intersect only a small percentage of the cells in a grid. Hierarchical spatial partitions adaptively sample the function near the isosurface or use geometric criteria to place fewer samples in flat regions of the isosurface. However, hierarchical representations pose an additional set of challenges for isosurfacing methods as the interface between adaptive cells must be handled to avoid cracks and other artifacts that may arise. The first step of creating an adaptive partition of space was taken in early works [Blo88, MS91]. With a few exceptions, such as [MW97, GK04], researchers use octrees to partition space. Some methods extend MC to octrees through crack-patching between multi-resolution cells [WKE99, VT01] or bending high-resolution vertices to adjacent, low-resolution contours [SFYC96]. Another technique [ZCK97] uses a ROAM-like [DWS*97] algorithm to adaptively partition space into tetrahedra for isosurface extraction. However, this partitioning method constrains the tetrahedra's vertices to lie at midpoints of edges and produces a balanced tree. In contrast, we place no such restrictions on our octree decomposition and the vertices of our tetrahedra can move freely inside their cells to reproduce sharp features of the underlying isosurface.

Many of these early techniques place restrictions on the octree's adaptivity, and patching strategies often result in visual artifacts. Recent methods extract smooth, manifold surfaces from unrestricted octrees. One method [SGM04] creates crack-free tetrahedralizations by performing a Delaunay tetrahedralization of the octree over which MT can be run. More recently, an algorithm was developed [KKDH07] that ensures compatible contours across faces of octree cells using edge trees. In both of these methods, the vertices of the cells that partition space (the vertices of the octree) are fixed and surface vertices are constrained to edges of the partitions. Without freedom to place partition vertices it is impossible to reconstruct thin and sharp features.

When additional information about the function other than its value is available, it is possible to reproduce sharp features that are otherwise impossible to resolve. One approach to capturing sharp features replaces the function with a directional distance field [KBSS01]. However, most methods that reconstruct sharp features only assume knowledge of the gradient, and several methods exist that find sharp features in an adaptively sampled function. Dual Contouring (DC) [JLSW02, ZHK04] reproduces sharp features by placing surface vertices dual to octree cells. However, DC produces non-manifold, self-intersecting surfaces in ambiguous sign configurations as shown in Figures 1 (left) and 2. DC produces non-manifold topology because no more than one surface vertex is generated for each grid cell, but multiple sheets of surface may pass through a single grid cell and will intersect at that surface vertex.

Extensions of DC generate either manifold topology [SJW07] or an intersection-free surface [Ju06], but neither method produces surfaces that are both intersection-free and manifold. Although the partition of space described in Intersection Free Contouring on an Octree Grid (IFC) [Ju06] is similar to ours, IFC maintains the same topology as the DC surface and, hence, produces non-manifold surfaces in the same sign configurations that DC does, as shown in Figure 1. Volumetric data stored in KD-trees can also be contoured using an extension of DC [GK04], but the method has many of the same problems as DC. Cubical Marching Squares (CMS) [HWC*05] generates surfaces with sharp features from Hermite data sampled over an octree, but performs complicated intersection of areas on faces and intersection of volumes in cells.

Thin features of a surface are pieces of the surface


(a) DMC (b) Ours

(c) DMC (d) Ours

Figure 3: *A mechanical part created through CSG operations is contoured using Dual Marching Cubes (DMC) and our method. The top shows the surfaces without wireframe where the dark areas are caused by non-convex cells that generate folded-back, self-intersecting triangles. Below illustrates a zoom-in of the triangulation to clearly show the self-intersecting triangles.*

that are smaller than the sampling density. In the special case of a distance function, methods have been developed to detect the existence of thin features to guide subdivision [VKKM03, VKSM04] and thereby guarantee that the topology of the isosurface is reproduced. However, it is possible to actually reconstruct features that are smaller than the sampling resolution. Dual Marching Cubes (DMC) [SW04] creates a partition of space that is dual to the octree grid. DMC reproduces thin features by positioning vertices of the dual grid at the features of $F(p)$ (i.e. intersections between smooth regions of $F(p)$). DMC then contours the dual partition by applying the MC table to each partition cell. However, when reconstructing sharp features, DMC can generate self-intersecting surfaces because the dual grid may be non-convex, as shown in Figure 1 (right) and Figure 3. In contrast, our method uses an adaptive octree to generate a tetrahedral partition that not only reproduces sharp features, but also guarantees that tetrahedra cannot invert. Therefore, our surfaces cannot self-intersect.

3. Partitioning Space

We make no assumptions about $F(p)$ other than that the function is piecewise smooth and continuous so that a gra-

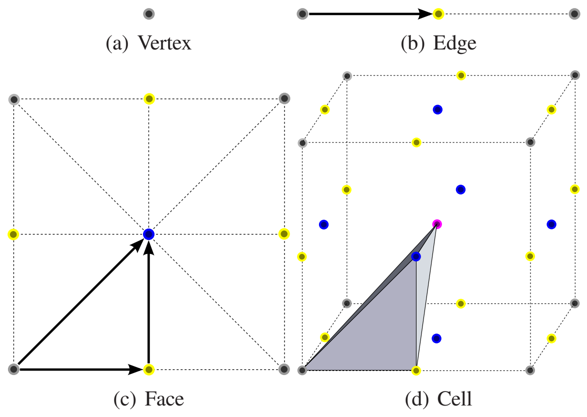


Figure 4: *Space is recursively decomposed into simplices. Starting from a single vertex in (a), we form a line segment by adding an edge vertex in (b). We add a face vertex to the line segment in (c) to form a triangle, from which we form a tetrahedron by adding a cube vertex in (d).*

dient is well-defined almost everywhere. Places where the gradient is not defined are sharp features of the function. Our method is related to DMC in that we partition $F(p)$ along its features to improve our piecewise linear approximation. We focus on accurately representing the function rather than the isosurface because an accurate function estimate produces an accurate isosurface from the resulting spatial partition. Although our method easily extends to extracting manifolds of co-dimension 1 from functions of arbitrary dimension, we restrict our discussions to assuming $F(p)$ is defined over $\mathbb{R}^3$ and that our isosurface is 2-manifold.

An octree is composed of cubes, faces, edges, and vertices, which we refer to as 3-cells, 2-cells, 1-cells, and 0-cells respectively. Given an octree, we place vertices dual to each minimal m-cell. A minimal m-cell is a cell of dimension m that does not contain any cells of dimension m as a subset. For example, a minimal cube is a leaf in the octree and a minimal face is the smaller of the two shared faces of adjacent leaf cubes.

For each minimal m-cell, we create a dual vertex $\bar{p} = <p, F(p)> \in \mathbb{R}^4$ with $p \in \mathbb{R}^3$ and constrain each vertex to lie within its cell (e.g. vertices dual to 3-cells are constrained to cubes, 2-cells to faces, 1-cells to edges, and 0-cells to the position of the 0-cell vertex). Notice that, for 0-cells, the position of the dual vertices coincides with vertices of the octree. For m-cells where $m > 0$ we enforce an additional constraint that vertices cannot lie on the boundary of their cell to avoid degenerate tetrahedra and T-intersections in the resulting isosurface. Hence we constrain these points to a cell of size $1-\varepsilon$ of their containing cell.

We use dual vertices to define a partition of space over which we extract the isosurface. Since dual vertices are free to move within their cells, we use this freedom to position

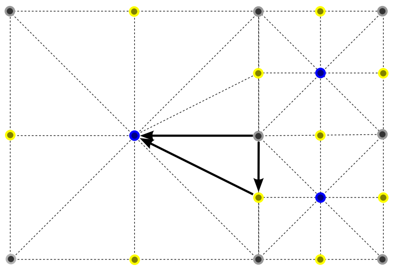


Figure 5: *Adaptive simplicial partitioning using minimal cells. The highlighted triangle shows how dual vertices to a 0-cell are connected to a 1-cell and finally to a 2-cell to form triangles.*

dual vertices such that linear interpolation between dual vertices approximates $F(p)$ well and reproduces sharp features that occur at the intersection of smooth pieces of the function. This strategy allows us to reproduce thin shapes with far less subdivision than methods such as MC or DC where function samples are restricted to the vertices of an octree.

To find sharp features in $F(p)$ we observe some properties of the function. First, $F(p)$ defines a four dimensional surface. Because $F(p)$ is piecewise smooth, $F(p)$ can be locally approximated by tangent hyperplanes except at sharp features. Second, sharp features occur at intersections of locally smooth regions of $F(p)$, and we can approximate these features using intersections of hyperplanes sampled locally around the features. We therefore place dual vertices at sharp features by minimizing their distance to the hyperplanes in the neighborhood of the vertex.

For each m-cell, we sample the function $F(p)$ at each octree vertex p_i on the boundary of the cell as well as the normal $\bar{n}_i = <\nabla F(p_i), -1>$. When $m < 3$, we restrict the gradient to the cell by projecting $\nabla F(p_i)$ onto the subspace defined by the cell. Each of the pairs $(\bar{p}_i, \bar{n}_i)$ forms a tangent plane and we place the vertex $\bar{x}$ dual to this cell at the position that minimizes the squared distance to each plane

$$\min_{\bar{x}} \sum_i (\bar{n}_i \cdot \bar{x} - \bar{n}_i \cdot \bar{p}_i)^2 \tag{1}$$

such that the spatial coordinates of $\bar{x}$ lie within its corresponding cell (scaled by $1-\varepsilon$). Lindstrom et al. [LS01] provides a simple method for minimizing such quadratics over box constraints.

Unfortunately, the function value returned by minimizing Equation 1 is often a poor estimate of the actual function value in regions where $F(p)$ has high curvature. Depending on if the function is locally convex or concave, the estimated function values will be consistently high or low. We have found that mixing consistently wrong function estimates with the value of $F(p)$ at vertices results in a dimpled surface and that, at the cost of an extra function evaluation off of the regular grid, using $<x, F(x)>$ as the dual vertex

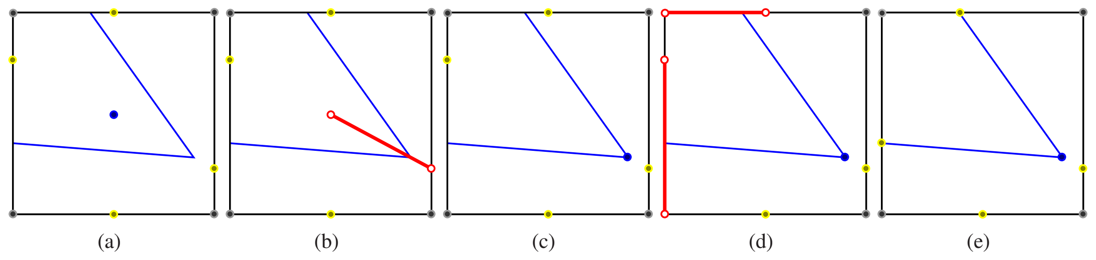


Figure 6: *Moving dual vertices to the isosurface improves the triangulation. Thick blue lines indicate the isosurface. Left to right: (a) original placement of dual vertices, (b) finding an appropriate vertex with opposite sign, (c) placement of face vertex on isosurface, (d) finding an end-point on the edge with opposite sign of edge vertex, and (e) placing edge vertices on isosurface.*

gives better results than using $\bar{x}$. Notice that this step requires evaluation of $F(p)$ away from a regular grid. For some functions (such as those produced by CSG operations), this requirement is trivial. When such functions are not available, this step can either be left out or an interpolant can be used to extend the function $F(p)$ between its samples at grid vertices.

We can also measure the error of minimized vertices by evaluating Equation 1. The error is larger in regions with high curvature than in regions with low curvature, because planes estimate nearly planar surfaces well and nonplanar surfaces poorly.

4. Isosurfacing

Given an octree containing dual vertices as described in Section 3, we provide a simple method for tetrahedralizing these cells. Using these tetrahedra, we then apply Marching Tetrahedra [DK91] to extract an isosurface.

Our tetrahedralization proceeds recursively in terms of the dimensionality of the dual vertices in the octree. We recursively build an m-simplex (simplex of dimension m) by connecting dual vertices of minimal ℓ-cells where $\ell \leq m$. The base case is a 0-simplex (a point) composed of the dual vertex for a 0-cell. Given a minimal i+1-cell with a dual vertex, we connect this vertex to all of the i-simplices defined on its boundary to build a set of i+1-simplices. Figure 4 shows an example of this recursive process over uniform cells of different dimensions. We start with every 0-cell as its own simplex. Edges are then created by adding edge-dual vertices to each 0-cell so that each edge is composed of two line segments. Face-dual simplices are added to each adjacent line segment to form triangles in each face, and, finally, cube-dual vertices are added to each triangle to form tetrahedra. Figure 5 shows how this process also works in the adaptive case where dual vertices corresponding to minimal cells are used to create the simplicial decomposition. Two minimal edges are shared between the large square and the smaller squares in the figure, which means that a total of four line segments are formed on the right boundary of the large square, which then form four triangles connecting the large square to the smaller squares.

As long as the dual vertices lie within their corresponding cells, this process cannot produce any inverted tetrahedra and creates a partition of space. This is because octree decompositions are convex and each dual vertex lies within the kernel (region of the cell visible by every point on its boundary) of the cell. Moreover, none of the tetrahedra will be degenerate since we constrain dual vertices so that they cannot lie on the boundary of their cell (dual vertices are constrained to lie in cells of size $1-\varepsilon$ times the original cell size). Also, as long as no samples are zero, the polygons produced by MT will be contained solely by the tetrahedra that generate them. Since our decomposition is one-to-one, our surface cannot intersect itself. Each edge in our surface will also be contained by exactly two polygons since tetrahedra share a common triangle face with adjacent tetrahedra. Finally, since vertices of the isosurface cannot lie at the end-points of the edges of the tetrahedra, our isosurface is guaranteed to be manifold and intersection-free.

4.1. Triangulation Improvement

As we have described the algorithms so far, the surface that is generated will approximate contours well. However, like MT, our isosurfaces contain numerous small, sliver triangles when vertices of the tetrahedra pass close to the isosurface. We can improve the triangulation using a method similar to Hall et al. [HW90] by placing the vertices that are dual to each cell directly on the isosurface. Triangles that become degenerate from this operation are then removed. Figure 9 shows an example of this process where many of the undesirable triangles are removed.

Our strategy is to move the dual vertices of each m-cell to the isosurface while keeping the dual vertex contained within its m-cell. Notice that we cannot use the value at the dual vertices to indicate how close the vertex is to the isosurface, nor does the magnitude of the gradient provide this information, because we have not placed any restrictions on the function $F(p)$. Furthermore, we want to preserve sharp features in the object when moving the dual vertices.

When moving dual vertices onto the surface, we want to maintain the accuracy of our approximation of the function. In the original partition of each cell, the dual vertex was

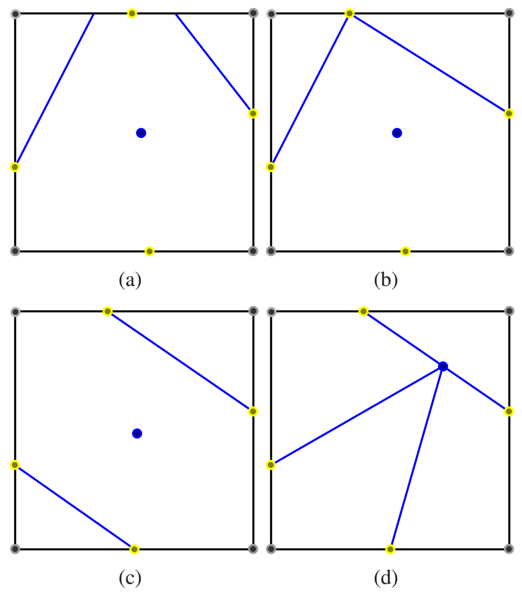


Figure 7: *Moving dual vertices onto the isosurface may change the topology for cells that do not contain a single sheet. Thick blue lines indicate the isosurface. (a) shows the original topology that is altered by moving an edge sample to the isosurface in (b). The original topology in (c) is also altered by moving the face sample to the isosurface in (d).*

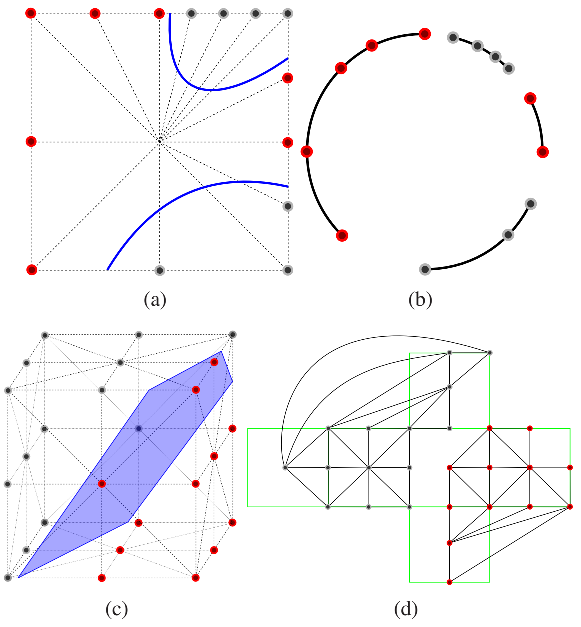


Figure 8: *Top: a multiresolution face (left) whose connectivity graph is shown (right) that fails the topological safety test. Bottom: a 3D cube (left) whose connectivity graph is shown (right) that passes the topological safety test.*

placed to minimize the squared distance to a set of local planes. We must find a direction to move the dual vertex in that introduces the least error and guarantees that we can move the dual vertex to the surface. To do so, we evaluate the error at each dual vertex in the boundary with opposite sign using Equation 1 and choose to move in the direction of the vertex that has the smallest error. We then perform a bisection algorithm between the dual vertex and the boundary vertex to compute the new position of the dual vertex x such that $F(x) = 0$ and x lies on the isosurface. For each m-cell, triangulation improvement depends only on vertices dual to i-cells where $i < m$. Hence, we proceed from higher dimensional cells to lower dimensional cells so that each vertex can be processed independently. This process is shown in Figure 6.

Notice that we must be careful when moving dual vertices as we may change the topology of the surface in some cells, which could result in non-manifold topology since we now allow zero isovalues. Figure 7 demonstrates this problem. Parts (a) and (b) show how moving a vertex dual to an edge can result in non-manifold geometry without a topological test. Parts (c) and (d) illustrate a similar problem with a vertex dual to a face that creates non-manifold geometry.

We provide a test that preserves the topology of the isosurface within each m-cell by detecting if the portion of the isosurface intersecting the m-cell is a topological disk and restricting vertex movement if the test fails. We reduce our topological safety test to counting connected components in a graph. Given an m-cell, we build a graph consisting of all of the edges of the simplices on the boundary of the cell that connect all dual vertices of dimension $i < m$.

Note that some cells in the octree form graphs that have special forms for which we can calculate connected components easily. For an edge, there are only two vertices on its boundary, so we simply test that the vertices have opposite sign. The graph formed by a face will be a ring with an even number of sign changes between vertices, because for any transition from negative to positive there must be a transition back. For faces, we can therefore check that there are exactly two sign changes along its boundary. For a cube, we use the Union-Find algorithm, which is a simple algorithm that takes nearly linear time to find connected components in a graph. Figure 8 shows this reduction applied to a multiresolution face and a cube.

Our topology test is exact in that we preserve the topology of the surface if and only if the test is satisfied. Assume that the topology test is satisfied. This means the boundary of the cell forms two disjoint vertex sets of inside/outside vertices and the boundary is a connected sheet. Since the dual vertex of the m-cell is connected to all of the dual vertices on the boundary, the value and position of the vertex are irrelevant

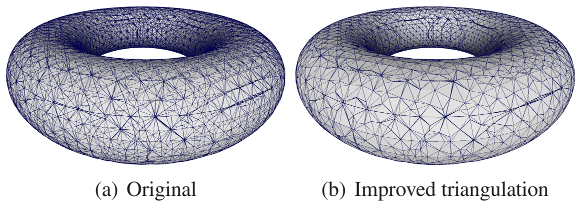


Figure 9: *Our method for improving surface triangulation reduces the number of triangles from* 44192 *on the left to* 14720 *on the right.*

as the contour through the cell is still a topological disk (the boundary is completely connected as a single sheet).

The converse statement is also true. Suppose we give a vertex, x, that is dual to an m-cell a value of zero by placing it on the isosurface. Then any portions of the contour on the boundary of the cell will form a sheet that passes through x, because x is connected to all of the vertices on the boundary of the cell. Since each sheet touches x, the sheets all touch each other and create a single surface. Placing the central dual vertex on the isosurface pinches the surface to a point and creates a non-manifold vertex when portions of the contour are not connected before moving the dual vertex. Even if the boundary does not contain any of the isosurface (all vertices have the same sign), the test still works, since placing the central dual vertex on the isosurface collapses a topological sphere to a point or creates an isolated point.

5. Implementation

The tetrahedralization in Section 4 requires that the minimal edges and minimal faces are enumerated to build the tetrahedra we use to generate the isosurface. Our implementation uses the recursive octree traversal of DC [JLSW02] to find the leaves in the octree that surround a minimal edge in the tree. Of the (up to) four cubes surrounding a minimal edge, the cube at the deepest depth will contain the minimal edge of the tree. This minimal edge creates two line segments corresponding to the octree vertices of the minimal edge and its dual vertex. Next, for each pair of face-adjacent cubes surrounding the minimal edge, we connect these two line segments to the vertex dual to the minimal face between these pairs of cubes to create two triangles for each minimal face. Finally, we connect each of these triangles to the vertex dual to the cube they are adjacent to, forming four tetrahedra for each face-adjacent cube. Our algorithm requires no auxiliary data structures beyond the octree itself and the dual vertices to efficiently construct our tetrahedralization for any octree. We then use a table-based method based on the sign configuration of each tetrahedron to contour the cell [DK91].

5.1. Octree Generation

While our method operates on arbitrary octrees, we provide a simple method to create an octree that conforms well to the

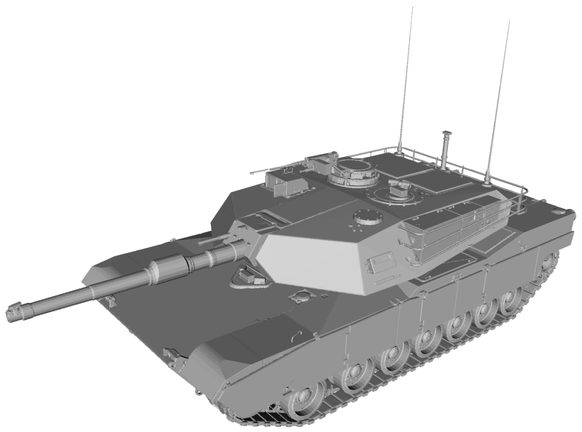


Figure 10: *An M1 Abrams tank reconstructed at depth 8 by our method is shown. Notice that high resolution details like the antenna are found correctly.*

isosurface using the error metric from Section 3. There are two classes of approaches to generating an octree for isosurface calculation. A bottom-up approach samples the function uniformly in a full octree and then prunes cells that do not contain a contour. In many applications, the function we contour is expensive to evaluate, which makes simple bottom-up octree calculation infeasible.

Therefore, we use a top-down contour-finding approach that adds cells to the tree by refining the sampling around detected contours. To begin, we use a uniform refinement down to a prescribed level to capture the coarse features of the function. Then, we analyze $F(p)$ at the dual vertices of each cube. We refine each cube that the dual vertices indicate the contour intersects until the sum of the errors from Equation 1 for each dual vertex in a cube is below a set threshold down to a prescribed maximum depth.

While this strategy does not detect all of the possible pieces of the isosurface, we note that this goal is impossible without further knowledge of the function $F(p)$. However, beginning with a uniform sampling does guarantee that we detect all features greater than the size of a cube in the uniform grid. These features, as well as any additional features detected during refinement, will be refined to the level of accuracy specified by the user. We use an error criterion simply to avoid refinement in regions of the function that are well approximated by a linear function such as flat regions of the isosurface. However, any octree generation method can be used with our technique.

6. Results

Our algorithm allows us to extract a manifold surface from arbitrary octrees and all of the examples in this paper were generated using octree decompositions. Figures 3 and 9 were generated with the octree generation method in Section 5.1;

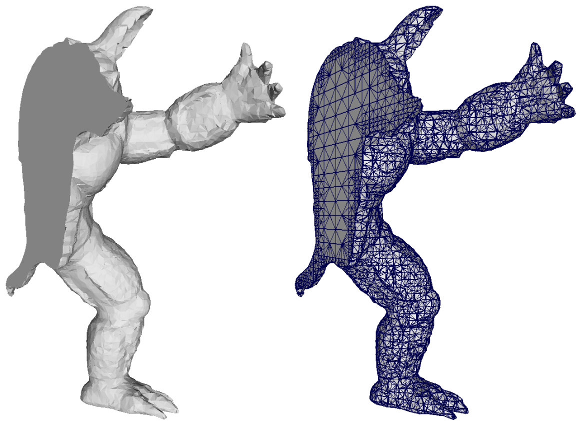


Figure 11: *The armadillo man cut in half by a CSG plane and reconstructed by our method. The adaptive nature of the surface is evident in the flat region.*

	Armadillo man	Mech. part	Lens
Depth	8	9	10
Ours	2.58s	4.81s	9.72s
Ours improv.	2.69s	6.80s	10.35s
DMC	1.85s	3.54s	6.42s
DC	1.35s	2.97s	5.99s

Table 1: *Times taken by our method with and without triangulation improvement, DMC, and DC for a variety of models. Each method uses the same octree sampling density. The armadillo man is shown cut in half in Figure 11, the mechanical part is shown in Figure 3, and the lens is defined by CSG intersection of two spheres.*

the octrees of Figures 10 and 11, however, were calculated from the signed distance function of a polygonal surface and refined only where polygons intersect the octree cells.

We can generate manifold surfaces over arbitrary octrees because our algorithm combines the strengths of primal and dual contouring methods. The primal octree grid ensures that cells are convex and that the contoured surface cannot self-intersect. The dual structure of the octree provides a topology for transitions between neighboring cells at different resolutions, and only constrains positions of points to remain in the cell that they are dual to. We use the extra degrees of freedom in placing vertices to find better approximations of the function being contoured so that we can recover sharp features. By forming a piecewise linear approximation of the function, we are also able to recover sharp features in the function itself that result in surface features that may be smaller than the grid resolution.

Given the same octree, our method essentially samples the function at one higher resolution than DC or DMC because of the extra samples on edges, faces, and cells in the octree.

	Ours	Ours improv.	DMC	DC
Time total	8.78s	8.19s	5.29s	3.78s
Time tree	4.02s	6.13s	3.41s	2.69s
Time extract	4.77s	2.06s	1.88s	1.09s
Triangles	3.63M	1.13M	1.16M	727k

Table 2: *Details are given for each method when calculating the surface of the tank shown in Figure 10. We refine the octree to depth 9 for DC and DMC, and to depth 8 for our method so that sampling densities are equal. Notice that our times are about twice as long as DMC and number of triangles are comparable to DMC.*

To make the comparison between methods fair, we calculate the surface in our method at a lower resolution than DC and DMC. Running times for our method, DC, and DMC are shown in Table 1 for a variety of functions. The fastest method is DC, because it minimizes three dimensional error functions to position surface vertices at sharp features, whereas DMC and our method minimize four dimensional error functions to position dual vertices with function values. DMC is approximately 10-20% slower than DC, and our method takes an additional 50% longer than DMC.

Table 2 shows a breakdown of the times for each method when reconstructing the tank shown in Figure 10. Although the octree for the tank was refined only around the surface, the antenna is of higher resolution than the sampling grid. DC only samples near the isosurface and is unable to detect small details like the antenna and railings of the tank. Our method and DMC, on the other hand, form a grid dual to the octree structure and detect the antenna of the tank even at depth 6. However, DMC produces several flaps of surface that fold back on themselves in the tank, whereas our method avoids such self-intersections. Contouring methods that partition space into tetrahedra are notorious for generating many triangles, but with triangulation improvement, our method generates fewer triangles than DMC does.

We focus our discussion on three dimensional isosurfaces, but our method extends to contouring functions of any dimension. Our method's construction and proofs of correctness are inductive, which means that we can apply our algorithm to higher dimensions. By contrast, a direct extension of MC to higher dimensions is complicated by the MC table. The MC table is difficult to compute, and has 2^{2^m} entries for dimension m, whereas the simplex table used in our method is easy to compute and is size 2^{m+1}. These higher dimensional isosurfaces may be useful for time-dependent simulations, surface reconstruction of moving objects, or robotics for path planning.

7. Future Work

We want to further improve the triangle quality in the surface generated by our method. Moving dual vertices to lie directly on the isosurface greatly improves the triangulation of isosurfaces, but some small and poor aspect ratio triangles still remain. These triangles are generated near 0-cell vertices that the isosurface passes close to because 0-cell vertices cannot move to ensure convexity of all d-cells in the octree. One possibility we would like to investigate is allowing 0-cell vertices to move. So long as we can ensure that higher-dimensional dual vertices remain in the kernel of their cell, our tiling of space is guaranteed to be convex and the surface will be manifold.

We would also like to improve octree generation. Currently, we use an error metric that is zero in flat regions and positive in curved regions of the function, but we do not directly measure the curvature of the surface. For example, $F(p) = p_x^2 - 1$ has a completely planar isosurface, but our current refinement criteria detects that the function is curved near the contour and refines the octree more than necessary. By more accurately determining the curvature of the surface, it will be possible to generate accurate surfaces with fewer triangles and using less memory. Alternatively, it may be possible to design a back-tracking algorithm that uses the samples generated during expansion to collapse parts of the tree that have been refined too far.

Acknowledgements
This work was supported by NSF grant CCF- 07024099 and DARPA grants HR0011-08-1-0014, HR0011-09-1-0042.

References

[Blo88] BLOOMENTHAL J.: Polygonization of implicit surfaces. *Comput. Aided Geom. Des. 5*, 4 (1988), 341–355.

[DK91] DOI A., KOIDE A.: An efficient method of triangulating equivalued surfaces by using tetrahedral cells. *IEICE Transactions Communication E74*, 1 (1991), 214–224.

[DWS*97] DUCHAINEAU M., WOLINSKY M., SIGETI D. E., MILLE M. C., ALDRICH C., MINEEV-WEINSTEIN M. B.: Roaming terrain: Real-time optimally adapting meshes. In *IEEE Visualization* (1997), pp. 81–88.

[GK04] GRESSA., KLEIN R.: Efficient representation and extraction of 2-manifold isosurfaces using kd-trees. *Graph. Models 66*, 6 (2004), 370–397.

[HW90] HALL M., WARREN J.: Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics and Applications 10*, 6 (1990), 33–42.

[HWC*05] HO C.-C., WU F.-C., CHEN B.-Y., CHUANG Y.-Y., OUHYOUNG M.: Cubical marching squares: Adaptive feature preserving surface extraction from volume data. *Computer Graphics Forum 24*, 3 (Sept. 2005), 537–546.

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of Hermite data. In *Proceedings of SIGGRAPH* (2002), pp. 339–346.

[Ju06] JU T.: Intersection-free contouring on an octree grid. Pacific Graphics Poster, 2006. http://www.cs.wustl.edu/~taoju/research/interfree_paper_final.pdf.

[KBSS01] KOBBELT L. P., BOTSCH M., SCHWANECKE U., SEIDEL H.-P.: Feature sensitive surface extraction from volume data. In *Proceedings of SIGGRAPH* (2001), pp. 57–66.

[KKDH07] KAZHDAN M., KLEIN A., DALAL K., HOPPE H.: Unconstrained isosurface extraction on arbitrary octrees. In *Proceedings of the Symposium on Geometry Processing* (2007), pp. 125–133.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 163–169.

[LS01] LINDSTROM P., SILVA C. T.: A memory insensitive technique for large model simplification. In *IEEE Visualization* (2001), pp. 121–126.

[MS91] MÜLLER H., STARK M.: Adaptive generation of surfaces in volume data. *The Visual Computer 9* (1991), 182–199.

[MW97] MÜLLER H., WEHLE M.: Visualization of implicit surfaces using adaptive tetrahedrizations. In *DAGSTUHL '97: Proceedings of the Conference on Scientific Visualization* (1997), pp. 243–250.

[Nie03] NIELSON G. M.: On marching cubes. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (2003), 283–297.

[Nie08] NIELSON G. M.: Dual marching tetrahedra: Contouring in the tetrahedronal environment. In *ISVC '08: Proceedings of the 4th International Symposium on Advances in Visual Computing* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 183–194.

[NL09] NIELSON G. M., LEE K.: Urban terrain data modeling with adaptive dual marching tetrahedra and feature detecting techniques. In *Proceedings of the Conference on Information Science, Technology and Applications* (2009), pp. 30–39.

[SFYC96] SHEKHAR R., FAYYAD E., YAGEL R., CORNHILL J. F.: Octree-based decimation of marching cubes surfaces. In *IEEE Visualization* (1996), pp. 335–342.

[SGM04] SCHROEDER W. J., GEVECI B., MALATERRE M.: Compatible triangulations of spatial decompositions. In *IEEE Visualization* (2004), pp. 211–218.

[SJW07] SCHAEFER S., JU T., WARREN J.: Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (2007), 610–619.

[SW04] SCHAEFER S., WARREN J.: Dual marching cubes: Primal contouring of dual grids. In *Proceedings of Pacific Graphics* (2004), pp. 70–76.

[VKKM03] VARADHAN G., KRISHNAN S., KIM Y. J., MANOCHA D.: Feature-sensitive subdivision and isosurface reconstruction. In *IEEE Visualization* (2003), pp. 99–106.

[VKSM04] VARADHAN G., KRISHNAN S., SRIRAM T., MANOCHA D.: Topology preserving surface extraction using adaptive subdivision. In *Proceedings of the Symposium on Geometry Processing* (2004), pp. 235–244.

[VT01] VELASCO F., TORRES J.: Cell octree: A new data structure for volume modeling and visualization. In *the VI Fall Workshop on Vision, Modeling and Visualization* (2001), pp. 151–158.

[WKE99] WESTERMANN R., KOBBELT L., ERTL T.: Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer 15* (1999), 100–111.

[ZCK97] ZHOU Y., CHEN B., KAUFMAN A.: Multiresolution tetrahedral framework for visualizing regular volume data. In *IEEE Visualization* (1997), pp. 135–142.

[ZHK04] ZHANG N., HONG W., KAUFMAN A.: Dual contouring with topology-preserving simplification using enhanced cell representation. In *IEEE Visualization* (2004), pp. 505–512.