Eurographics/ IEEE-VGTC Symposium on Visualization 2008
A. Vilanova, A. Telea, G. Scheuermann, and T. Möller
(Guest Editors)

Volume 27 (2008), Number 3

Quality Isosurface Mesh Generation Using an Extended Marching Cubes Lookup Table

Sundaresan Raman and Rephael Wenger †

Abstract

The Marching Cubes Algorithm may return degenerate, zero area isosurface triangles, and often returns isosurface triangles with small areas, edges or angles. We show how to avoid both problems using an extended Marching Cubes lookup table. As opposed to the conventional Marching Cubes lookup table, the extended lookup table differentiates scalar values equal to the isovalue from scalar values greater than the isovalue. The lookup table has $3^8 = 6561$ entries, based on three possible labels, '$-$' or '$=$' or '$+$', of each cube vertex. We present an algorithm based on this lookup table which returns an isosurface close to the Marching Cubes isosurface, but without any degenerate triangles or any small areas, edges or angles.

1. Introduction

A three dimensional *scalar field* is a continuous function f from $\mathbb{R}^3$ to $\mathbb{R}$. Given a regular grid sampling of a scalar field $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ and a scalar value σ, the Marching Cubes Algorithm [WMW86, LC87] constructs a piecewise linear approximation to the level set $\{x : f(x) = \sigma\}$. The piecewise linear approximation is called an *isosurface* and the value σ is called an *isovalue*.

The Marching Cubes Algorithm partitions all the grid vertices into two classes. Grid vertices with scalar value STRICTLY LESS than the isovalue are assigned a negative, '$-$', label, while vertices with scalar value GREATER THAN or EQUAL TO the isovalue are assigned a positive, '$+$', label. Note the arbitrary asymmetry in which vertices with scalar value equal to the isovalue are clustered with vertices whose scalar value is greater than the isovalue.

Since each of the eight cube vertices has two possible labels, each cube has $2^8 = 256$ possible configurations of '$+$' and '$-$' vertex labels. The Marching Cubes Algorithm uses an isosurface lookup table containing an isosurface patch for each configuration. For each cube, the algorithm retrieves an isosurface patch from the lookup table based on the cube's configuration of vertex labels. The isosurface patch is a set of triangles whose vertices lie on the edges of the cube. The algorithm positions each isosurface vertex on an edge based on the scalar values of the edge endpoints.

A grid edge is *bipolar* if one endpoint is labeled '$+$' while the other is labeled '$-$'. If some endpoint of a bipolar edge has scalar value equal to the isovalue, the Marching Cubes Algorithm will position an isosurface vertex at that endpoint. The Marching Cubes Algorithm uses the same lookup table entries for vertices with scalar value equal to the isovalue and scalar value strictly greater than the isovalue. Because of this, the algorithm sometimes creates triangles which have two or three isosurface vertices positioned at the same grid vertex.

For example, the grid cube in Figure 1 has one vertex, v, with scalar value four and all others with scalar values two. For isovalue four, vertex v has a '$+$' label and all other vertices have a '$-$' label. The isosurface patch for a configuration with a single '$+$' vertex is a single triangle. Marching Cubes will retrieve that triangle and then position all its vertices at grid vertex v, creating a degenerate isosurface triangle.

A post processing step could eliminate degenerate isosurface triangles by identifying isosurface vertices which have been mapped to the same grid vertex. Instead, we show how such degenerate triangles can be completely avoided by using a lookup table which distinguishes between scalar values equal to the isovalue and scalar values greater than the isovalue. Grid vertices are assigned a '$+$', '$-$' or '$=$' label, depending upon whether their scalar value is greater than, less than or equal to the isovalue. Each grid vertex has three

† Department of Computer Science and Engineering, The Ohio State U., Columbus, OH 43210, USA. Research partially supported by NSF grant CCF-0635008.

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

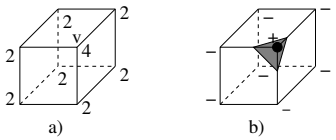


Figure 1: *a) Grid cube with one vertex with scalar value four and all others with scalar values two. The grid cube has a single '+' vertex when the isovalue is four. b) Lookup table isosurface patch for configuration with a single '+' vertex.*

possible labels, '+' or '−' or '='. The extended isosurface lookup table has $3^8 = 6561$ entries, one for each possible configuration of the labels on the cube vertices.

The algorithm for isosurface construction based on the extended isosurface lookup table is exactly the same as the Marching Cubes Algorithm. The challenge is in constructing the extended lookup table. Lachaud and Montanvert in [LM00] and independently Bhaniramka et. al. in [BWC00, BWC04] gave convex hull based algorithms for automatically generating isosurface lookup tables. We show how to modify those algorithms to generate extended isosurface lookup tables based on '+', '−' and '=' vertex labels.

Just as isosurface vertices located on grid vertices can create degenerate triangles, isosurface vertices located near grid vertices can create triangles with small areas, edges or angles. They can also create triangles with angles near 180°. Recently, Labelle and Shewchuk [LS07] presented an algorithm for constructing tetrahedral meshes with good dihedral angles. By combining their algorithm with the extended isosurface lookup table, we can modify Marching Cubes to produce an isosurface with good triangle angles. We determine where Marching Cubes will place the isosurface vertex on each bipolar grid edge. If that vertex is too close to a grid vertex, then we modify the scalar value of the grid vertex to exactly equal the isovalue. This forces the isosurface vertex to be "snapped" to the grid vertex. We run Marching Cubes on the modified scalar grid using the extended lookup table and then reposition the "snapped" isosurface vertices. Our algorithm is called SnapMC. Figure 2 contains an example of the output of our algorithm compared with Marching Cubes.

Labelle and Shewchuk's algorithm constructs a 3D mesh filling the volume bounded by an isosurface but can easily be modified to generate only an isosurface. However, Labelle and Shewchuk's algorithm requires converting the input regular grid into a body centered lattice partitioned into tetrahedra. In contrast, SnapMC constructs the isosurface directly on the original grid cubes. SnapMC also uses a slightly different technique for "snapping" isosurface vertices to grid vertices, modifying scalar values instead of warping the grid as in [LS07].

Processing degenerate or small triangles is time consuming, but not a major visualization problem. However, when isosurface triangulations are used for modeling and simulation, small or large triangle angles can create significant numerical problems [She02]. SnapMC produces an isosurface triangulation with guaranteed upper and lower bounds on the angles of any triangle in the triangulation.

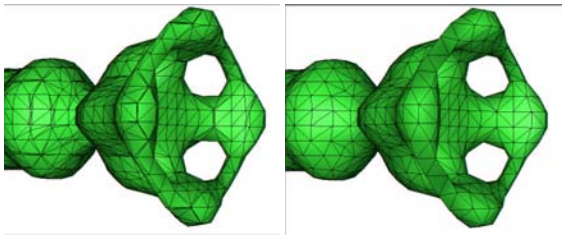


Figure 2: *Isosurface from fuel data set (www.volvis.org), isovalue 80. a) Marching Cubes isosurface. b) SnapMC isosurface (snap parameter 0.3.)*

SnapMC has two drawbacks. First, it can change isosurface topology, eliminating small tunnels and components and merging vertices, edges and faces which are not connected in the isosurface triangulation but are geometrically close. Second, because our algorithm merges geometrically close vertices, edges and faces, it can and often will produce non-manifold isosurfaces. Non-manifold surfaces can be a real problem for numerical simulation software. Post processing can be used to unglue merged vertices, edges and facets, creating a manifold with well-shaped triangles.

Our paper makes the following contributions:

1. An algorithm for constructing extended isosurface lookup tables with '+' or '−' or '=' labels assigned to vertices. The Marching Cubes Algorithm run with this extended lookup table does not create any degenerate triangles.
2. An algorithm for constructing isosurfaces without small areas, edges or angles. The algorithm is similar to the one in [LS07], but it uses a lookup table for cubes not tetrahedra. It can also be modified for other convex mesh elements, such as pyramids or bipyramids.

2. Background

Lorensen and Cline [LC87] published the Marching Cubes Algorithm in 1987. A year earlier, Wyvill et. al. [WMW86] published a somewhat similar isosurface extraction algorithm, but without the use of isosurface patch lookup tables. Marching Cubes is a fast, efficient, easily implementable algorithm because of its use of lookup tables.

There are numerous variations and improvements of the original Marching Cubes Algorithm. We cite only

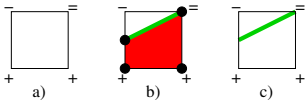


Figure 3: *2D illustration of isosurface patch construction. a) Square vertices labelled '+', '−' and '='. b) The convex hull of the midpoint of the bipolar edges and the '+' and '=' vertices. c) The line segment on the boundary of the convex hull which is not on the square boundary.*

the two most relevant to our paper. Lachaud and Montanvert in [LM00] and independently Bhaniramka et. al. in [BWC00,BWC04] gave algorithms for automatically generating isosurface lookup tables for the Marching Cubes Algorithm. Their algorithms apply to generating lookup tables for other convex mesh elements such as tetrahedra or pyramids and to mesh elements in dimension $\mathbb{R}^4$ or higher.

Schreiner et. al. [SSS06] gave an advancing front method for constructing isosurfaces with good triangles. However, the method has difficulties in a few cases where the front meets itself. Dey and Levine [DL07] gave an isosurface construction and meshing algorithm based on Voronoi diagrams and Delaunay triangulations. Meyer et. al. [MKW07] use a particle based system to construct a point sampling of the isosurface and then reconstruct the isosurface from the sample points using the TIGHT COCONE software [DG03]. All three algorithm are adaptive, producing large triangles in flat, featureless regions and small triangles in regions with high curvature.

We cite only the most relevant articles on mesh generation. Bernd, Eppstein and Gilbert [BEG94] introduced the idea of warping a background grid for mesh generation in 2D. Mitchell and Vavasis [MV00] generalized the algorithm to higher dimensions. Labelle and Shewchuk [LS07] applied the grid warping to a body centered lattice and combined it with a Marching Cubes style lookup table to quickly generate tetrahedral meshes with guaranteed bounds on dihedral angles.

3. Constructing Isosurface Patches

Given a convex polyhedron Φ with vertices labeled '+' or '−' or '=', we construct triangles representing an isosurface patch in Φ. As previously defined, an edge of Φ is *bipolar* if one endpoint has label '+' and another endpoint has label '−'.

Our algorithm is as follows. Create a set W_Φ of the midpoint of the bipolar edges of Φ and the vertices of Φ which are labeled '+' or '='. Construct the convex hull, $\mathrm{conv}(W_\Phi)$, of W_Φ. (See Figure 3.)

If $\mathrm{conv}(W_\Phi)$ is three dimensional, then triangulate $\partial(\mathrm{conv}(W_\Phi))$, the boundary of $\mathrm{conv}(W_\Phi)$. Remove from this triangulation all triangles contained in a facet of Φ. The remaining set of triangles forms the isosurface patch. If $\mathrm{conv}(W_\Phi)$ is contained in a facet of Φ, then return all the triangles in a triangulation of $\mathrm{conv}(W_\Phi)$. If $\mathrm{conv}(W_\Phi)$ is one or two dimensional but not contained in a facet of Φ, then return the empty set. (See Figure 4.)

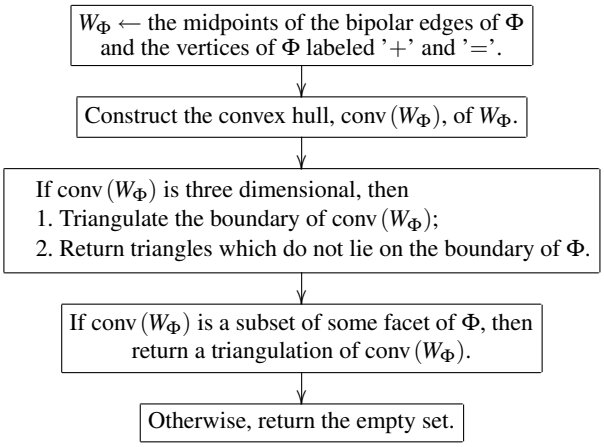


Figure 4: *Isosuface patch construction in a convex polyhedron Φ.*

If Φ has no vertices with label '=' and W_Φ is not empty, then $\mathrm{conv}(W_\Phi)$ is a three dimensional set. In this case, the construction of the isosurface patch is exactly the same as that given in [LM00, BWC00, BWC04].

4. Constructing the Extended Isosurface Table

The extended isosurface lookup table contains isosurface patches for all possible vertex labellings of the unit cube. Each vertex has three possible labels, '+' or '−' or '='. A cube has eight vertices so there are $3^8 = 6561$ entries in the extended lookup table.

Each entry in the extended lookup table contains a list of triangles representing an isosurface patch within a cube. This triangles list is generated by applying the algorithm in Figure 4 to the unit cube with the appropriate vertex labels.

The extended isosurface lookup table is generated in a preprocessing step and stored in a file. A file (in xml format) representing the extended isosurface lookup table can be downloaded from www.cse.ohio-state.edu/graphics/isotable along with C++ source code for reading this file into an isosurface generator.

5. MC Isosurfaces with No Degenerate Triangles

The algorithm for constructing isosurfaces using the extended isosurface lookup table follows the classical Marching Cubes algorithm. To add isosurface triangles lying on the

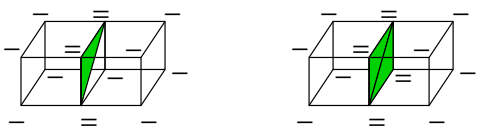


Figure 5: *Grid facet with duplicate isosurface triangles. Identical (but oppositely oriented) isosurface triangles are created in each of the cubes incident on the facet.*

boundary of the grid, the algorithm includes an additional step for processing the grid boundary.

The extended isosurface lookup table is read from a file. Each grid vertex v is labeled '+' if its scalar value s_v is greater than the isovalue σ, '$-$' if s_v is less than σ, and '=' if s_v is equal to σ. For each grid cube, find the lookup table entry corresponding to its vertex labels. Retrieve the isosurface triangles from that entry, forming an isosurface patch within the grid cube.

The isosurface vertices in the lookup table lie either on cube vertices with label '=' or on the midpoints of bipolar cube edges. As in Marching Cubes, we reposition the isosurface vertices on bipolar grid edges using linear interpolation. Repositioning the isosurface vertices implicitly repositions all the isosurface triangles. The isosurface is the union of all the isosurface triangles.

The final step of the algorithm adds isosurface triangles to the grid boundary. For every square on the grid boundary with four vertices labelled '=', use a diagonal of the square to form two isosurface triangles covering that square. For every square on the grid boundary with three vertices labelled '=' and one labelled '$-$', form an isosurface triangle from the vertices labelled '='.

We claim the constructed isosurface has the following two properties:

1. The isosurface does not contain any zero area triangles.
2. The isosurface separates vertices with scalar value greater than the isovalue from vertices with scalar value less than the isovalue;

An isosurface *separates* point p from point q if every path in the grid from p to q intersects the isosurface.

Proof sketch of 1. The extended isosurface lookup table does not contain any degenerate triangles. The Marching Cubes algorithm with the extended lookup table never positions two triangle vertices at the same location and never moves three triangle vertices onto the same grid edge. Thus no three triangle vertices are ever collinear and the isosurface contains no degenerate triangles. □

The proof of 2 is based on the following lemma:

Separation Lemma. *If X and Y are closed subsets of $\mathbb{R}^d$ and $X \subseteq Y$, then $X \cap \mathrm{cl}(Y - X)$ separates X from $Y - X$.*

Proof sketch of 2. For each cube in the grid, we define a positive and negative region. The positive region is the convex hull of the midpoints of the cube's bipolar edges and its vertices labeled '+' and '='. The negative region is the complement in the cube of the positive region. The isosurface lookup table returns a set T of triangles which separate the positive region of a cube from its negative one.

Let R^+ be the union of all the positive regions of all the grid cubes. Let Ω be the region covered by the grid. Set R^+ contains all the positive grid vertices while $\Omega - R^+$ contains all the negative ones. We claim that set T contains $R^+ \cap (\Omega - R^+)$. (The intersection $R^+ \cap (\Omega - R^+)$ may be a proper subset of T.) By the Separation Lemma, T separates R^+ from $\Omega - R^+$, and thus the positive grid vertices from the negative ones. The linear interpolation step in Marching Cubes never moves an isosurface vertex passed a grid vertex, so the isosurface also separates the positive grid vertices from the negative ones. □

Topological Properties

When the scalar value at a grid vertex equals an isovalue, the constructed isosurface may not be a manifold (with boundary). This reflects the behaviour of the trilinear interpolant. Let f be the scalar field defined by applying trilinear interpolation within each grid cube. Some (but not necessarily all) of the critical points of f lie on grid vertices. If the isovalue σ equals the scalar value of such a grid vertex, then $f^{-1}(\sigma)$ is not a manifold.

The constructed isosurface may also have duplicate triangles, albeit with opposite orientation. Consider three or four vertices labelled '=' which lie on on a grid facet and are surrounded by grid vertices with label '$-$'. (See Figure 5.) The two grid cubes containing the facet each contribute identical, although oppositely oriented, triangles lying in the facet. Intuitively, the isosurface has collapsed onto itself.

6. Snapping for Quality Mesh Generation

The Marching Cubes Algorithm is notorious for producing isosurface triangles with small angles. Such triangles are created when two triangle vertices are very close to a grid vertex while the third is far away. To avoid creating such a triangles we modify the scalar field so that isosurface vertices close to a grid vertex are "snapped" onto that grid vertex.

Algorithm SnapMC takes an input parameter γ in the range $[0, 0.5]$ to control the snapping. All isosurface vertices less than distance γL of a grid vertex are "snapped" to a grid vertex where L is the length of the grid edges.

The algorithm begins by determining all the bipolar edges of the original scalar grid and using linear interpolation to locate isosurface vertices. If an isosurface vertex lies within distance γL of a grid vertex, then the scalar value at the grid vertex is set to σ, the isovalue.

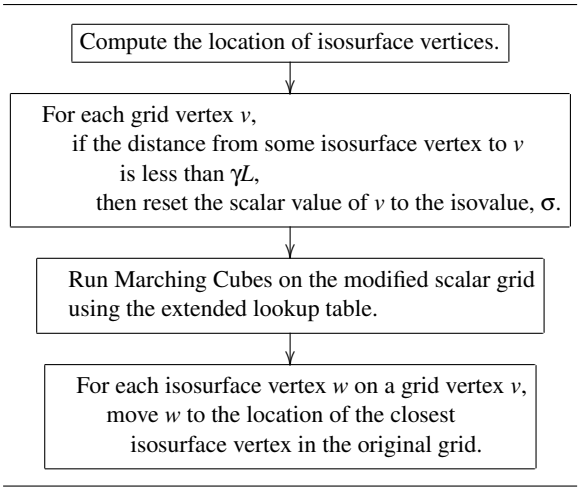


Figure 6: *Algorithm SnapMC. The snap parameter γ controls the snapping of isosurface vertices to grid vertices. L is the length of grid edges.*

	Values of γ			
	0.1	0.2	0.3	0.4
min length	0.14	0.28	0.42	0.6
min area	0.01	0.04	0.08	0.1
min angle	4.7°	8.9°	12.7°	6.4°
max angle	164.1°	149.6°	144.2°	162.4°

Table 1: *Min. edge lengths, min. area, and min. and max. angles for different values of γ on a grid with unit edge length $L = 1$. Angles are in degrees.*

After resetting all the appropriate grid values, algorithm SnapMC runs the Marching Cubes algorithm on the new scalar grid, using the extended lookup table described in Section 4. As claimed in Section 5, the algorithm does not create any degenerate triangles.

The final step moves "snapped" vertices back to one of the original isosurface vertices. Each isosurface vertex lying on a grid vertex is repositioned to the location of the closest isosurface vertex in the original grid. The algorithm is presented in Figure 6.

Topology

Snapping combines geometrically close isosurface vertices, even if these vertices are not connected by isosurface edges and lie in different "parts" of the isosurface. Combining such vertices can create topological changes in the isosurface. (See Figure 7.) It can eliminate small isosurface components, join different components and close loops in the isosurface. However, such topological changes are "small" in the sense that they are produced by small perturbations of the isosurface of a distance less than the length of a single grid edge.

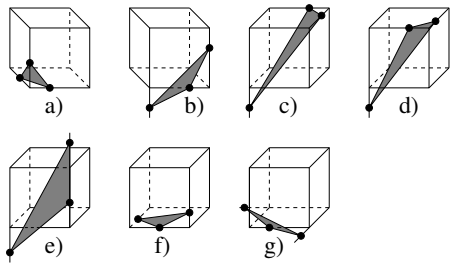


Figure 8: *Extremal configurations. a) Min. edge length for all snap parameters. Min. area for $\gamma = 0.1, 0.2, 0.3$. b) Min. area for $\gamma = 0.4$. c) Min. angle for $\gamma = 0.1, 0.2$. d) Min. angle for $\gamma = 0.3$. e) Min. angle for $\gamma = 0.4$. f) Max. angle for $\gamma = 0.1, 0.2$. g) Max. angle for $\gamma = 0.3, 0.4$.*

Because snapping will merge geometrically close vertices which may not share an edge, it can and often will produce non-manifold surfaces. The surface will have a structure of a manifold (with boundary) which is glued to itself at different vertices, edges and facets. Facets which are glued together will appear as duplicate triangles in the isosurface, although they will have opposite orientation.

Variants of SnapMC

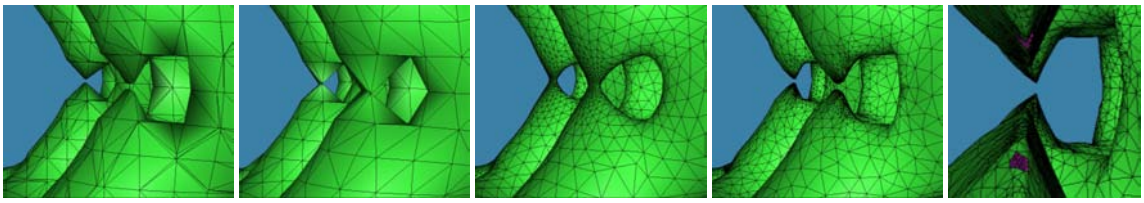
The algorithm for generating an isosurface lookup table which distinguishes '+', '−' and '=' vertices applies to any convex polyhedral mesh element, not just cubes. In particular, it can be used to generate an extended lookup table for tetrahedra, pyramids and bipyramids. The snapping algorithm also can be used for any such mesh although the resulting bounds on isosurface angle size depends on the geometry of the original mesh elements.

7. SnapMC Isosurface Properties

Bounds on the SnapMC edge lengths, triangle areas and triangle angles are a function of γ and are based on extremal configurations. The extremal configurations have isosurface vertices at distance γ from a cube vertex or at the midpoint of some cube edge. Because of snapping, it is possible for an isosurface vertex to lie outside of a cube vertex, as in Figure 8e). We determined the extremal configuration for $\gamma = 0.0, 0.1, \ldots, 0.5$, using a computer program to enumerate all possible combinations of isosurface vertices at distance γ from a cube vertex. Some combinations are not possible and those were eliminated.

The extremal configurations are:

a) $(\gamma, 0, 0)$, $(0, \gamma, 0)$, $(0, 0, \gamma)$:
Min. edge length and min. area, $\gamma = 0.1, 0.2, 0.3$;



a) Marching Cubes b) SnapMC c) Afront d) DelIso e) DelIso (closeup)

Figure 7: *Example of topological changes. Isosurface from silicium data set (www.volvis.org), isovalue 130. a) Marching Cubes isosurface. b) SnapMC isosurface (snap parameter 0.3.) c) Afront (rho 0.5.) d) DelIso. e) Closeup of DelIso showing two holes in the reconstructed isosurface.*

	grid		Regular Lookup Table		Extended Lookup Table		SnapMC ($\gamma = 0.3$)	
	dimensions	isovalue	# triangles	cpu time	# triangles	cpu time	# triangles	cpu time
aneurism	$256 \times 256 \times 256$	100	175,832	0.71 sec	174,300	0.73 sec	106,356	1.52 sec
bonsai	$256 \times 256 \times 256$	30	1,284,542	1.16 sec	1,117,304	1.19 sec	729,623	2.19 sec
engine	$256 \times 256 \times 128$	100	608,416	0.58 sec	593,963	0.59 sec	427,338	1.08 sec
lobster	$301 \times 324 \times 56$	20	312,948	0.33 sec	300,340	0.34 sec	181,058	0.77 sec

Table 2: *Number of isosurface triangles and cpu times of Marching Cubes with regular and extended lookup tables and of SnapMC. CPU time does not include time to read or write data.*

b) $(0,-\gamma,0)$, $(1-\gamma,0,0)$, $(1,1-\gamma,0)$: Min. area, $\gamma = 0.4$;
c) $(1-\gamma,1,1)$, $(0,-\gamma,0)$, $(1,1,1-\gamma)$:
Min. angle, $\gamma = 0.1, 0.2$;
d) $(1-\gamma,1,0)$, $(0,-\gamma,0)$, $(1,1,\gamma)$: Min. angle, $\gamma = 0.3$;
e) $(1,\gamma,0)$, $(0,-\gamma,0)$, $(1,1+\gamma,0)$: Min. angle, $\gamma = 0.4$;
f) $(0,0,\gamma)$, $(0,0.5,0)$, $(1,\gamma,0)$: Max. angle, $\gamma = 0.1, 0.2$;
g) $(-\gamma,0,1)$, $(\gamma,0,0)$, $(1,0,-\gamma)$: Max. angle $\gamma = 0.3, 0.4$.

See Figure 8. Minimum area and minimum and maximum angles for specific values of γ are shown in Table 1.

Isosurface triangles in Figure 8b), 8e), and 8g) overlap a face of the cube. While such triangles cannot be created by the original Marching Cubes algorithm, they can be created using the extended isosurface lookup table described in Section 4. The triangle in the extended lookup table connected three vertices in a cube face. These three vertices were repositioned to the three isosurface vertices seen in the figure.

	Values of γ			
	0.1	0.2	0.3	0.4
min length	0.141	0.29	0.43	0.58
min area	0.001	0.04	0.09	0.15
min angle	4.87°	9.7°	13.6°	12.5°
max angle	161.3°	146.7°	135.1°	144.0°
Hausdorff	0.80	0.81	0.86	0.93
mean dist	0.014	0.02	0.04	0.05
RMS dist	0.060	0.07	0.09	0.11

Table 3: *Measurements of lobster isosurface. Min. edge length, min. area, min. and max. triangle angles, directed Hausdorff distance, mean directed distance and root mean squared directed distance. Measurements are for the isosurface with isovalue 20 except for the distances which compared isosurfaces with isovalue 20.01.*

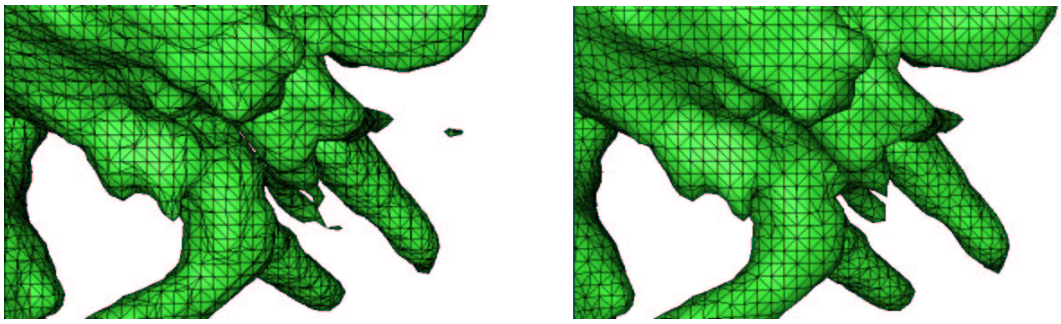
8. Results

We implemented our algorithms in C++ and applied them to the publicly available data sets at www.volvis.org. (See Figures 2 and 9.) Running times are for a computer with two Intel Xeon 2.80 GHz CPU's, a 2048K cache and 8 GB RAM running Linux.

We first compared the regular Marching Cubes algorithm and Marching Cubes with the extended lookup table. See Table 2. The CPU times were the almost same. The reported CPU times measure only the time to run construct the isosurface and do not include the time to read in the scalar data or write out the isosurface mesh. The time to read in the isosurface lookup table is negligible for both the regular and extended table.

The regular Marching Cubes algorithm produces an isosurface containing degenerate triangles. The difference in the number of isosurface triangles between the Marching Cubes with the regular and extended lookup tables is the number of such degenerate isosurface triangles.

Output size and CPU times for SnapMC with snap parameter 0.3 are reported in Table 2. SnapMC reduced the number of isosurface triangles by 25-40% over the Marching Cubes isosurface. Of course, the reduction depends greatly on the snap parameter γ, with smaller values of γ giving less reduction in the number of isosurface triangles.



a) b)

Figure 9: *Isosurface from lobster data set (www.volvis.org), isovalue 20. a) Marching Cubes isosurface. b) SnapMC isosurface (snap parameter 0.3.)*

	isovalue	min edge length	min area	min angle	max angle	min radius ratio	directed Hausdorff
aneurism	100	0.425	0.078	13.09	135.20	0.29	0.86
bonsai	30	0.427	0.083	13.35	135.67	0.28	0.86
engine	100	0.428	0.080	13.96	134.71	0.26	0.66
fuel	80	0.428	0.104	14.30	135.51	0.39	0.30
lobster	20	0.428	0.087	13.55	135.13	0.25	0.86
Marschner-Lobb	100	0.442	0.231	14.58	122.63	0.35	0.71

Table 4: *Measurements on SnapMC isosurfaces. Snap parameter* $\gamma = 0.3$. *The minimum radius ratio is the minimum ratio of the inscribed to circumscribed circle for any isosurface triangle. Directed Hausdorff distance was computed on isosurfaces with isovalue 0.01 greater than the isovalues listed in column two.*

SnapMC took approximately twice as long as the Marching Cubes algorithm. The extra time was spent in "snapping" scalar values in the grid to the isovalue.

We ran SnapMC on different data sets varying the snap parameter γ. We measured the minimum isosurface edge length, the maximum isosurface triangle area, the minimum and maximum angle in an isosurface triangle, and the minimum inradius to circumradius ratio. Output measurements for the data set lobster (Figure 9) are presented in Table 3. Output measurements for other data sets and snap value 0.3 are presented in Table 4. As can be seen, the actual minimum and maximum values are quite close to the theoretical ones.

To measure how much SnapMC modified the isosurface, we measured the difference between the SnapMC isosurface and the original isosurface using the tool METRO [CRS98]. METRO measures the Hausdorff distance between two surfaces. For technical reasons, we measured only the directed Hausdorff distance from the SnapMC surface to the original surface and used a slightly perturbed isovalue. The directed Hausdorff distance from P to Q is $\max_{p\in P}\min_{q\in Q}|p-q|$. We also measured the mean directed distance defined as $\sum_{p\in P}(\min_{q\in Q}|p-q|)\,/\,|P|$, and the root mean squared directed distance defined as $\sum_{p\in P}(\min_{q\in Q}|p-q|)^2/|P|$. The sums are over a set of sample points chosen from the surface. METRO reports all these distances.

The directed distances for lobster are in Table 3. Distances for the other data sets with snap value 0.3 are in Table 4. Note that all distances are less than 1.

Comparison with DelIso and Afront

We compared SnapMC with two publicly available isosurface meshing programs, Afront [SSS06] and DelIso [DL07]. (See Figure 7 and Table 5.) Afront meshing is controlled by a parameter rho which we set to 0.5. As expected, SnapMC ran an order of magnitude faster than the other two. Afront failed to complete on two of the data sets and DelIso failed on one. Afront and DelIso created triangles with extremely small angles and DelIso often created degenerate triangles with zero area or edge length.

On all the data sets, SnapMC joined different regions of the isosurface, creating a non-manifold. However, on many of the data sets DelIso also produced a non-manifold, creating duplicate triangles, or edges or vertices whose neighborhoods were not manifolds (nor manifolds with boundary.) DelIso almost produced a manifold on engine but created a single non-manifold vertex. All the other non-manifold isosurfaces produced by DelIso had multiple non-manifold vertices or edges. The surface produced by DelIso sometimes had "holes". (See Figure 7e).)

		SnapMC ($\gamma = 0.3$)				DelIso				Afront			
	isoval	# tri	cpu	angle	man	# tri	cpu	angle	man	# tri	time	angle	man
aneurism	100.5	106K	3.7s	13.2	no	242K	5.7m	0	no	602K	0.5h	1.6	yes
bonsai	30.5	691K	4.4s	13.4	no	2745K	29m	0	no		10h+		
engine	100.5	427K	2.0s	13.6	no	244K	2.2m	0.2	no		10h+		
lobster	20.5	223K	0.8s	13.3	no	Aborted				3353K	2h	0.6	yes
lobster	30.5	183K	0.7s	13.4	no	2354K	24m	0	no	1222K	50m	1.0	yes
silicium	60.5	25K	0.0s	15.2	no	125K	1.2m	0	no	52K	3m	11.2	yes
silicium	130.5	22K	0.0s	14.5	no	390K	3.9m	0.0	no	160K	5m	3.1	yes
marschner	100.5	10K	0.0s	15.4	no	946K	11m	0.1	no	379K	15m	2.3	yes

Table 5: *Comparison with DelIso and SnapMC. CPU times for SnapMC and DelIso do not include time to read or write data. CPU times for SnapMC and Deliso are in seconds and minutes, respectively. Afront times include time to read and write data and are in hours or minutes as indicated. Column angle represents the minimum triangle angle. Column man indicates whether the isosurface is a manifold (with boundary.)*

On the data sets which Afront completed, Afront produced manifolds, but it sometimes changed the topology of the isosurface. (See Figure 7c).) When we set Afront's parameter "rho" to 0.8, instead of 0.5, Afront failed to produce a manifold on all these data sets.

Both Afront and DelIso are adaptive, adding more triangles in areas of high curvature. SnapMC is not.

Numerical Simulation

SnapMC produces well-shaped triangles which are good for numerical simulation. Unfortunately, SnapMC often produces non-manifold isosurfaces which are problematic for simulation software. Snapping causes geometrically close vertices, edges and triangles, to merge, collapsing the surface onto itself. Post processing could reverse this process, by ungluing merged vertices and edges, and separating duplicate triangles. We have not yet implemented such a post processing algorithm.

Source Code

SnapMC source code and executables are publicly available at www.cse.ohio-state.edu/graphics/isotable.

References

[BEG94] BERN M., EPPSTEIN D., GILBERT J.: Provably good mesh generation. *J. of Computer and System Sciences 48*, 3 (1994), 384–409.

[BWC00] BHANIRAMKA P., WENGER R., CRAWFIS R.: Isosurfacing in higher dimensions. In *Proceedings of Visualization 2000* (2000), pp. 267–273.

[BWC04] BHANIRAMKA P., WENGER R., CRAWFIS R.: Isosurface construction in any dimension using convex hulls. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (Mar./Apr. 2004), 130–141.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17* (1998), 167–174.

[DG03] DEY T. K., GOSWAMI S.: Tight cocone: a watertight surface reconstructor. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications* (New York, NY, USA, 2003), ACM, pp. 127–134.

[DL07] DEY T., LEVINE J.: Delaunay meshing of isosurfaces. In *Proc. Shape Modeling International* (2007), pp. 241–250.

[LC87] LORENSEN W., CLINE H.: Marching cubes: a high resolution 3d surface construction algorithm. *Comput. Graph. 21*, 4 (1987), 163–170.

[LM00] LACHAUD J.-O., MONTANVERT A.: Continuous analogs of digital boundaries: A topological approach to iso-surfaces. *Graphical Models 62* (2000), 129–164.

[LS07] LABELLE F., SHEWCHUK J. R.: Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph. 26*, 3 (2007), 57.

[MKW07] MEYER M., KIRBY R. M., WHITAKER R.: Topology, accuracy, and quality of isosurface meshes using dynamic particles. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1704–1711.

[MV00] MITCHELL S. A., VAVASIS S. A.: Quality mesh generation in higher dimensions. *SIAM J. Comput. 29*, 4 (2000), 1334–1370.

[She02] SHEWCHUK J. R.: What is a good linear element? In *Eleventh International Meshing Roundtable* (2002), pp. 115–126.

[SSS06] SCHREINER J., SCHEIDEGGER C., SILVA C.: High-quality extraction of isosurfaces from regular and irregular grids. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1205–1212.

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for *soft* objects. *The Visual Computer 2*, 4 (1986), 227–234.