Edge Transformations for Improving Mesh Quality of Marching Cubes

Carlos A. Dietrich, Carlos Scheidegger, John Schreiner, João L. D. Comba, Luciana P. Nedel, and Cláudio T. Silva, *Member, IEEE*

***Abstract*— Marching Cubes is a popular choice for isosurface extraction from regular grids due to its simplicity, robustness, and efficiency. One of the key shortcomings of this approach is the quality of the resulting meshes, which tend to have many poorly shaped and degenerate triangles. This issue is often addressed through post processing operations such as smoothing. As we demonstrate in experiments with several datasets, while these improve the mesh, they do not remove all degeneracies, and incur an increased and unbounded error between the resulting mesh and the original isosurface. Rather than modifying the resulting mesh, we propose a method to modify the grid on which Marching Cubes operates. This modification greatly increases the quality of the extracted mesh. In our experiments, our method did not create a single degenerate triangle, unlike any other method we experimented with.**

Our method incurs minimal computational overhead, requiring at most twice the execution time of the original Marching Cubes algorithm in our experiments. Most importantly, it can be readily integrated in existing Marching Cubes implementations, and is orthogonal to many Marching Cubes enhancements (particularly, performance enhancements such as out-of-core and acceleration structures).

***Index Terms*— Meshing, Marching Cubes.**

I. Introduction

Isosurfaces are ubiquitous in visualization [14], where they are often the main computational component of important processing pipelines and are heavily used in practice. Some visualization applications require the ability to render an isosurface, and this can be done by first converting this implicit surface into a triangle mesh and then rendering it. Other applications, however, operate on the resulting mesh, such as finite element method simulations, tetrahedral mesh generation and inverse problems. These applications require meshes of good quality, which are often dictated by the quality of its worst triangle [30], *regardless* of any other triangle in the mesh.

The classic approach to compute isosurfaces is to apply the Marching Cubes (MC) algorithm [22] or one of its variants, such as [5], [11]. Although robust and simple to implement, these generate surfaces that require additional processing steps to improve triangle quality and mesh size. All of these algorithms are cell-based: they work by iteratively examining each cell of the grid on which the scalar function f is defined, and producing a triangulation for each cell separately. These triangulations are created in such a way that when they are connected together, they produce a watertight manifold mesh [25]. The simplicity of these methods allows robust and efficient algorithms, which have been expanded and extended in significant ways [2], [7], [20], [36]. Among these techniques are optimization strategies that are orders of magnitude faster than the original algorithm, and can work on data of arbitrarily large sizes. However, they still produce low quality triangles.

In this paper, we address the issue of improving the quality of the worst triangle generated by MC. Figure 1 illustrates a comparison of MC against several other strategies, showing triangle quality, as well as the histogram of triangle qualities and Hausdorff distances between the mesh generated by each approach and the original MC mesh (as measured by Metro [8]). The red line in the histograms show the quality of the worst triangle. In essence, that is the measure we are interested in, since it often dictates the quality of the simulations performed on the mesh [30].

We show in this paper a novel method to improve the quality of the triangle mesh generated by MC, by modifying the MC sampling grid in a very simple and intuitive way. We call our method **Ma**rching **C**ubes with **E**dge **T**ransformations (**Macet**). Macet generates a mesh with identical connectivity to the MC mesh, which is very close to the MC mesh as measured by Hausdorff distance. Most importantly, it consistently generates meshes whose worst triangles are well above the current state-of-the art techniques. Macet is simple to implement and very fast. Although we have no lower bound proof for the triangle quality in the general case, we provide extensive experimental evidence and a fully open-source implementation.

MC often generates bad triangles because the regular sampling structure of the grid leads to intersections being computed at inconvenient locations on a subset of the active edges (often close to one of the endpoints of the active edge). By allowing active edges to be repositioned (i.e. moving their endpoints) to more adequate locations, we improve the quality of the triangles which use the intersections generated. Although the initial sampling grid of the MC is modified, we provide conditions that preserve the topology of the mesh. Our algorithm is easy to understand and implement. It keeps most the MC structure intact, being implemented as a new stage between the detection of active edges and the intersection calculation. Given this, most of suggested MC optimizations (like Span Space [29]) in the literature still apply.

The main contributions introduced in this paper are:

- A novel approach for understanding the triangle quality distribution of meshes generated by Marching Cubes. This work motivated the ideas behind the edge transformations that are shown to eliminate the bad triangles generated by Marching Cubes.
- A revised MC algorithm called Macet that combines edge transformations to generate an output mesh with the same connectivity and small error (as measured by Metro) to MC,

Carlos A. Dietrich, João L. D. Comba and Luciana P. Nedel are with Instituto de Informática, Universidade Federal do Rio Grande do Sul, Brasil. E-mail: {cadietrich,comba,nedel}@inf.ufrgs.br.

Carlos Scheidegger, John Schreiner and Cláudio T. Silva are with the Scientific Computing and Imaging Institute, University of Utah, E-mail: {cscheid,jmschrei,csilva}@sci.utah.edu

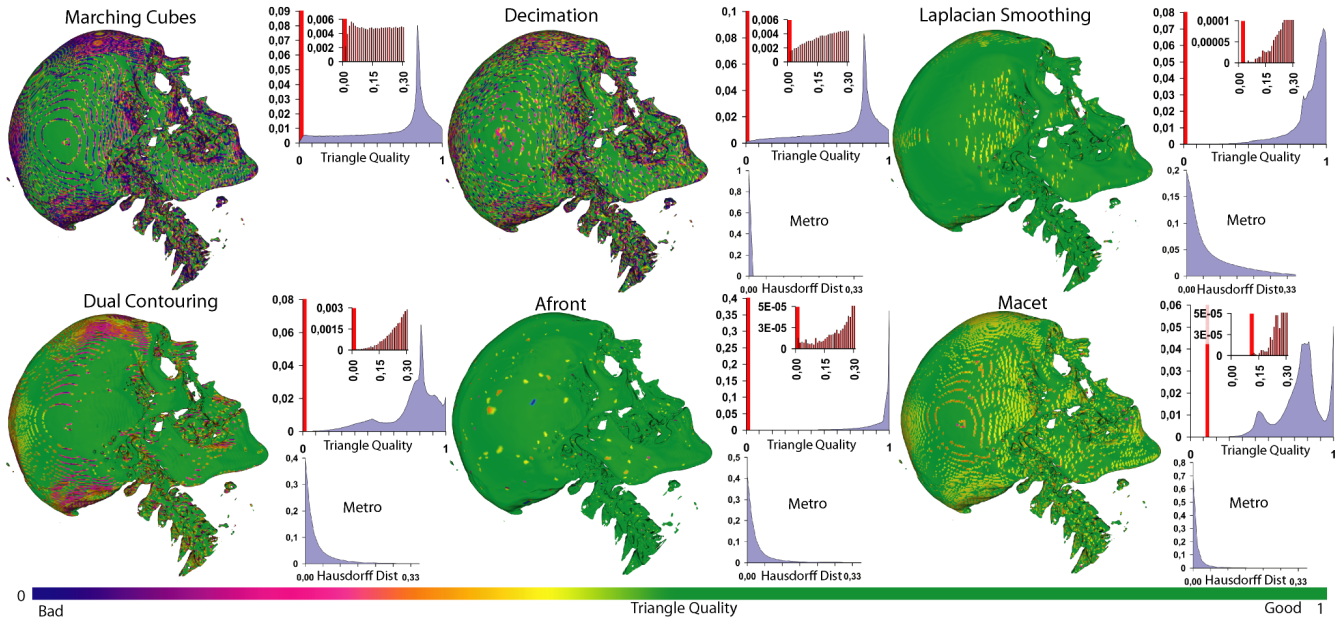


Fig. 1. Improving mesh quality in Marching Cubes. Top-row: Visible Human dataset, mesh generated by the classic Marching Cubes (base mesh used in the comparisons), and results obtained with a post-processing using Decimation to 90%, Dual Contouring, afront and our modified **MA**rching **C**ubes using **E**dge **T**ransformations (Macet). Quality is color-coded on a per vertex basis, defined from the minimum radii-ratio of all incident triangles, and color coded from 0 (worst triangle quality, blue) to 1 (equilateral triangle, green). Intermediate row: Metro histogram with the forward distance between the mesh generated by Marching Cubes and each alternative. Bottom row: triangle quality histograms and insets showing the quality of the worst mesh triangles (the red line shows the quality of the worst triangle). Even though smoothing methods greatly improve the overall quality of the mesh, they still generate a considerable amount of badly shaped triangles. Macet, on the other hand, did not produce a single degenerate triangle during our experiments.

and also with improved quality of its worst triangle.
- A detailed experimental study comparing the triangle quality of existing techniques.

In Section II, we present a discussion on related techniques proposed to solve several issues raised in this work. Then, in Section III, we study the issues related to the mesh quality generated by Marching Cubes. This leads us to revisit the inner computation in those techniques, and to propose an alternative scheme, described in Section IV. The latter part of the paper presents our experimental results, discussion, and future work.

II. Related Work

Isosurface polygonization methods are efficient tools for extraction and visualization of isosurfaces since the pioneering work in the early 80s [1], [6], [15]. Methods based on surface tracking place *seed* sampling points on the isosurface and perform an iterative refinement search for optimal positions or generation of new seeds. Examples include pseudo-physical algorithms [10], [37] and advancing front algorithms [28]. Spatial decomposition methods, described first by Herman and Udupa [15], rely on the assumption that inside a smaller cell of a grid we could assume that the underlying scalar function is locally linear [35], and thus the isosurface can be represented by a plane. Using this simple assumption, Lorensen and Cline [22] proposed the Marching Cubes (MC) algorithm, that is arguably the most important isosurface polygonization algorithms due to its simplicity, efficiency, and robustness.

However, MC is also known for the poor quality of the resulting triangle mesh. Several strategies for measuring triangle quality are discussed in [26]. The metric most commonly used for measuring a single triangle quality is the radii ratio: the ratio between the triangle's incircle and circumcircle. The radii ratio is a *fair* metric: every degenerate triangle has radii ratio zero. It penalizes both small and large angles, which makes it suitable as a "first-order" metric: although some applications might not have problems with small or large angles, a fair metric will favor meshes which are suitable across many application domains. For measuring the quality of meshes, Shewchuk [30] states that the quality of the worst triangle gives a better estimate of mesh quality than other aggregate metrics, such as average or median quality.

In order to overcome such shortcomings of MC, several extensions are discussed in the literature [3]–[5], [16], [21], [23], [24], [33]–[35]. To obtain a high-quality triangle mesh from MC, post-processing steps are typically applied directly to the triangle mesh [10]. For instance, one might apply a standard smoothing algorithm such as Laplacian Smoothing (LS) to improve the triangle qualities, or a decimation procedure to remove degenerate triangles. Observe in Figure 1 that although LS improves the quality, it also moves the surface away from the original MC mesh (as can be seen on the Hausdorff error histogram). QEM-based decimation [12] is more faithful to the original mesh, but fails to remove many degenerate triangles. Several other polygonization methods have been proposed to generate better quality meshes [9], [10], [13], [28]. These techniques often take a more global approach that try to optimize vertex sampling over the complete isosurface. For instance, the recently developed advancing-front (afront) algorithm of Schreiner et al [28] works by first creating an initial seed point on the isosurface, and iteratively growing the triangulation over the entire isosurface, while guaranteeing a global grading constraint over most triangles. It can produce very high quality adaptive triangle meshes, albeit at a computational cost that is much higher than MC. Additionally, the algorithm itself is more involved, leading to a higher complexity of the implementation, and still prone to generating a few degenerate

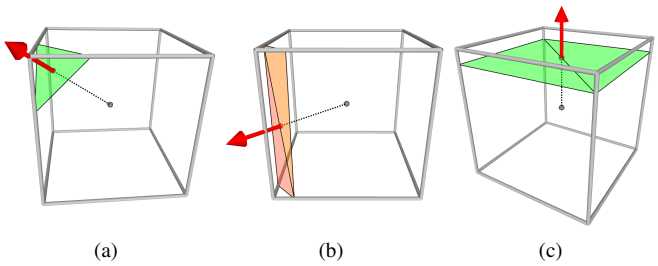


Fig. 2. Several possible plane orientations colored by triangle quality.

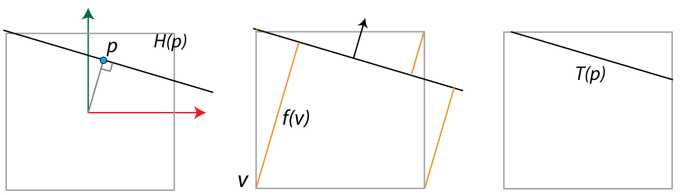


Fig. 3. A point p is associated with an isosurface orientation given by $H(p)$ that passes through p and has normal $p/|p|$. This plane is used to create the function f by setting the value of f at the vertices of the cell to be the signed distance from $H(p)$. The marching method can then be run on the cell to produce a set of triangles $T(p)$.

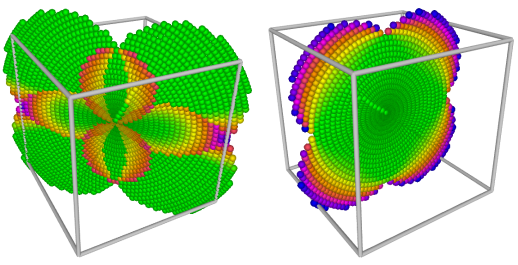


Fig. 4. Two slices of the quality field. Worst quality is obtained when points are closer to the center of edges.

triangles (Figure 1). Dual methods such as Dual Contouring [16] modify the sampling grid and produce better meshes than MC (as can be seen on Figure 1). The purpose of Dual Contouring, however, is to generate adaptive meshes while preserving the sharp features of the isosurface, instead of high-quality meshes. Although the quality of the resultant mesh is visually impressive, many degenerate triangles are still generated. The Dual Contouring method was later improved by Schaefer et al. [27], which propose an extension that constructs adaptive *manifold* surfaces. Although a manifold mesh is better suited for subsequent mesh smoothing methods, their extension produces meshes with the same quality of Dual Contouring meshes.

Following recent work [3], [16], [34], we show that the triangle quality can be significantly improved by modifying the grid *before* the mesh generation. We start from the observation that MC cells generate well-shaped triangles in many, but not all, of the possible intersections with a planar isosurface. Modifying a cell to produce good triangles would then result in a high-quality mesh that accurately reproduces the isosurface, reducing the need of post-processing steps.

Unlike our approach, the Warping Cubes approach of Tzeng [34] allows topological changes in the mesh, but requires user parameters to decide which triangles should be removed. Volume Warping [3] also deforms the grid where isosurfaces are extracted, but the goal there is to increase sampling in a specific user-defined region, instead of improvements in mesh quality. Recently, Labelle and Shewchuk have proposed a related algorithm to generate tetrahedral meshes that warps a different lattice, the BCC (body centered cubic) [18]. The approach used in that paper to prove bounds in the dihedral angles might be fruitful to prove bounds in our case.

III. Studying the Mesh Quality of Marching Cubes

In this section, we show that Marching Cubes generates bad triangles only in some particular configurations. This basic insight leads to the transformations we propose later. Our analysis is centered on a simple scenario – computing all possible intersections of a planar isosurface against a single grid cell centered at the origin. Observe in Figure 2 some examples of such intersections, with each intersection color-coded using the radii-ratio triangle quality as done in Figure 1.

One way to analyze all plane orientations intersecting a cell is to uniquely associate each of them with a single point p in the embedding space. For each plane H, we associate it with the point p on the plane that is closest to the origin of the grid cell. We denote this association by $H(p)$. Given $H(p)$, we reconstruct the function f at the cell vertices as the signed distance to $H(p)$. After defining the value of the function f at each vertex, we run MC on the cell to extract the set of triangles $T(p)$ representing the isosurface (Figure 3). Finally, we compute the average quality of these triangles to create a single scalar quality value for the isosurface parameterized by the point p.

We use this mapping from the point p, to the plane $H(p)$, to the set of triangles $T(p)$, and finally to the quality to define a scalar field $Q(p)$ over $\mathbb{R}^n$. This scalar field can then be visualized to get an estimate on the triangle qualities generated by MC (Figure 4 shows two slices of Q). A clear structure can be seen in the visualization of the quality field Q. The best triangles are created with points p near the corners of the cell, where the plane cuts off the corner with a nearly equilateral triangle (Figure 2.a). These high-quality triangles make up the majority of triangles that are generated, as can be seen in Figure 4. We observe that the worst quality triangles are generated with points p from the center of the cell out toward the edge centers (Figure 2.b). These points correspond to isosurfaces that are nearly parallel to some of the edges of the cell. This pattern of poor quality triangles is independent of cell shape, being observed with cubes, tetrahedra, and octahedra. This is the key insight: *cell edges nearly parallel to the isosurface are correlated with poor triangles in Marching Cubes.* Our modification to Marching Cubes directly addresses this issue.

IV. Marching Cubes Using Edge Transformations

From now on, we focus our discussion only on Marching Cubes, since extensions to other cell types are straightforward. Marching Cubes is conceptually a simple algorithm designed to process an implicit function, defined over a three-dimensional grid, and given by its samples $f(i,j,k)$. It operates by *marching* over the *cubes* implicitly defined by the 3-D sampling grid, looking at each cell of eight vertices independently, and computing small patches of the overall isosurface by considering the intersection of the function f with a cell.

One way to see how Marching Cubes operates on a given 3-D grid cell is to consider its eight vertices $v_1, v_2, \ldots, v_8$, and how

their scalar values $f(v_1), f(v_2), \ldots, f(v_8)$ compare to the desired isovalue λ. For simplicity of presentation, it is convenient to work with the function $f^*(x) = f(x) - \lambda$. Clearly, if all $f^*(v_i)$ share the same sign, the isosurface does not intersect the cell (using trilinear reconstruction). Not taking into account symmetries, there are 2^8 possible configurations. A lookup table of template topologies for these configurations can be created in such a way that the triangles from each cell are joined into a conforming mesh.

For each configuration, the set of *active edges*, i.e., edges where the function f^* has different signs at the endpoints, independently determines both the geometry and topology of the mesh. The topology is determined by the configuration of active edges for each cell. The geometry is determined by the location of the isovalue along each active edge. This independence leaves room to change the geometry of the mesh while keeping the topology intact.

Our modification to Marching Cubes explores this independence. Some edge transformations do not change the underlying topology: they only change the geometry of the isosurface. In particular, consider an active edge (v,w) of the regular sampling grid illustrated in left of Figure 5. If we move each of its endpoints continuously, with the constraint that $f^*(v)$ and $f^*(w)$ do not change signs, this edge will remain active (two different strategies are shown on Figure 5). For each active edge, then, we create a "detached" edge (v',w'), which allows shared endpoints to move independently. We will move the detached edges to improve their intersection points with the isosurface. To guarantee that the geometry of the isosurface stays valid (i.e., the mesh remains oriented, does not self-intersect, and is consistent with the initial topology), we need two *conditions* on the transformations:

- An active edge must keep only one intersection with the isosurface throughout the motion of its endpoints. Since each active edge is shared by several cells, we must enforce that each intersection computation, which is done independently by MC, generates the same intersection point. This is trivially done in MC since this computation is performed only once for each active edge.
- The intersection induced by the detached edge (v',w') with the isosurface must not result in triangle flips.

Now, with these conditions in place, we can return to our goal of improving triangle shape. In Section III we traced the poor triangle quality to those triangles created when the active edges of a cell are nearly parallel to the isosurface. Here, we will use this observation, together with the necessary conditions discussed above to transform detached versions of the active edges of a cell to make them orthogonal to the isosurface. Below, we show two possible methods for achieving this followed by an algorithm proposal that combines both approaches.

A. Gradient Transformation

The intuition behind our first method is based on moving the vertices of the cell along the direction ∇f away from the isosurface. Since ∇f is orthogonal to the isosurface at points near the isosurface, this will increase the distance between the endpoints, but will not significantly move them tangentially. This results in edges that are more perpendicular to the isosurface, and thus produce higher quality triangles.

Given an active edge (v,w), we first generate a detached edge (v',w'). Each vertex of the detached edge will be moved independently. Each vertex is moved away from the surface in the direction of the gradient, until one of the two previously described conditions is violated. To avoid long computations, we limit the displacement to half of the original edge length. In other words, we choose a vertex p' that maximizes α in

$$p' = p + \alpha \tilde{\nabla} f(p)$$

where $\tilde{\nabla} f$ denotes the direction of the gradient, but pointing away from the isosurface.

The weighting factor $\langle \nabla f(p'), \nabla f(p) \rangle$ is used to limit displacement, and to guarantee that edge endpoints will not cross the isosurface. As the vertices move, we need to update the induced intersection points. Figure 7 shows three steps while applying a gradient transform on a spherical isosurface. Even for small displacements, the triangle quality is significantly better. Complete results are shown in Section V.

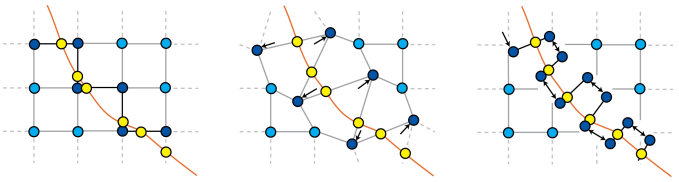


Fig. 5. Standard Marching Cubes is depicted on the left. We propose two methods for improving the shapes of the elements by transforming the active edges. In the middle, we move the edge endpoints along ∇f, and on the right we move the endpoints parallel to the isosurface. The intersection between the edge and the isosurface is recomputed after the edge transformation, which guarantees that the resulting mesh adheres to the isosurface defined by the interpolant. Both methods improve the quality of the resulting mesh.

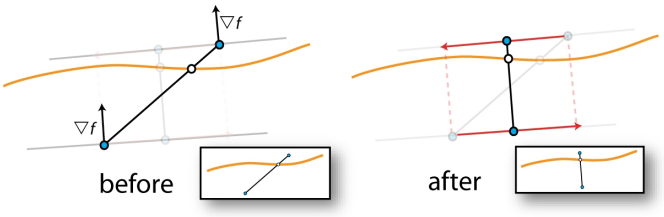


Fig. 6. Edge track computation. A track for a vertex is defined by the projection of the edge to be orthogonal to ∇f. The displacement added to the vertex is half of the track, and the intersection with the isosurface is recomputed.

B. Tangential Transformation

The other way to make the active edges more perpendicular to the isosurface is to move the endpoints orthogonally to ∇f. This method for transforming edges works by moving the edge endpoints parallel to the isosurface. This tangential transform allows tracking of complex isosurface behavior by moving the vertices of active edges in such way that they become *nearly*-perpendicular to the isosurface.

The main step of this transform consists in calculating, for each edge vertex, a *vertex track*, tangent to the isosurface and aligned to the edge. We will optimize the vertex positions along the tracks so that the active edges become more orthogonal to the isosurface.

More formally, a vertex track is defined as the projection of the active edge on the isosurface, displaced to start at each edge vertex. If the isosurface is planar, the track is a line segment whose direction is given by

$$d(v) = (I - \hat{\nabla} f(p) \hat{\nabla} f(p)^T)(v - w)$$

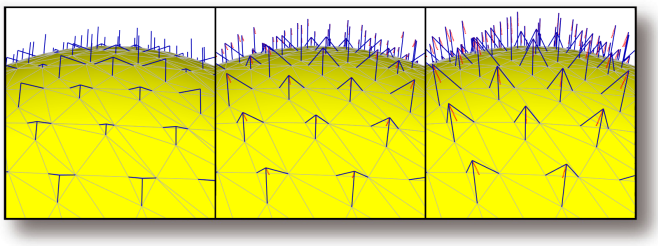


Fig. 7. Three steps of the gradient transform on active edges (blue lines). The procedure iteratively moves endpoints along gradient vectors (dotted red lines), improving the underlying mesh.

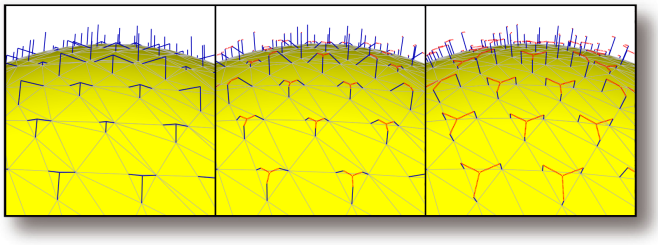


Fig. 8. Three steps of the tangential transform applied to the active edges (blue lines). The procedure iteratively moves the active edges endpoints tangentially to the isosurface (dotted red lines), effectively improving the underlying triangle mesh (gray lines).

$$d(w) = (I - \hat{\nabla} f(p) \hat{\nabla} f(p)^T)(w - v)$$

where v and w are the active edge vertices, and $I - vv^T$ is the matrix which projects a vector onto the orthogonal complement of v. In our implementation, we simplify the track by using a set of line segments that approximate the track. The vertex positions are, then, the one which maximizes $\langle \nabla f(t), v' - w' \rangle$, where

$$v' = v + \alpha d(v)$$

$$w' = w + \alpha d(w)$$

and t is the intersection of the edge (v', w') and the isosurface. If the gradients were all constant, it is easy to show that the optimal vertex position would be with $\alpha = 0.5$. In practice, the gradients are not constant, so this might not be true. For the curved case, then, we find the mid-point of a piecewise linear approximation of the track. Figure 6 illustrates the transformation, while Figure 8 shows three steps while applying a tangential transform on a spherical isosurface.

C. Combining Edge Transformations

Although gradient and tangential transforms are based on different approaches to make edges nearly-perpendicular to the isosurface, they generate the same transformed edge *iff* the conditions we impose for the transformations to operate (discussed in the beginning of this section) are satisfied. However, due to practical difficulties in moving points inside the scalar field, we have found that conditions are not simultaneously satisfied for both transformations, and therefore each transformation tends to improve the overall triangle quality in different situations. That is, in areas where the gradient transform does not significantly change the triangle quality, the tangential transform may exhibit a large improvement, and vice versa. This observation suggested to us that combining the transformations would further improve the overall triangle quality when compared to Marching Cubes.

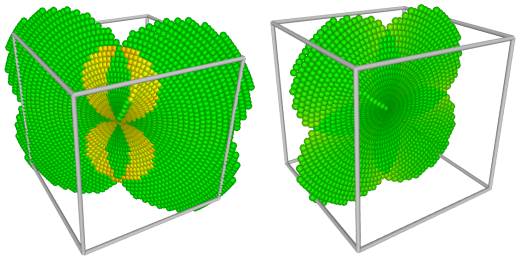


Fig. 9. Visualization of two slices of the *Macet* quality field. Notice the reduction (or extinction) of cases where bad-shaped triangles are generated inside the cell. Compare the inset to Figure 4.

We use an iterative approach for combining edge transformations. Since both transformations only alter vertex positions, while keeping intact the mesh topology, we can choose which of the two positions produces a better average quality for the triangles connected to that vertex. Since the choice for a particular vertex might affect its neighbors, this process might require several iterations. For all of our tests, we proceed with this iterative selection without explicitly building the mesh until the overall mesh quality stabilizes (often less than 4 steps in all our experiments). The results we obtain show that the output mesh has improved quality over each individual mesh produced by using a single edge transformation.

D. Implementation Details

The implementation of edge transformations requires only minor changes to Marching Cubes. The transformations are performed after finding active edges, as an intermediate stage before the intersection calculation. However, there are important issues related to each transform parameters and the way the interpolator is used. MC assumes a linear interpolation along sampling edges of the grid. Even though this is a hard constraint that might lead to topological inconsistencies in the polygonization, it is also a reasonable assumption for fine sampling granularities. This assumption is no longer valid when we move edge endpoints freely inside the volume, because small movements of a grid vertex will result in non-linear variations of the scalar field along its incident grid edges (even if we prevent active edges from changing their state upon vertex movements). Therefore, we need a more robust intersection calculation procedure to track the intersection of the edge against the isosurface. We use standard bisection-based root finding procedures, which do not assume linearity along the edge.

The vertex track used in the edge transformations also demands the continuity of the gradient field (C^1 continuity). Since the trilinear interpolation often used with MC is not differentiable at the cell boundaries, there may be unpredictable results. In this work, we use a cubic spline interpolation to reconstruct the derivatives of the scalar field. We also use the corresponding cubic spline to reconstruct the scalar field itself at the sampling grid vertices.This means there might be more than one root in active edges. However, since the topology has already been determined by the lookup into the Marching Cubes tables and the intersection calculation returns only one vertex, this presents no practical problems.

Dataset	Method	radii ratio		Metro
		min	avg	frac of bb diagonal
Engine 256x256x128 isovalue 49.5	MC	0.000448	0.701	-
	LS	0.000546	0.868	0.00168
	Dec	9.50e-7	0.723	4.20e-5
	DC	0.00528	0.829	0.00246
	afront	0.000995	0.938	0.00832
	Macet	0.236	0.775	0.00129
Lobster 301x324x56 isovalue 30.5	MC	0.000970	0.673	-
	LS	0.0314	0.865	0.00245
	Dec	6.80e-7	0.700	8.10e-5
	DC	0.00226	0.833	0.00204
	afront	0.000396	0.939	0.00915
	Macet	0.0757	0.763	0.00163
Bonsai 256x256x256 isovalue 49.5	MC	0.000192	0.680	-
	LS	0.000792	0.862	0.00182
	Dec	7.53e-7	0.709	7.50e-5
	DC	4.52e-5	0.818	0.00265
	afront	1.49e-5	0.935	0.0824
	Macet	0.100	0.767	0.00140
Silicium 98x34x34 isovalue 140.5	MC	0.00298	0.666	-
	LS	0.0847	0.872	0.00685
	Dec	8.96e-5	0.697	0.000370
	DC	0.000336	0.832	0.00758
	afront	0.0283	0.936	0.0321
	Macet	0.0885	0.764	0.00561

TABLE I

RESULTS COMPARING SEVERAL ISOSURFACE EXTRACTION METHODS: MARCHING CUBES (MC), MC FOLLOWED BY LAPLACIAN SMOOTHING (LS), MC FOLLOWED BY DECIMATION, DUAL COUNTOURING (DC), THE ADVANCING FRONT METHOD OF SCHREINER ET AL. [28] (AFRONT), AND OUR METHOD (MACET). HIGHER RADII-RATIOS INDICATE HIGHER QUALITY MESHES. THE METRO RESULTS SHOW THE HAUSDORFF DISTANCE OF EACH MESH TO THE MESH GENERATED BY ORIGINAL MC.

Dataset	MC	Macet	Overhead(%)
Engine	44.4s	81.5s	183
Lobster	27.4s	46.5s	169
Bonsai	86.9s	125.1s	143
Silicium	0.77s	2.6s	341

TABLE II

PERFORMANCE RESULTS OF MC COMPARED TO MACET. NOTICE HOW THE OVERHEAD DECREASES AS THE DATASET SIZE INCREASES.

Dataset	Method	Time	Steiner Points	Output		
				Points	Faces	Tets
Silicium	MC	152.9s	169K	192K	1.45M	646K
	Macet	62.6s	66.8K	87.4K	598K	257K
Lobster	MC	3130.5s	1.72M	1.98M	16.0M	7.27M
	Macet	767.9s	693K	912K	6.72M	2.97M
Engine	Macet	1362.97s	1.27M	1.69M	10.7M	4.57M

TABLE III

COMPARISON OF THE TETRAHEDRAL MESHES THAT TETGEN CREATES WHEN GIVEN BOTH MC AND MACET SURFACE MESHES AS INPUT. TETGEN WAS NOT ABLE TO CREATE A MESH WHEN USING THE MC INPUT FOR THE ENGINE DATASET.

The combination of edge transforms is implemented by keeping two buffers with vertex locations, one for the gradient transforms and another for the tangential transforms. After the transformations have been performed, we use a greedy algorithm to pick which of the two vertices should be used in the final improved version. The vertex pairs are processed one at a time, and the previously computed vertices are used as soon as they are available, in an iterative fashion. We stop the iterative procedure when the mesh quality (the quality of the worst triangle) is no longer improved (maximized), or the number of iterations is greater than the maximum number of iterations allowed. In our experiments, this final pass has small overhead when compared to the full surface extraction.

V. RESULTS

The techniques described in Section IV have been implemented in C++. Table I and II show a summary of our experimental results. All the timings reported were obtained on a Pentium 4 3.0GHz PC with 2GB RAM. The edge transforms require additional accesses to the scalar field, in order to determine the vertex displacement. As the dataset complexity increases (high curvature or sharp features), the constraints force the vertex movement to happen in smaller steps, which increase the processing time. Notice, however, that the overhead for Macet decreases as the dataset size increases, and that for bigger datasets this overhead is at most a factor of two.

Figure 11 shows the forward Hausdorff distance results produced by Metro, which measures the accuracy of the mesh in relation to the original MC mesh. Although QEM-based decimation produces a mesh very close to MC, it creates more degenerate triangles, as can be seen in Table I. Figure 12 shows visual comparisons and triangle quality histograms for the same data. In particular, we highlight a zoomed version the complete histogram to allow a better evaluation on the distribution of degenerate triangles. Since Figure 4 provided the motivation for the development of edge transformations, it is important to check the quality of our triangulations under that metric. Figure 9 shows two slices of the quality field after applying Macet, with a clear reduction of degenerate triangles. Similar improvement is also present in real datasets, as can be seen in Table I.

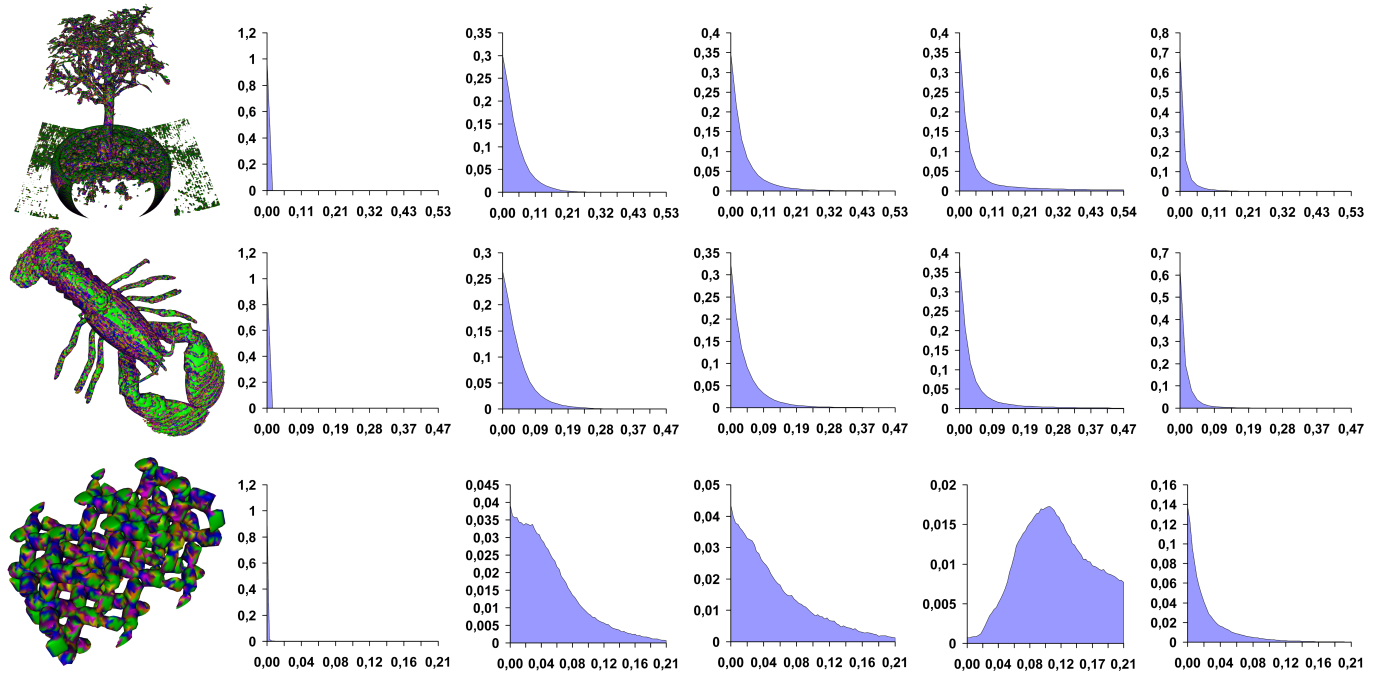
VI. APPLICATION: CDT FOR MESH GENERATION

One way to illustrate the importance of removing degenerate triangles is given below. tetgen is a suite of geometric algorithms that allows computing a Constrained Delaunay Tetrahedralization (CDT) from a piecewise linear complex [31], [32]. Further processing of CDTs often require quality bounds for the generated mesh. tetgen allows imposing quality constraints to the CDT construction, such as a minimum radius-edge ratio, and such constraints are enforced by adding Steiner points to the CDT. This process is prone to numerical problems and very sensitive to the triangle quality of the input mesh.

Table III shows tetgen results setting minimum radius-edge ratio to be 4 (Silicium and Lobster) and 6 (Engine). tetgen crashes using the MC mesh for the engine dataset, but works for Macet. Note that to compensate for bad quality triangles, tetgen generates many more Steiner points for MC, which increases all output elements generated, as well as processing time. Figure 10 compares the tetgen CDT generated for MC and Macet meshes.

VII. CONCLUSIONS AND FUTURE WORK

In this work we have presented edge transforms as a simple and effective way to generate higher quality meshes. We incorporated them into an existing marching cubes implementation, which we called Macet, that has a relatively modest performance penalty



(a) Dataset (b) Decimation (c) Laplacian (d) Dual Contouring (e) afront (f) Macet

Fig. 11. Metro Results: histogram shows the forward distance between mesh generated by each approach and original MC mesh. The error histograms for afront show worse results than expected because afront triangulates a cubic spline surface, while MC uses trilinear interpolation.

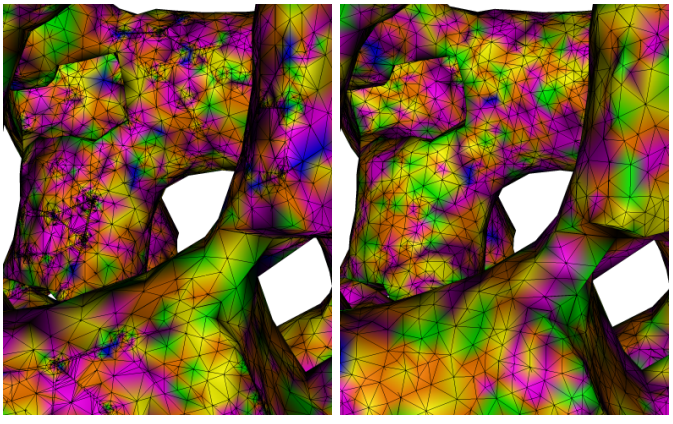


Fig. 10. Boundary of the CDT generated by tetgen for the MC (left) and Macet (right) meshes. Note several degenerate tetrahedra in the MC mesh.

(especially when compared to non-marching algorithms). Since we are able to produce better results, there is much less of a dependence on post-processing operations to improve the quality of the triangles. Through detailed analysis and comparison against competing strategies, we show how Macet generates meshes that improve the quality of degenerate triangles, and are faithful to the isosurface generated by MC (as measured by Metro). We show how this improvement on quality generated by our method is crucial for further mesh processing, such as the generation of CDTs using tetgen.

For future work we plan to extend this approach for working with other variants of Marching Cubes, in particular, variants that provide topological guarantees, are able to keep sharp corners, and adaptive versions of Marching Cubes [17], [19]. Also, we want to investigate the effect of warping the cells and active edges on the reconstructed surface on a theoretical level, and, in particular, try to provide formal theoretical bounds on the quality of the triangles.

Reproducibility

We have made the techniques presented in this paper publicly available, so that the reader can accurately assess the validity of the experiments and algorithms. Every result generated for this paper can be reproduced with open-source software and publicly available datasets. Here is a link to the source code with instructions:

http://www.vistrails.org/index.php/ImprovingMeshQualityOfMarchingCubes

Acknowledgments

The work of Carlos Dietrich is supported by a CAPES scholarship. The work of João Comba and Luciana Nedel is partially supported by CNPq grants 478445/2004-0, 305947/2005-2 and 306099/2004-7. Carlos Scheidegger, John Schreiner, and Claudio Silva are partially funded by the National Science Foundation (grants CCF-0401498, EIA-0323604, OISE-0405402, IIS-0513692, and CCF-0528201), the Department of Energy, IBM Faculty Awards (2005, 2006, and 2007), Sandia National Laboratories, and Lawrence Livermore National Laboratory. Carlos Scheidegger is also funded by an IBM Ph.D. fellowship. All datasets used in this work are available at http://www.volvis.org. We thank the authors of Metro and tetgen for making their code available as open source.

REFERENCES

[1] E. Artzy, G. Frieder, and G. T. Herman. The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. In *ACM SIGGRAPH 1980*, pages 2–9, 1980.

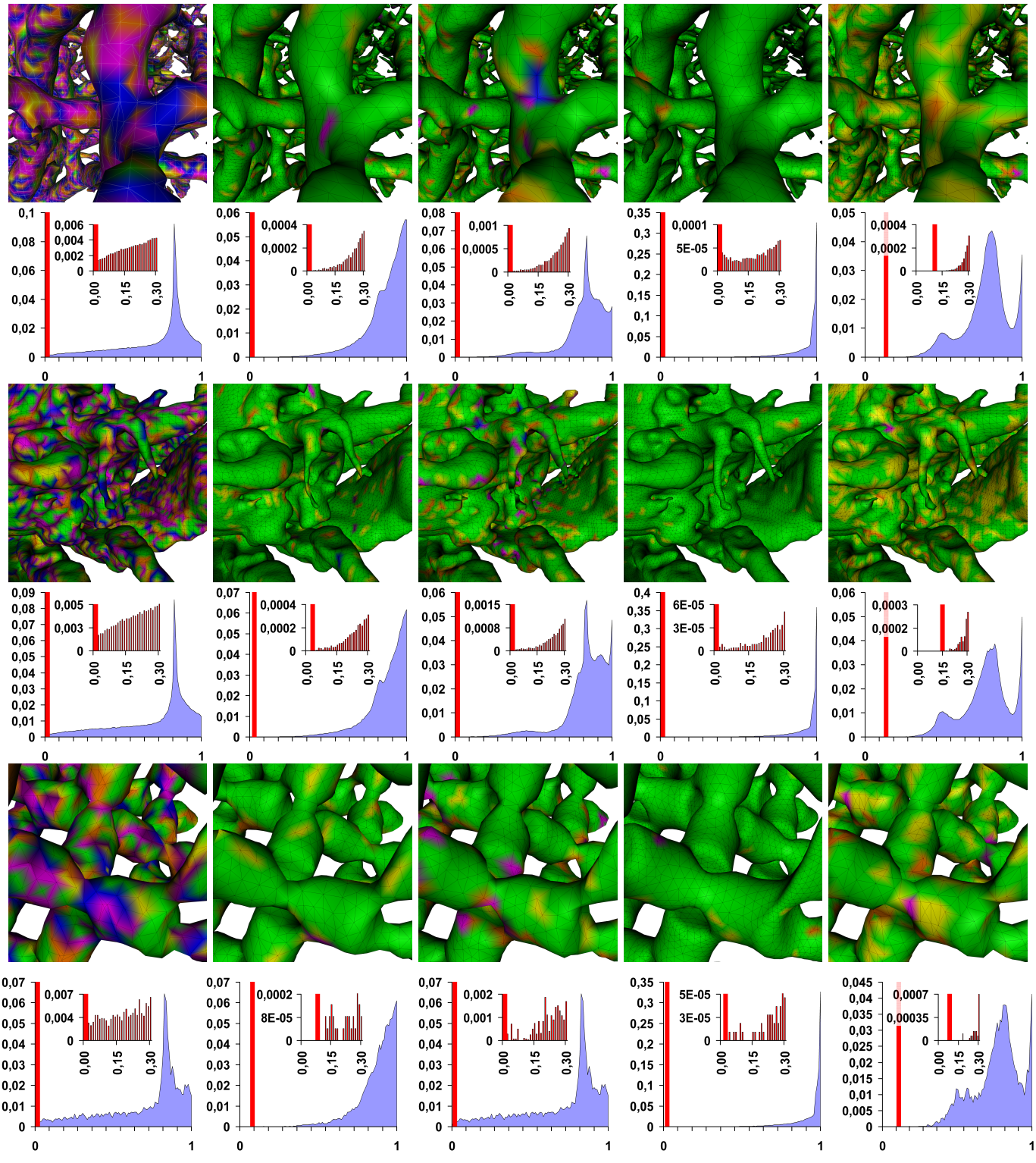


Fig. 12. Comparisons on enlarged sections of Bonsai (rows 1-2), Lobster (rows 3-4) and Silicium (rows 5-6). Color-coded visualization shows that most methods improve the overall quality of triangles. The quality histogram shown below each screenshot shows how Macet consistently reduces degenerate triangles — pay special attention to the zoomed version of the histogram, and compare to the minimum quality results on Table I

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

[3] L. Balmelli, C. J. Morris, G. Taubin, and F. Bernardini. Volume warping for adaptive isosurface extraction. In *IEEE Visualization '02*, pages 467–474, 2002.

[4] H. Carr, T. Möller, and J. Snoeyink. Artifacts caused by simplicial subdivision. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):231–242, 2006.

[5] H. Carr, T. Theussl, and T. Möller. Isosurfaces on optimal regular samples. In *Symposium on Data Visualisation 2003*, pages 39–48, 2003.

[6] L.-S. Chen, G. T. Herman, R. A. Reynolds, and J. K. Udupa. Surface

shading in the cuberille environment. *IEEE Computer Graphics and Applications*, 5(12):33–43, 1985.

[7] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Optimal isosurface extraction from irregular volume data. *Symposium on Volume Visualization*, pages 31–38, 1996.

[8] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[9] P. Crossno and E. Angel. Isosurface extraction using particle systems. In *IEEE Visualization 1997*, pages 495–498, 1997.

[10] L. H. de Figueiredo, J. de Miranda Gomes, D. Terzopoulos, and L. Velho. Physically-based methods for polygonization of implicit surfaces. In *Graphics Interface '92*, pages 250–257, 1992.

[11] A. Doi and A. Koide. An efficient method of triangulating equivalued surfaces by using tetrahedral cells. *IEICE Transactions on Communications and Electronics Information Systems*, E74(1):214–224, January 1991.

[12] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH 1997*, pages 209–216, 1997.

[13] M. Gavriliu, J. Carranza, D. Breen, and A. Barr. Fast extraction of adaptive multiresolution meshes with guaranteed properties from volumetric data. In *IEEE Visualization '01*, pages 295–302, 2001.

[14] C. D. Hansen and C. R. Johnson, editors. *The Visualization Handbook*. Academic Press, 2004.

[15] G. T. Herman and J. K. Udupa. Display of 3d digital images: Computational foundations and medical applications. *IEEE Computer Graphics and Applications*, 3:39–46, 1983.

[16] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *ACM SIGGRAPH 2002*, pages 339–346, 2002.

[17] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature-sensitive surface extraction from volume data. In *ACM SIGGRAPH 2001*, pages 57–66, 2001.

[18] F. Labelle and J. Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions On Graphics*, 26, 2007.

[19] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[20] Y. Livnat and X. Tricoche. Interactive point-based isosurface extraction. In *IEEE Visualization '04*, pages 457–464, 2004.

[21] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):16–29, 2003.

[22] W. Lorensen and H. Cline. Marching Cubes: A high-resolution 3D surface construction algorithm. In *ACM SIGGRAPH 1987*, pages 163–169, 1987.

[23] S. V. Matveyev. Approximation of isosurface in the marching cube: Ambiguity problem. In *IEEE Visualization '94*, pages 288–292, 1994.

[24] G. M. Nielson. Dual marching cubes. In *IEEE Visualization '04*, pages 489–496, 2004.

[25] G. M. Nielson and B. Hamann. The asymptotic decider: Resolving the ambiguity in marching cubes. In *IEEE Visualization '91*, pages 83–91, 1991.

[26] P. P. Pebay and T. J. Baker. A comparison of triangle quality measures. In *10th International Meshing Roundtable*, pages 327–340, 2001.

[27] S. Schaefer, T. Ju, and J. Warren. Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):610–619, 2007.

[28] J. Schreiner, C. Scheidegger, and C. T. Silva. High-quality extraction of isosurfaces from regular and irregular grids. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1205–1212, September 2006.

[29] H.-W. Shen, C. D. Hansen, Y. Livnat, and C. R. Johnson. Isosurfacing in span space with utmost efficiency (ISSUE). In *IEEE Visualization '96*, pages 287–294, 1996.

[30] J. R. Shewchuk. What is a good linear element? - interpolation, conditioning, and quality measures. In *11th International Meshing Roundtable*, pages 115–126, 2002.

[31] H. Si. On refinement of constrained delaunay tetrahedralizations. In *15th International Meshing Roundtable*, pages 509–528, 2006.

[32] H. Si and K. Gaertner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *14th International Meshing Roundtable*, pages 147–163, 2005.

[33] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: Improved iso-surface extraction. *Computers & Graphics*, 23:583–598, 1999.

[34] L. Tzeng. Warping cubes: Better triangles from marching cubes. In *20th European Workshop on Computational Geometry*, 2004.

[35] A. van Gelder and J. Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13(4):337–375, 1994.

[36] J. Wilhelms and A. V. Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, 1992.

[37] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *ACM SIGGRAPH 1994*, pages 269–277, 1994.