

Verifiable Visualization for Isosurface Extraction

Tiago Etienne, Carlos Scheidegger, L. Gustavo Nonato,
Robert M. Kirby, *Member, IEEE*, and Cláudio T. Silva, *Senior Member, IEEE*

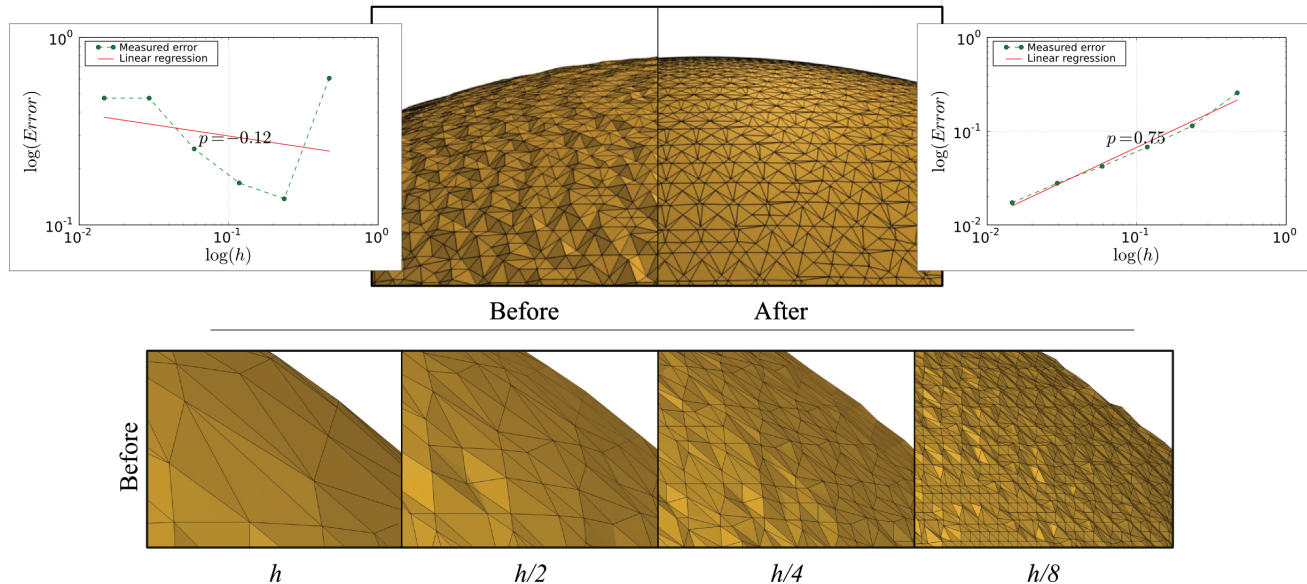


Fig. 1. Through the verification methodology presented on this paper we were able to uncover a convergence problem within a publicly available marching-based isosurfacing code (top left) and fix it (top right). The problem causes the mesh normals to *disagree* with the known gradient field when refining the voxel size h (bottom row). The two graphs show the convergence of the normals before and after fixing the code.

Abstract—Visual representations of isosurfaces are ubiquitous in the scientific and engineering literature. In this paper, we present techniques to assess the behavior of isosurface extraction codes. Where applicable, these techniques allow us to distinguish whether anomalies in isosurface features can be attributed to the underlying physical process or to artifacts from the extraction process. Such scientific scrutiny is at the heart of *verifiable visualization* – subjecting visualization algorithms to the same verification process that is used in other components of the scientific pipeline. More concretely, we derive formulas for the expected *order of accuracy* (or convergence rate) of several isosurface features, and compare them to experimentally observed results in the selected codes. This technique is practical: in two cases, it exposed actual problems in implementations. We provide the reader with the range of responses they can expect to encounter with isosurface techniques, both under “normal operating conditions” and also under adverse conditions. Armed with this information – the results of the verification process – practitioners can judiciously select the isosurface extraction technique appropriate for their problem of interest, and have confidence in its behavior.

Index Terms—Verification, V&V, Isosurface Extraction, Marching Cubes.

1 INTRODUCTION

In this age of scientific computing, the simulation science pipeline of mathematical modeling, simulation and evaluation is a rendition of the scientific method as commonly employed as the traditional experimental pipeline. Critical to this simulation approach is the evaluation stage in which numerical data are post-processed, visualized and then examined in order to answer the original queries that instigated the investigation. In fact, visualization of scientific results has become as much a part of the scientific process as mathematical modeling or

numerical simulation.

Despite its growing importance in the scientific computing process, visualization has not fallen under the same rigorous scrutiny as other components of the pipeline like mathematical modeling and numerical simulation. Unlike in traditional computational science and engineering areas, there is no commonly accepted framework for verifying the accuracy, reliability, and robustness of visualization tools. This precludes a precise quantification of the visualization error budget in the larger scientific pipeline. Furthermore, very few investigations have focused on how the error originating from other components of the computational pipeline impacts visualization algorithms.

In this paper, we advocate the use of techniques from the *verification and validation* process used in the engineering community (Section 2 presents V&V in more detail). While the lack of a well-established framework for verifying visualization tools has meant that a variety of analysis techniques have been developed [26, 29], we believe that visualization has achieved sufficient importance to warrant investigation of stricter verification methodologies. Several authors have already asserted this need [8, 12], and in this paper we present techniques that are a concrete step towards reducing the gap between best practices in simulation science and visualization.

The purpose of this work is to present a verification methodology

- Etienne, Scheidegger, Kirby and Silva are with the School of Computing and Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, 84112, USA. E-mail: {tetiene,cscheid,kirby,csilva}@cs.utah.edu.
- Nonato is with Universidade de São Paulo, Brazil. E-mail: gnonato@icmc.usp.br.

Manuscript received 31 March 2009; accepted 27 July 2009; posted online 11 October 2009; mailed on 5 October 2009.

For information on obtaining reprints of this article, please send email to: tvcg@computer.org .

for visualization tools that has comparable rigor to that of other components of the computational scientific pipeline. More specifically, we set out to define a verification methodology in the mold of the area of V&V, in the context of visualization. Furthermore, we illustrate our proposal by testing several publicly available isosurface extraction codes to the verification procedure, giving a detailed description of the steps involved in the process.

It is important to emphasize that the point of verification procedures is not to compare algorithms to one other with the hope of finding the best alternative. This procedure equips developers of new algorithms and/or implementations with a process that provides a systematic way of identifying and correcting errors in both algorithms and implementations. The goal is to provide users with a methodology that will give them a more concrete model for the behavior of the implementation, which will increase confidence in the visualization tools. As we will show, a fair verification analysis can bring out unforeseen behavior, and quickly detect implementation problems that might not be caught through standard comparisons or visual inspection.

The contributions of this work are threefold. To the best of our knowledge, we, for the first time, apply the framework of verification to assess the correctness of visualization tools. Furthermore, we provide a detailed description of how to accomplish the verification procedure by subjecting different isosurfacing tools to the battery of tests comprising the V&V process. Our second contribution is the underlying mathematical analysis and associated manufactured solutions developed to analyze the isosurfacing methods. We should clarify that when applying MMS for other techniques (even in the case of isosurface extraction), the theoretical analysis should be tailored to the particular features of these algorithms. The manufactured solutions presented here are simple but general enough to be promptly employed for evaluating other visualization tools besides isosurfacing. Our third contribution is a comprehensive set of results obtained using the technique, including the finding of implementation errors in two publicly available isosurface extraction codes.

2 VALIDATION AND VERIFICATION

In this section we provide a brief introduction to the idea of *verification and validation* (V&V), and in particular, its application in visualization scenarios. We will also review efforts to verify implementations in the specific context of isosurface extraction.

Babuška and Oden define verification as “the process of determining if a computational model obtained by discretizing a mathematical model of a physical event and the code implementing the computational model can be used to represent the mathematical model of the event with sufficient accuracy” [1]. Although they review the concept only in the context of computational science and engineering applications, it is important to appreciate that the same idea applies to scientific visualization. *Verification is about investigating to what extent a (numerical) approximation scheme – both in algorithm and corresponding implementation – represents the desired mathematical model.* Validation, on the other hand, is about ensuring that the model represents physical reality. In this paper, we will concern ourselves only with verification, under the assumption that the model has been validated by the user of the technique. This is the perspective Kirby and Silva suggest for “Verifiable Visualization” [12].

One of the main requirements for verifiable visualization is to have a rigorous analysis which predicts the results of the algorithm and its implementation when evaluating it on a known model problem. The more complete this analysis is, the more thorough the testing procedures can be. This *continuous process* of verification through refinement of key controllable input parameters of the method (such as grid spacing) and testing is different from a one-shot process. The verification process should involve a suite of tests with corresponding results from which one can progressively increase reliance on the method under analysis. When appropriately applied, verification provides ways to appreciate the nuances of the applicability of the method. As we will see in this paper, writing down the analysis for the expected result of isosurface extraction gives us concrete bounds on what features we can expect the resulting surface to have, and these are extremely important for users.

A common practice in the visualization community is to test implementations by using complicated, “real-world” datasets. The value of these tests is that they provide evidence of the algorithm’s applicability. We advocate a complementary approach: developers should carefully manufacture test cases that can be mathematically modeled and analyzed, called *manufactured solutions*. These manufactured solutions can then be used to test the implementations. In this paper, we present analysis that describe the expected rate of convergence of several isosurface features, and test implementations acting on our model problems using simple analytical volumes. As we show in Sections 3 and 4, this method helps pin down the mathematical characteristics of the technique, and, more practically, it is quite effective at uncovering implementation bugs.

The challenge behind manufactured solutions is to construct them in a way that allows us to predict the expected behavior of the method under investigation. Moreover, the manufactured solutions should tax the method vigorously, bringing out potential problems. In Section 5, we will present some situations where incorrectly chosen manufactured solutions have a big impact in the results. We do so to emphasize that all components of the pipeline, even the construction and evaluation of the manufactured solutions, must be meticulously handled to maintain the rigor of the verification process.

2.1 Isosurface Extraction Algorithms

Isosurface extraction is a popular visualization technique, being a tool currently used in science, engineering, and applications. This popularity makes it a natural target for this first application of verification mechanisms in the context of visualization. This same popularity has also driven a large body of work comparing different isosurface extraction algorithms.

Previous researchers have examined topological issues [14, 18], mesh quality [6, 23], accuracy [19, 29], and performance [24]. The influence of different reconstruction schemes and filters in scalar visualization has also been examined [3, 20]. In this paper, we focus on techniques to verify the correctness of algorithms and their corresponding implementations. In particular, we provide mathematical tools that other researchers and developers can use to increase their confidence in the correctness of their own isosurface extraction codes. A traditional way to test implementations in scientific visualization is to perform a visual inspection of the results of the Marschner-Lobb dataset [16]. In the context of isosurface extraction, researchers routinely use tools such as Metro [4] to quantitatively measure the distance between a single pair of surfaces. We argue that the methodology presented here is more effective and more explicit at elucidating a technique’s limitations. In particular, our proposal pays closer attention to the interplay between a theoretical convergence analysis and the experimental result of a *sequence* of approximations.

Globus and Uselton [8] and more recently Kirby and Silva [12] have pointed out the need for verifying both visualization techniques and the corresponding software implementations. In this paper, we provide concrete tools for the specific case of isosurface extraction. Although this is only one particular technique in visualization, we expect the general technique to generalize.

It is important to again stress that verification is a *process*: even when successfully applied to an algorithm and its implementation, one can only concretely claim that the implementation behaves correctly (in the sense of analyzed predicted behavior) for all test cases to which it has been applied. Because the test set, both in terms of model problems and analyzed properties, is open-ended and ever increasing, the verification process must continually be applied to previous and new algorithms as new test sets become available. This does not, however, preclude us from formulating a basic set of metrics against which isosurface extraction methods should be tested, as this is the starting point of the process. This is what we turn to in the next section.

3 VERIFYING ISOSURFACE EXTRACTION ALGORITHMS

In this section, we describe the technique we use for verifying isosurface extraction algorithms, namely the *method of manufactured solutions* (MMS). We illustrate a possible implementation of MMS in Figures 2 and 3. This technique requires us to write down the expected

MMS(f, u, h_1)

- 1 ▷ Let f be a scalar field containing the solution surface S
- 2 ▷ Let u be a given property (f , normals, area, etc.)
- 3 ▷ Let h_1 be the initial grid size
- 4 **for** $i \leftarrow 1$ **to** n
- 5 **do** $G_{h_i} \leftarrow$ an approximation of f at grid size h_i
- 6 $S_{h_i} \leftarrow$ an approximation of S computed from G_{h_i}
- 7 $E_{h_i} \leftarrow \|u(S_{h_i}) - u(S)\|_u$
- 8 $x_i \leftarrow \log h_i, y_i \leftarrow \log E_{h_i}$
- 9 $h_{i+1} \leftarrow h_i/2$
- 10 $\tilde{q} \leftarrow$ slope of best-fit linear regression of (x_i, y_i)
- 11 Compare \tilde{q} and q

Fig. 2. Overview of the method of manufactured solutions (MMS).

behavior of particular features of interest of the object (or model problem) being generated. In our case, we are generating triangular approximations of smooth isosurfaces, and the features of interest are geometric surface convergence, convergence of normals, area and curvature.

To use MMS, we first accomplish a mathematical analysis of the expected convergence rate of the features (or characteristics) of interest, known in the numerical literature as the *formal order of accuracy* of the characteristic. This analysis is done for solutions of the problem that can be conveniently described and analyzed (these are the manufactured solutions). Then, the code is executed with progressively refined versions of the data that is used in the generation or sampling of the manufactured solution. Finally, the empirical convergence rate is compared to the one predicted by the analysis. When the convergence rates are comparable, we increase our confidence in the algorithm. If the realizable behavior disagrees with the analysis, either (1) the analysis does not correspond to the correct behavior of the algorithm, (2) the assumptions upon which the analysis was built were violated by the input data and hence the predicted behavior is not valid for the circumstances under investigation, or (3) there are issues with the algorithm or with the implementation of the algorithm (depending on access to source code and algorithmic details, one may not be able to distinguish between these two – algorithmic or implementation – and hence we in this work always consider them together. Given sufficient information, the verification process can help further delineate between these two issues). Notice, however, that all three situations warrant further investigation. In the following sections, we will discuss these issues in more detail. Let us first clarify how we will arrive at theoretical and empirical convergence rates.

For a fixed grid size, we will strive to write the approximation error between the desired isosurface property and its approximation by:

$$E = |u_{approx} - u_{exact}|_u = O(h^p) = \alpha h^p \quad (1)$$

where u_{approx}, u_{exact} are the approximated and exact values of a property u , $|\cdot|_u$ is the norm used to compare the approximate and exact property, p is the order of accuracy and α is a constant.

Practically speaking, the polynomial expression (1) is not very convenient for numerical experimentation, as it is hard to find the value of p from the direct plot of h against E . The standard technique to estimate p is to linearize by working on a log-log scale:

$$\log E = \log(\alpha h^p) = \log \alpha + p \log h. \quad (2)$$

Using this linearized version, we estimate p from the slope of the line that best fits the points $(\log h, \log E)$ in a least-squares sense. We use this technique in Section 4 when testing the isosurface codes.

MMS critically depends on an analysis of the order of accuracy of expected solutions. Although this seems quite simple, the order of accuracy under a sensitive norm like $\|\cdot\|_\infty$ has shown in practice to be very effective in bringing out implementation errors in numerical approximation schemes [1, 22]. In this paper, we show that this analysis is just as effective for isosurface extraction. In addition, we believe the convergence analysis required by MMS is interesting in its own right. As we will discuss in Section 5, it helps to shed light on the consequences of implementation choices.

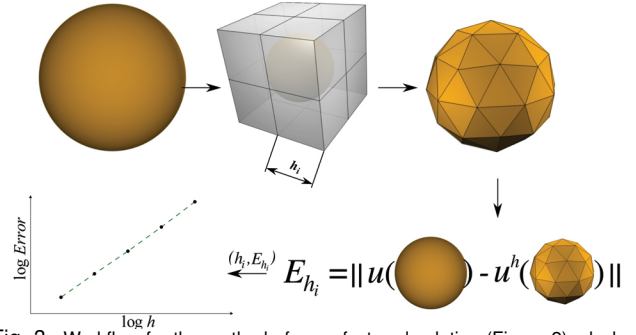


Fig. 3. Workflow for the method of manufactured solution (Figure 2), clockwise from the top left.

In the context of isosurface methods, manufactured solutions can be built by specifying a “solution surface” to be the exact solution and deriving a scalar field that contains such a solution surface as a level set. The verification methodology then proceeds as following: (1) use the manufactured scalar field as input for the isosurfacing methods, (2) run the methods, and (3) check the output surface against the solution surface (sometimes called the *ansatz solution* within the mathematical verification literature). In many cases, the manufactured scalar field can be derived analytically, making the observed order of accuracy tractable (we give examples in next section).

In what follows, we will derive expected orders of accuracy for several features of surfaces produced by isosurface extraction codes. We keep our assumptions about the actual algorithms to a minimum to maximize the applicability of the arguments given. We essentially only assume that the maximum triangle size can be bounded above at any time, and use Taylor series arguments (under assumptions of smoothness) to derive convergence rates. It is important to point out that order of accuracy analysis of polyhedral surfaces has been studied by many researchers [9, 17, 27, 28]. In fact, the results presented below are in agreement with the ones reported in the literature. However, because we are considering isosurface extraction, some of our arguments benefit by being able to be condensed to simpler statements.

3.1 Convergence of Vertex Position

We start our analysis of isosurface extraction by studying the convergence of vertex positions. We analyze this convergence indirectly by relating the values of the scalar field at the vertex points and the distance between the vertices and the correct isosurface. Given a value λ such that the exact isosurface S is defined by $f(x, y, z) = f(\vec{v}) = \lambda$, the *algebraic* distance of \vec{v} to S is defined as $|f(\vec{v}) - \lambda|$ [25]. Notice that algebraic distances only makes sense for implicit surfaces: it requires a scalar field. In addition, we restrict ourselves to *regular* isosurfaces, ones where for every point x in S , $|\nabla f(x)|$ exists and is nonzero. Then, the geometric distance between \vec{v} and S is approximated by $|f(\vec{v}) - \lambda|/|\nabla f(\vec{v})|$ [25]. We illustrate this relation in Figure 4. Since, by assumption, $|\nabla f(x)| > k$ for some $k > 0$, and all x in S , convergence in algebraic distance implies convergence in geometric distance. Convergence in algebraic distance, however, is much more tractable mathematically, and this is the item to which we turn our focus.

Many isosurface methods estimate vertex positions through linear interpolation along edges of a grid. Let $f : U \subset \mathbb{R}^3 \rightarrow \mathbb{R}$ be the a smooth real function defined in a subset $U = [a_x, b_x] \times [a_y, b_y] \times [a_z, b_z]$, where $[a_i, b_i], i \in x, y, z$ are real intervals. We assume the intervals $[a_i, b_i]$ have the same length and let $a_x = x_0, \dots, x_n = b_x$, $a_y = y_0, \dots, y_n = b_y$, and $a_z = z_0, \dots, z_n = b_z$ be subdivisions for the intervals such that $x_i = x_0 + ih$, $y_i = y_0 + ih$, $z_i = z_0 + ih$, $i = 0, \dots, n$, where h is the grid size and $c_{ijk} = [x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}]$ is a grid cell. Through a Taylor series expansion of f , one can evaluate f at a point $\vec{p} \in c_{ijk}$ as:

$$f(\vec{p}) = f_{ijk} + \nabla f_{ijk} \cdot \vec{\delta} + \frac{1}{2} \vec{\delta}^T H(\vec{\xi}) \vec{\delta} \quad (3)$$

where $f_{ijk} = f(x_i, y_j, z_k)$, ∇f_{ijk} is the gradient of f in (x_i, y_j, z_k) , $H(\vec{\xi})$ is the Hessian of f at a point $\vec{\xi}$ connecting (x_i, y_j, z_k) and \vec{p} , and $\vec{\delta} = (u, v, w)^T$ is such that $\vec{p} = (x_i + uh, y_j + vh, z_k + wh)^T$.

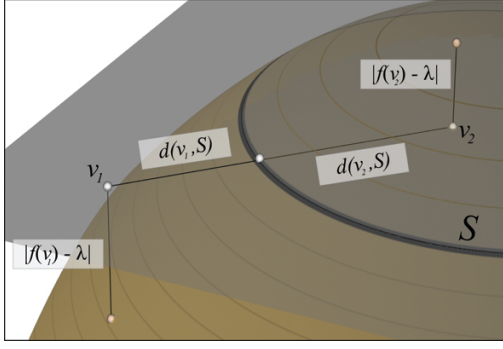


Fig. 4. The distance between a point \vec{v} and the isosurface S with isovalue λ can be approximated by the algebraic distance divided by the gradient magnitude of the scalar field at \vec{v} , $|f(\vec{v}) - \lambda|/|\nabla f(\vec{v})|$. In the figure, the thick circle represents the isosurface S and the fainter isolines illustrate changes in gradient magnitude: in regions of small gradient magnitude, the algebraic distance is small but geometric distance is large, and vice-versa for large gradient magnitude.

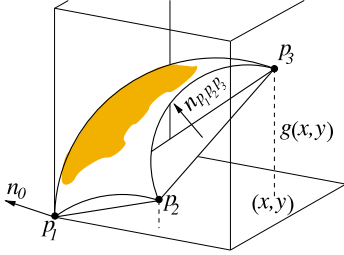


Fig. 5. Isosurface local parametrization and approximation.

Let the linear approximation of f in \vec{p} be defined by

$$\tilde{f}(\vec{p}) = f_{ijk} + \nabla f_{ijk} \cdot \vec{\delta} \quad (4)$$

and consider a point \vec{x}_λ such that $\tilde{f}(\vec{x}_\lambda) = \lambda$, that is, \vec{x}_λ is a point on the isosurface λ of \tilde{f} .

The algebraic distance between the exact isosurface $f(x, y, z) = \lambda$ and the linearly approximated isosurface can be measured by $|f(\vec{x}_\lambda) - \lambda|$. From Equations 3 and 4 one can see that

$$\begin{aligned} |f(\vec{x}_\lambda) - \lambda| &= |f_{ijk} + \nabla f_{ijk} \cdot \vec{\delta} + \frac{1}{2} \vec{\delta}^T H(\xi) \vec{\delta} - \lambda| = \\ |\tilde{f}(\vec{x}_\lambda) + O(h^2) - \lambda| &= O(h^2) \end{aligned} \quad (5)$$

thus, the linearly approximated isosurface is of second-order accuracy.

3.2 Convergence of Normals

Assume, generally, that the scalar field $f(x, y, z) = \lambda$ can be locally written as a graph of a function in two-variables $g(x(u, v), y(u, v)) = \lambda - f(x(u, v), y(u, v), z_k)$, as illustrated in Figure 5, where $x(u, v) = x_i + uh$ and $y(u, v) = y_j + vh$. This is acceptable because we have already assumed the isosurface to be regular. Still without losing generality we write $g(x(0, 0), y(0, 0)) = 0$, that is, the isosurface contains the point (x_i, y_j, z_k) . Let $\vec{\Phi}(u, v) = (x(u, v), y(u, v), g(x(u, v), y(u, v)))$ be a parametrization for the isosurface $f(x, y, z) = \lambda$ in c_{ijk} and

$$\frac{\partial \vec{\Phi}}{\partial u} \times \frac{\partial \vec{\Phi}}{\partial v} = h^2 \left(-\frac{\partial g}{\partial x}, -\frac{\partial g}{\partial y}, 1 \right)^T = h^2 \vec{n}_0 \quad (6)$$

be the normal vector in $\vec{\Phi}(0, 0) = (x_i, y_j, g(x_i, y_j))$ (the partial derivatives of g are evaluated at $(x(0, 0), y(0, 0)) = (x_i, y_j)$).

Consider now the triangle defined by the points \vec{p}_1, \vec{p}_2 , and \vec{p}_3 approximating the isosurface $f(x, y, z) = \lambda$ in the grid cell c_{ijk} (see Figure 5). Let \vec{p}_1 be the grid point (x_i, y_j, z_k) , so $\vec{p}_1 = \vec{\Phi}(0, 0)$, $\vec{p}_2 = \vec{\Phi}(u_2, v_2)$, and $\vec{p}_3 = \vec{\Phi}(u_3, v_3)$. Using the cross product in \mathbb{R}^3 , the normal of the triangle $p_1 p_2 p_3$ can be computed by:

$$\begin{aligned} n_{p_1 p_2 p_3} &= (\vec{p}_2 - \vec{p}_1) \times (\vec{p}_3 - \vec{p}_1) = \\ &\left(\begin{array}{l} h(v_2 g(x(u_3, v_3), y(u_3, v_3)) - v_3 g(x(u_2, v_2), y(u_2, v_2))) \\ h(u_3 g(x(u_2, v_2), y(u_2, v_2)) - u_2 g(x(u_3, v_3), y(u_3, v_3))) \\ h^2(u_2 v_3 - u_3 v_2) \end{array} \right). \end{aligned} \quad (7)$$

Expanding $g(x(u_i, v_i), y(u_i, v_i))$, $i \in \{2, 3\}$ in a Taylor series, some terms cancel and the normal $n_{p_1 p_2 p_3}$ becomes:

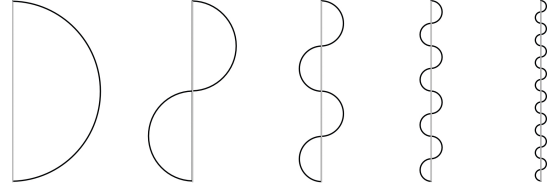


Fig. 6. Uniform convergence does not imply convergence in area. The sequence of curves converges uniformly to a straight line, but the length of the curves does not change.

$$n_{p_1 p_2 p_3} = rh^2 \left(-\frac{\partial g}{\partial x} + O(h), -\frac{\partial g}{\partial y} + O(h), 1 \right)^T \quad (8)$$

where $r = u_2 v_3 - u_3 v_2$. Comparing the exact normal vector \vec{n}_0 in Equation 6 with $n_{p_1 p_2 p_3}$ above, we recover first-order of accuracy for normals. In addition, notice that the usual scheme of estimating vertex normals by the arithmetic mean of triangle normals does not decrease the order of accuracy; that is, vertex normals (computed by arithmetic mean) are at least first-order accurate.

3.3 Convergence of Area

Currently, much less is known about convergence in area, compared to convergence of vertices or normals. To illustrate the difficulty involved in approximating lengths and areas, consider the sequence of approximations to a straight line shown in Figure 6. Even though the function sequence converges uniformly to the line, the length of the approximation stays constant.

To the best of our knowledge, the only relevant results establish convergence in area given convergence in vertex positions *and* convergence in normals, such as in Hildebrandt *et al.* [9]. However, the authors only establish asymptotic convergence, with no order of accuracy associated with it. The argument is more mathematically involved than space allows here, so we refer the reader to that paper. Currently, this means that the only information the observed order of accuracy provides to us is that if we expect convergence in normals, we should also expect convergence in area, and vice-versa.

3.4 Convergence of Curvature

The following formula gives an estimate of the curvature at a vertex p :

$$K(p) = \frac{2\pi - \sum \theta_{i+1}}{\frac{1}{3} A_{i+1}} \quad (9)$$

where θ_{i+1} and A_{i+1} are the angle $\angle p_i p p_{i+1}$ and area of the triangle $p_i p p_{i+1}$ respectively (summation is over all triangles comprising the star of p) [17]. Meek and Walton [17] showed that the curvature computed via Equation 9 does not converge in general; that is, if the vertices of the star of p are arbitrarily distributed around p , one cannot expect curvature convergence. In fact, they described a more general result stating that $O(h)$ accuracy can only be obtained if the normals are known to have accuracy $O(h^2)$. Subsequently, Xu [27] presented a very particular distribution of vertices around p under which the curvature estimated by Equation 9 has accuracy $O(h^2)$.

Curvature discretization schemes other than the one given in Equation 9 such as the quadratic-fit and spherical-image method (see Meek and Walton [17] for details) also demand particular vertex distributions to ensure convergence. In our context of keeping the analysis applicable for many isosurfacing algorithms, this means we cannot use the lack of observed curvature convergence as an indication of problematic behavior. Based on the results mentioned above, one should actually expect curvature not to converge for most isosurface extraction algorithms. More generally, this indicates a weakness of MMS, namely that some features of interest (such as curvature) will not have sufficient theoretical order of accuracy to be used in numerical measurements. Notice, in addition, that if we had not written down the theoretical model for curvature convergence, we might have expected some sort of curvature approximation. Even a negative result such as the one presented in this section can increase the confidence in the results generated by an implementation.

4 EXPERIMENTAL RESULTS

In this section we present the results of applying the afore-described methodology. We use the framework to verify six different isosurface

extraction codes, namely: VTK Marching Cubes [15], SnapMC [21], Macet [6], Dual Contouring [11], Afront [23], and DellIso [5]. All these implementations are open source and/or publicly available¹. Before presenting the actual results of subjecting these implementations to the verification process, we briefly review their salient features.

VTK Marching Cubes: Marching Cubes [15] (MC) is arguably the most popular isosurface extraction algorithm. It reduces the problem of generating an isosurface triangulation to a finite set of cases by considering the *signs* of how the isosurface intersects each cell of a regular background grid. As there are only 256 different types of intersections between the isosurface and a regular Cartesian 3D cell, a template of triangles is set to each case, making the implementation quite simple through a look-up table. The vertices of the triangles lie on the edges of the cubic cells, and they are computed by linearly interpolating the implicit function values stored at the corners of the grid cell.

SnapMC: SnapMC [21] is a recently proposed algorithm that extends the original Marching Cubes look-up table to cases where the isosurface goes exactly through the corners of the background grid. The new look-up table is automatically built by an adaptation of the convex hull scheme proposed by Bhaniramka *et al.* [2]. Even though the traditional Marching Cubes algorithm can easily handle these cases by some kind of symbolic perturbation, SnapMC *perturbs the scalar field* to avoid edge intersections close to grid corners. In particular, it changes the values on the grid so that the surface is “snapped” to the grid corners.

Macet: Macet [6] is another variant of Marching Cubes that tries to improve the shape of the triangles in a mesh. Unlike SnapMC, it *perturbs the active edges* of Marching Cubes cases by moving the vertices before the triangulation step. The motivation behind Macet is that poorly-shaped triangles tend to be generated when the intersection between the isosurface and a grid cell is approximately parallel to an edge of the grid cell. Therefore, some corners of the background grid are displaced so as to avoid the parallel-like intersections.

Dual Contouring: Dual Contouring [11] is a feature-preserving isosurfacing method to extract crack-free surfaces from both uniform and adaptive octree grids. This technique can be seen as a combination of Extended Marching Cubes [13] and SurfaceNets [7] as it makes use of Hermite data and quadratic error function minimization to position the vertices of the surface mesh (as Extended Marching Cubes) and the dual topology to connect such vertices (as SurfaceNets). Dual Contouring tends to generate better quality triangles than Marching Cubes while still being very effective in representing sharp features, rendering this implicit polygonalization method a good alternative to the popular Marching Cubes.

Afront: Afront [23] is an advancing-front method for surface extraction. Although we focus on applying Afront to isosurface extraction, it can also be used for remeshing and triangulating point-set surfaces. The outstanding feature of Afront is that it generates triangles adapted to the local details of a surface, namely its maximum absolute curvature. In this sense, Afront is fundamentally different from the other algorithms we analyze. In lieu of grid refinement, we will use its ρ parameter to control triangulation size. Because the manufactured solution we use is a sphere, reducing ρ by half is roughly equivalent to reducing the maximum triangle size by half. A full analysis of Afront (and, in particular, the influence of the other main parameter η) warrants further investigation, but is beyond the scope of this paper.

DellIso: DellIso [5] is a Delaunay-based approach for isosurfacing. It computes the restricted Delaunay triangulation from a 3D Voronoi Diagram. We run our tests on a customized version of DellIso 16 bit, and our examples use the default set of parameter.

In what follows, we present the results of applying the verification process to these algorithms. We will describe the manufactured solutions we use and their observed convergence rate on the isosurface extraction algorithm.

4.1 Observed order of accuracy

We start by investigating the behavior of the algorithms under the manufactured solution given by the scalar field $f(x, y, z) = x^2 + y^2 + z^2 - 1$

¹Links at <http://www.sci.utah.edu/~etiene/>

	Vertex $O(h^2)$	Normal $O(h)$	Area –	Curvature $O(1)$
VTK MC	1.94	0.93	2.00	–3.35
SnapMC	1.93	0.82	2.14	–0.29
Afront*	–0.06	0.80	1.93	–0.27
Macet ^{1,*}	0.98	–0.12	0.29	–2.41
Macet ^{2,*}	0.03	0.75	2.02	–0.61
DC ¹	1.02	–0.11	0.69	–2.08
DC ²	1.96	0.96	1.89	–0.15
DellIso	1.49	1.07	2.04	0.07

Table 1. Comparison between formal order of accuracy and observed order of accuracy using $f(x, y, z) = x^2 + y^2 + z^2 - 1$ as a manufactured solution and for different algorithms. ¹ indicates the original source code and ² our fixed version. * indicates that a high-order spline was used instead of a linear interpolation (Section 3).

and isosurface $f(x, y, z) = 0$ in the domain $D = [-4, 4]^3$. Let \tilde{S}_k be a simplicial complex that approximates S for a given discretization parameter k (cell size h for marching cubes-based methods, accuracy ρ for Afront and maximum edge size t for DellIso).

The order of accuracy for VTK Marching Cubes, SnapMC, Macet and Dual Contouring depends on the cell size h . We run our tests with grid refinement $h_{i+1} = h_i/2$ and initial condition h_1 . For Afront, the order of accuracy depends on parameter ρ thus the refinement is given by $\rho_{i+1} = \rho_i/2$ with initial condition ρ_1 . Our customized version of DellIso has an additional parameter t that controls the largest edge on the output mesh. In this case, the refinement formula is $t_{i+1} = t_i/2$. In the particular case of SnapMC, we set the snap parameter γ to its maximum value ($\gamma = 1/2$). Even though the manufactured solution we selected is about as simple as can be imagined, comparing the formal order of accuracy with the observed one was enough to suggest bugs in two implementations. The observed order of accuracy of the examined properties is presented on Table 1.

4.1.1 Algebraic distance

Section 3.1 shows that one expects second-order convergence for function value on vertices if linear interpolation is used. We define the following approximation error on L_∞ norm:

$$E_k = \max_{j=1 \dots n} |\lambda - f(v_j)| \quad (10)$$

where λ is the isovalue of interest, v_j is a vertex of \tilde{S} and n the number of vertices. Figure 7(a) shows the vertex observed order of accuracy. VTK Marching Cubes, SnapMC have nearly quadratic convergence rates as shown in Figure 7(a). Afront shows a zero-order of accuracy though it presents very low error (in fact, the lowest in Figure 7(a)). This is due to the Catmull-Rom spline that is being used for surface approximation on the voxelized grid. Since it has cubic-order of accuracy, even for large values of ρ it can approximate with high precision the manufactured solution f . Next section shows that this is due to a poor choice for a manufactured solution. DellIso implementation has non-zero order of accuracy due to an outlier. Large values of t causes bad approximations of the manufactured solution.

The Macet and Dual Contouring curves suggest that the algorithms converge to a fixed value. In fact, there was indeed a problem in the implementation that was affecting the convergence of Macet and Dual Contouring (specifically, we found a hard-coded limit in the number of steps in a root-finding procedure that was being triggered by the high resolution of the volume). Once fixed, we obtain the results shown in Figure 8(a). Macet and Afront now have similar behavior in the observed order of accuracy of vertex position (Figure 8(a)). This is because both methods use high-order interpolation with splines, not linear interpolation as assumed before (see Section 5).

4.1.2 Normals

Section 3.2 shows that one expects first-order of accuracy for normal computations. We define the following approximation error using L_∞ norm:

$$E_k = \max_{j=1 \dots n} |\theta_{\sigma_j}| \quad (11)$$

where θ_{σ_j} is the angle between the normal of the triangle σ_j and the normal of the point in S closest to the centroid of σ_j . As shown in Figure 7(b), VTK Marching Cubes, Afront, SnapMC and DellIso have good observed order of accuracy above 0.8. However, only VTK

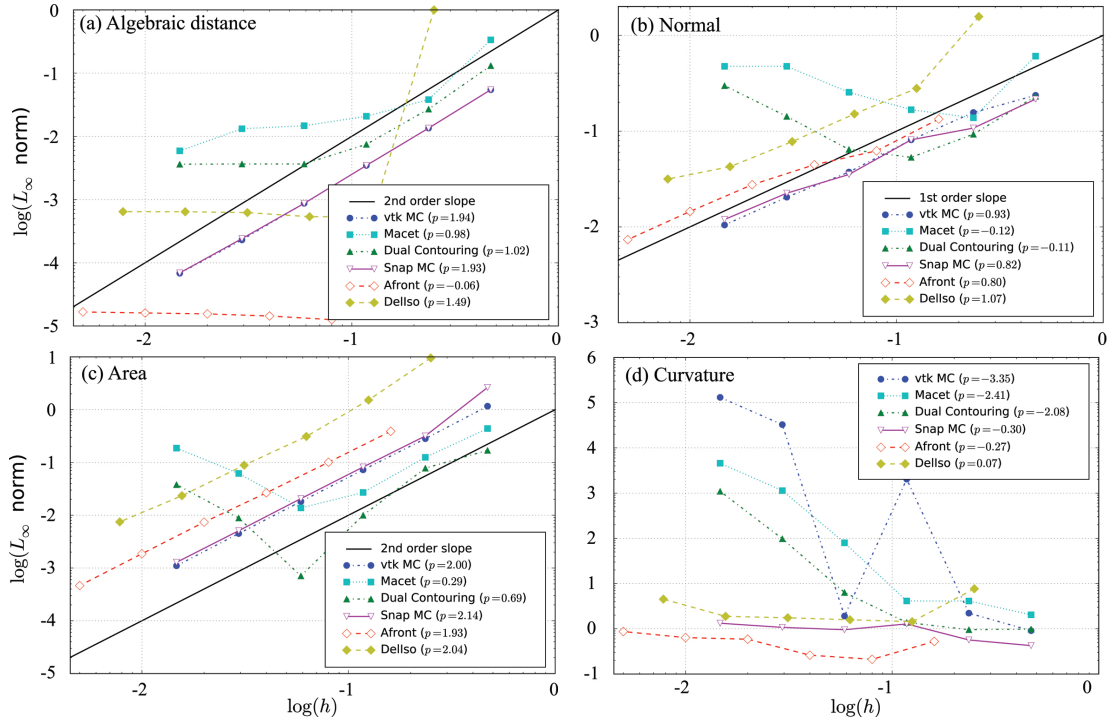


Fig. 7. Observed order of accuracy. The implementations of Macet and Dual Contouring have a bug that causes the deviation on errors. The black continuous line represents the expected behavior. p is the slope of the linear regression for each curve.

Marching Cubes and Dellso present close proximity to linear. Macet and Dual Contouring once again do not present a consistent order. Figure 8(b) shows the results after fixing both codes.

4.1.3 Area

Although there is no formal order of accuracy for area, one expects *some* convergence for it (Section 3.3). We define the following approximation error:

$$E_k = |A(S) - A(\tilde{S}_k)| \quad (12)$$

where A is the area function of a continuous or piecewise-linear surface. The results are shown in Figure 7(c). VTK Marching Cubes, Afront and Dellso present second-order of accuracy as shown in Figure 7(c). SnapMC accuracy is slightly better than quadratic due to poor approximation for large h . The error dropped faster than quadratic when the grid was refined for the first time. Macet and Dual Contouring exhibit once again unexpected behavior. Unlike the previous time, the curves now seem to diverge when h is too small. Once the bug is fixed the convergence curves changes, and they become quadratic (Figure 8(c)).

4.1.4 Curvature

Section 3.4 shows that one expects zero-th order of accuracy for curvature computation. We define the approximation error using L_∞ norm:

$$E_k = \max_{j=1 \dots n} |K(v_j) - \tilde{K}(v_j)| \quad (13)$$

where $K(v)$ is the Gaussian curvature at $v \in S$ and $\tilde{K}(v)$ is the Gaussian curvature at $v \in \tilde{S}$. In this particular case where S is a sphere, $K(v) = 1$ for every $v \in S$. The results are shown in Figure 7(d). Dellso, Afront and SnapMC are close to zeroth-order accuracy. The curvature order of accuracy for VTK Marching Cubes, on the other hand, diverges significantly. This unexpected behavior might deserve further investigation which we leave for future work. Although the curves shown in Figure 7(d) for Macet and Dual Contouring diverge, they change after fixing the code (Figure 8(d)).

4.2 Detected Bugs

We were able to find and fix bugs in two of the implementations under verification, namely, Macet and Dual Contouring, using as manufactured solution a sphere centered at origin with radius 1. The new result curves are shown in Figure 8. The observed order of accuracy for Dual

Bug #1	Bug #2	Quality	Observed accuracy
No Fix	No Fix	Good	Bad
Fix 1	No Fix	Good	Bad
Fix 1	Fixed	Bad	Good
Fix 2	No Fix	Good	Bad
Fix 2	Fixed	Bad	Good

Table 2. Table of results for Macet. Triangle quality versus convergence. We were not able to find a solution that provides both triangle quality and convergence.

Contouring is quite satisfactory for all manufactured solution. In particular, the normal order of accuracy has the best rate among the methods. Macet improved for its results for area. On the other hand, it still has some issues related to normals, which perhaps indicates a need for more tests and verification. The new order of accuracy for algebraic distance (Figure 8(a)) does not tell us much about the correctness of the code because of the zero-th order of accuracy (same for Afront).

The zero-th order of accuracy might happen if the formal order of accuracy is zero-th order, in which case the observed order matches the formal order. It might also happen due to a poor choice for manufactured solution. If it is not complex enough, the implementation being tested may approximate exactly the solution and therefore there is no error within the approximation although another error source (truncation error, for instance) may show up. The next section presents a detailed discussion concerning MMS.

Although we managed to fix the Macet convergence problem, we were not able to do so in a way that preserves triangle quality (Figure 1). Two were the problems we found in the source code, and we proposed two solutions for one of them. Table 2 shows that we could not find any combination that both fixed the convergence problem and preserved the triangle quality simultaneously. This sort of behavior raises the question if there is a theoretical problem that prevents both from being satisfied simultaneously, or it is just a matter of finding a better algorithmic fix. In both cases, further study and subsequent tests must be accomplished.

5 DISCUSSION

As we have shown, MMS is an effective means of diagnosing problems within the algorithms and implementations of isosurface extraction algorithms. In this work we have considered the two – algorithm

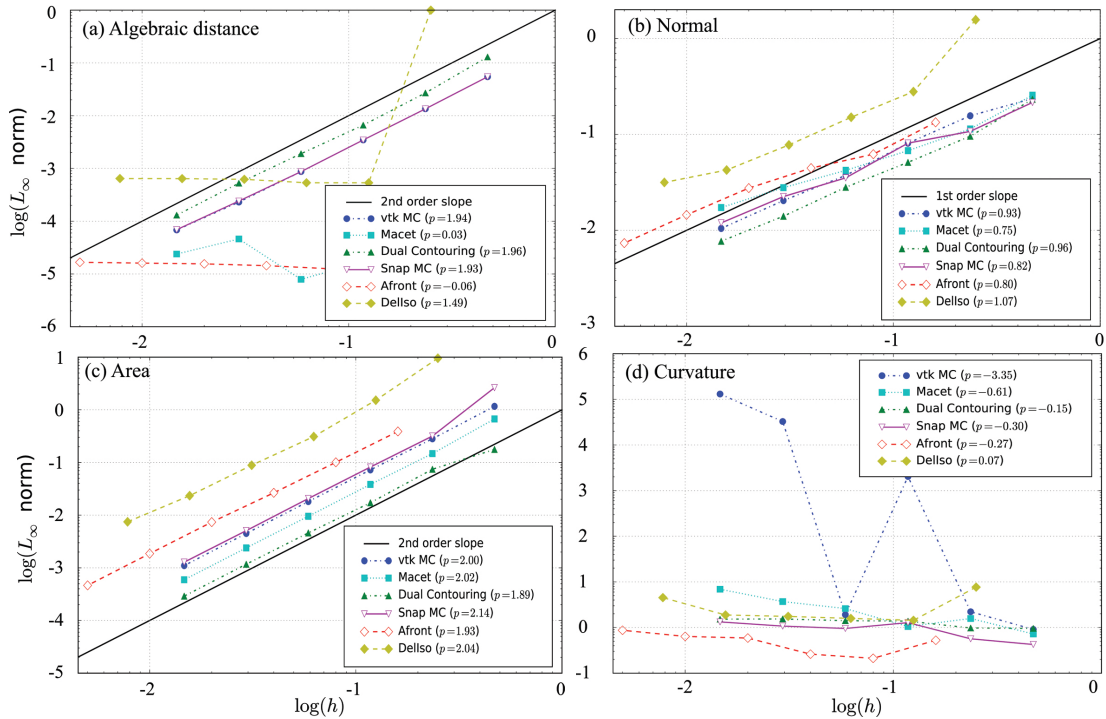


Fig. 8. Observed order of accuracy after fixing Macet and Dual Contouring code (other curves remain the same). The black continuous line represents the expected behavior. p is the slope of the linear regression for each curve.

and implementation – as one unit as one cannot always distinguish between the two if only limited information (source code and algorithmic details) is available. In this section, we present a more thorough discussion of the use of MMS, particularly for isosurface extraction.

On the implementation and use of MMS. One of the primary advantages of verifying simulation codes using MMS is that it is a non-intrusive method. MMS treats the code being verified as a blackbox, and so can be easily integrated into an existing test suite with little to no impact. However, MMS does not “see” the implementation, and so provides little direct information about where a particular bug might be when there is a discrepancy between the formal and observed orders of accuracy. In our experience, there are three main places where mistakes can happen: (1) in the design and construction of the manufactured solution, (2) in the coding of the algorithm being tested, and (3) in the evaluation and interpretation of the results. Mistakes on the evaluation of results have two flavors: misinterpretation or poor formal order of accuracy. The first heavily depends on testers’ and experts’ experience and ability to judge what a good result is. For example, should the normal observed order of accuracy for Afront and Macet on Figure 7(b) be considered linear ($p = 0.80$ and $p = 0.75$ respectively)? The latter depends on a rigorous formal order of accuracy analysis of the algorithm considering all sorts of errors; even round-off errors may be significant. In fact, we spent more time on writing out rigorously the analysis of the formal order of accuracy and on searching for possible sources of error than on the tests themselves. This again highlights the fact that verification using MMS is a process: it is typical to go back to the white board and refine formal analyses before arriving at conclusive answers. Although the formal order of accuracy analysis might be a painful process, the literature has many results that can be promptly used. As a consequence, if one wishes to write his own MC technique, for instance, his only concern is to write a test which exploits the results available within the literature.

On the complexity of the manufactured solution. The complexity of the manufactured solution can have a large influence on the effectiveness of verification. Suppose one chooses the manufactured solution to be $f(x, y, z) = x + y + k$, k constant, instead of a sphere. Since MC-based techniques use linear interpolation, one expects the approximation to be exact regardless of any discretization parameter h , i.e., $p = 0$ (notice that the evaluated error might be non-zero, implying there is some other error source that does not depend on h). Since such a func-

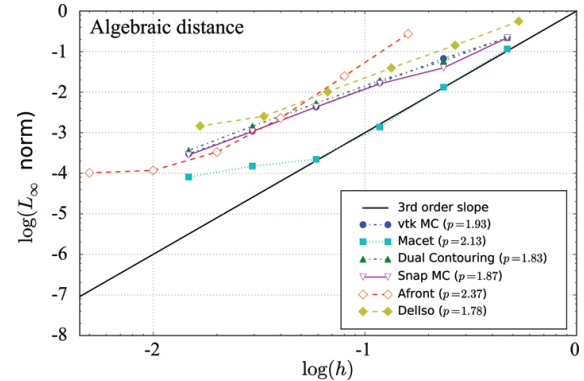


Fig. 9. Order of accuracy for a transcendental function $f(x, y, z) = x^2 + y^2 + z^2 + \cos(Ax)^2 + \cos(Ay)^2 + \cos(Az)^2$, A is a constant. The observed orders of accuracy for all implementations are relative to the voxel size h . We expect third-order accuracy for Afront and Macet due to their use of high-order spline approximations. Both have the expected convergence rate for all but the last two values.

tion f is extremely simple, it might not trigger bugs that would otherwise reduce the observed order of accuracy. In our experiments, the (problematic) implementation of Dual Contouring achieved the formal order of accuracy for this particularly simple function ($p = 0$).

Another example on the influence of manufactured solution arose with in our examination of Afront. Because Afront uses Catmull-Rom splines, some simple isosurfaces will converge to within numerical error for very rough volumes, and the numerically observed order of accuracy will be much lower than expected. With an implicit function whose isosurfaces are spheres, we observed zero-th order of accuracy for Afront for algebraic distance. With a modified implicit function that included transcendental functions, MMS reveals that Afront does not have the expected convergence rate on the full interval, as shown in Figure 9. Notice that Macet has similar behavior. Additional tests are needed to determine the source of this behavior within both codes.

On the order of accuracy. In this paper, we have chosen to make our formal analysis as generic as possible to accommodate as many implementations under verification as possible. Although we are able to evaluate many codes using the same manufactured solution, when

using MMS for a particular code, it is best to exploit as much detail about the algorithm as necessary. If the goal is to design a manufactured solution for verifying Marching Cubes-based techniques the manufactured solution should exercise all possible cases. Additionally, particular aspects of the manufactured solutions can be incorporated into the formal analysis. For example, the analysis for Afront becomes much more complicated if curvatures are not constant over the surface (in that case, its additional parameter η comes into play [23], and accurately bounding the triangle size is not practical).

The errors in Section 4.1 were measured at different locations on the mesh. Vertex convergence and Gaussian curvature were measured on triangle vertices, while normals were measured on the triangle centroid. More importantly, measurements performed at different locations may have different orders of accuracy. For example, Macet has cubic formal order of accuracy on vertices due to the spline approximation but quadratic formal order of accuracy on centroids.

In Section 3, we define the error using a pessimistic L_∞ norm. This makes MMS a very sensitive technique. In fact, it can detect subtle off-by-one mistakes in grid sizes and interactions between node-centric and cell-centric reconstructions, even for simple manufactured solutions. In these cases, it is important not to infer incorrect conclusions.

The numerical estimates for MMS should be performed on as wide a range of parameter values as possible. In our tests, we used $h \in (0.001, 1.0)$ and observed that both faulty implementations performed appropriately for large values of h . Just as the implementations might only enter the asymptotic regime and achieve the formal convergences for small values of h , it might be that (as we have experienced) bugs only manifest themselves on sufficiently small values of h .

On the limitations of the test. MMS does not cover every aspect of verification for isosurface extraction. For example, an important aspect we do not know how to test with MMS is the topological correctness of an extracted mesh. This is challenging because there does not seem to be a good measure of convergence for topological properties such as the Euler characteristic or Betti numbers. A proper study of these issues is a natural avenue for future work.

6 CONCLUSIONS AND FUTURE WORK

Because of its simplicity and effectiveness, we believe MMS could become a standard tool in the verification of scientific visualization software in the same that it has been adopted by the scientific simulation community as a trustworthy tool for assess code correctness. Using a simple manufactured solution, we were able to reveal bugs that prevented the convergence of some mesh properties of two publicly available isosurfacing codes. In particular, the by-products of the verification process, namely a continuous refinement of mathematical analysis of the algorithm's behavior and a numerical comparison of the results of the implementation against a known solution are valuable in their own right, and should be published together with new algorithms.

We are investigating the applicability of MMS to other visualization techniques such as streamline generation and volume rendering. In particular, MMS should clarify assumptions and errors intrinsic in these visualizations, a topic that has received recent attention [10]. More importantly, we hope the examples presented here will encourage the adoption of MMS by the visualization community at large, increasing the impact of its contributions to a wider audience.

ACKNOWLEDGMENTS

We thank Lauro Lins and Tom Peters for help with the paper, João Comba for help with the Macet code, and Tamal Dey and Joshua Levine for the customized version of DeIso used in this work. This work was supported in part by grants from NSF (grants IIS-0905385, IIS-0844546, ATM-0835821, CNS-0751152, OCE-0424602, CNS-0514485, IIS-0513692, CNS-0524096, CCF-0401498, OISE-0405402, CCF-0528201, CNS-0551724), DOE, IBM Faculty Awards and PhD Fellowship, the US ARO under grant W911NF0810517, ExxonMobil, and Fapesp-Brazil (#2008/03349-6).

REFERENCES

- [1] I. Babuska and J. Oden. Verification and validation in computational engineering and science: basic concepts. *Computer Methods in Applied Mechanics and Engineering*, 193(36-38):4057–4066, 2004.
- [2] P. Bhaniramka, R. Wenger, and R. Crawfis. Isosurface construction in any dimension using convex hulls. *IEEE TVCG*, 10(2):130–141, 2004.
- [3] H. Carr, T. Moller, and J. Snoeyink. Artifacts caused by simplicial subdivision. *IEEE TVCG*, 12(2):231–242, 2006.
- [4] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.
- [5] T. K. Dey and J. A. Levine. Delaunay meshing of isosurfaces. In *SMI '07: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, pages 241–250. IEEE Computer Society, 2007.
- [6] C. A. Dietrich, C. Scheidegger, J. Schreiner, J. L. D. Comba, L. P. Nedel, and C. Silva. Edge transformations for improving mesh quality of marching cubes. *IEEE TVCG*, 15(1):150–159, 2008.
- [7] S. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *IEEE Vol. Vis.*, pages 23–30, 1998.
- [8] A. Globus and S. Uselton. Evaluation of visualization software. *SIGGRAPH Comp. Graph.*, 29(2):41–44, 1995.
- [9] K. Hildebrandt, K. Polthier, and M. Wardetzky. On the convergence of metric and geometric properties of polyhedral surfaces. *Geometriae Dedicata*, (123):89–112, 2006.
- [10] C. R. Johnson and A. R. Sanderson. A next step: Visualizing errors and uncertainty. *IEEE CG&A*, 23(5):6–10, 2003.
- [11] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *SIGGRAPH '02*, pages 339–346. ACM, 2002.
- [12] R. Kirby and C. Silva. The need for verifiable visualization. *IEEE Computer Graphics and Applications*, 28(5):78–83, 2008.
- [13] L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *SIGGRAPH '01*, pages 57–66. ACM, 2001.
- [14] 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.
- [15] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comp. Graph.*, 21:163–169, 1987.
- [16] S. R. Marschner and R. J. Lobb. An evaluation of reconstruction filters for volume rendering. In *IEEE Visualization '94*, pages 100–107, 1994.
- [17] D. Meek and D. Walton. On surface normal and gaussian curvature approximation given data sampled from a smooth surface. *Computer-Aided Geometric Design*, 17:521–543, 2000.
- [18] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, 1993.
- [19] J. Patera and V. Skala. A comparison of fundamental methods for iso surface extraction. *Machine Graphics & Vision International Journal*, 13(4):329–343, 2004.
- [20] A. Pommert, U. Tiede, and K. Höhne. On the accuracy of isosurfaces in tomographic volume visualization. In *MICCAI'02*, pages 623–630, London, UK, 2002. Springer-Verlag.
- [21] S. Raman and R. Wenger. Quality isosurface mesh generation using an extended marching cubes lookup table. *Computer Graphics Forum*, 27(3):791–798, 2008.
- [22] C. J. Roy. Review of code and solution verification procedures for computational simulation. *J. Comput. Phys.*, 205(1):131–156, 2005.
- [23] J. Schreiner, C. Scheidegger, and C. Silva. High-quality extraction of isosurfaces from regular and irregular grids. *IEEE TVCG*, 12(5):1205–1212, 2006.
- [24] P. Sutton, C. Hansen, H.-W. Shen, and D. Schikore. A case study of isosurface extraction algorithm performance. In *Data Visualization 2000*, pages 259–268. Springer, 2000.
- [25] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE PAMI*, 16(3):287–303, Mar 1994.
- [26] M. Tory and T. Moeller. Human factors in visualization research. *IEEE TVCG*, 10(1):72–84, 2004.
- [27] G. Xu. Convergence analysis of a discretization scheme for gaussian curvature over triangular surfaces. *Comput. Aided Geom. Des.*, 23(2):193–207, 2006.
- [28] Z. Xu, G. Xu, and J.-G. Sun. Convergence analysis of discrete differential geometry operators over surfaces. In *Mathematics of Surfaces XI*, volume 3604 of *LNCS*, pages 448–457. Springer, 2005.
- [29] L. Zhou and A. Pang. Metrics and visualization tools for surface mesh comparison. In *Proc. SPIE - Visual Data Exploration and Analysis VIII*, volume 4302, pages 99–110, 2001.