Topology Verification for Isosurface Extraction

Tiago Etiene, L. Gustavo Nonato, Carlos Scheidegger, Julien Tierny, Thomas J. Peters, Valerio Pascucci, *Member*, *IEEE*, Robert M. Kirby, *Member*, *IEEE*, and Cláudio T. Silva, *Senior Member*, *IEEE*

Abstract—The broad goals of verifiable visualization rely on correct algorithmic implementations. We extend a framework for verification of isosurfacing implementations to check topological properties. Specifically, we use stratified Morse theory and digital topology to design algorithms which verify topological invariants. Our extended framework reveals unexpected behavior and coding mistakes in popular publicly available isosurface codes.

Index Terms—Verifiable visualization, isosurface, topology.

1 Introduction

Visualization is an important aspect of current large-scale data analysis. As the users of scientific software are not typically visualization experts, they might not be aware of limitations and properties of the underlying algorithms and visualization techniques. As visualization researchers and practitioners, it is our responsibility to ensure that these limitations and properties are clearly stated and studied. Moreover, we should provide mechanisms which attest to the correctness of visualization systems. Unfortunately, the accuracy, reliability, and robustness of visualization algorithms and their implementations have not in general fallen under such scrutiny as have other components of the scientific computing pipeline.

The main goal of verifiable visualization is to increase confidence in visualization tools [19]. Verifiable visualization tries to develop systematic mechanisms for identifying and correcting errors in both algorithms and implementations of visualization techniques. As an example, consider a recent scheme to check geometrical properties of isosurface extraction [15]. By writing down easily checkable convergence properties that the programs should exhibit, the authors identified bugs in isosurfacing codes that had gone undetected.

- *T. Etiene, V. Pascucci, R.M. Kirby and C.T. Silva are with the School of Computing and SCI Institute, University of Utah, 72 Central Campus Drive, Warnock Engineering Building, Salt Lake City, UT 84112-9200. E-mail: {tetiene, kirby}@cs.utah.edu, {pascucci, csilva}@sci.utah.edu.*
- *L.G. Nonato is with the Departamento de Matemática Aplicada e Estatística, Instituto de Ciências Matemáticas e de Comutação, Universidade de São Paulo, Av. Trabalhador São-carlense 400, São Carlos, SP 13560-970, Brazil. E-mail: gnonato@icmc.usp.br.*
- *C. Scheidegger is with the AT&T Labs—Research, 36 Kinney Street, Apt. 1, Madison, NJ 07940. E-mail: cscheid@research.att.com.*
- *J. Tierny is with the CNRS—LTCI—Telecom ParisTech, 46 Rue Barrault, Paris 75013, France. E-mail: tierny@telecom-paristech.fr.*
- *T.J. Peters is with Department of Computer Science and Engineering, U-2155 Department of Mathematics, University of Connecticut, Fairfield Way, Storrs, CT 06269-2155. E-mail: tpeters@cse.uconn.edu.*

Manuscript received 30 July 2010; revised 6 Apr. 2011; accepted 23 Apr. 2011; published online 16 June 2011.
Recommended for acceptance by T. Möller.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-07-0166.
Digital Object Identifier no. 10.1109/TVCG.2011.109.

We strive for verification tools which are both *simple* and *effective*. Simple verification methods are less likely to have bugs themselves, and effective methods make it difficult for bugs to hide. Alas, the mathematical properties of an algorithm and its implementation are both constructs of fallible human beings, and so perfection is an unattainable goal; there will always be the next bug. Verification is, fundamentally, a *process*, and when it finds problems with an algorithm or its implementation, we can only claim that the new implementation behaves more correctly than the old one. Nevertheless, the verification process clarifies *how* the implementations fail or succeed.

In this paper, we investigate isosurfacing algorithms and implementations and focus on their *topological properties*. For brevity, we will use the general phrase "isosurfacing" when we refer to both isosurfacing algorithms and their implementations. As a simple example, the topology of the output of isosurface codes should match that of the level set of the scalar field (as discussed in Section 3). Broadly speaking, we use the method of manufactured solutions (MMS) to check these properties. By manufacturing a model whose known behavior should be reproduced by the techniques under analysis, MMS can check whether they meet expectations.

Etiene et al. have recently used this method to verify geometrical properties of isosurfacing codes [15], and topological verification naturally follows. An important contribution of this paper is the selection of significant topological characteristics that can be verified by software methods. We use results from two fields in computational topology, namely, digital topology and stratified Morse theory (SMT).

In summary, the main contributions of this work can be stated as follows:

1. In the spirit of verifiable visualization, we introduce a methodology for checking topological properties of publicly and commercially available isosurfacing software.
2. We show how to adapt techniques from digital topology to yield simple and effective verification tools for isosurfaces without boundaries.

3. We introduce a simple technique to compute the Euler characteristic of a level set of a trilinearly interpolated scalar field. The technique relies on stratified Morse theory and allows us to verify topological properties of isosurfaces with boundaries.
4. We propose a mechanism to manufacture isosurfaces with nontrivial topological properties, showing that this simple mechanism effectively stresses isosurfacing programs. As input, we also assume a piecewise trilinear scalar field defined on a regular grid.

The verification process produces a comprehensive record of the desired properties of the implementations, along with an objective assessment of whether these properties are satisfied. This record improves the applicability of the technique and increases the value of visualization. We present a set of results obtained using our method, and we report errors in two publicly available isosurface extraction codes.

2 Related Work

The literature that evaluates isosurface extraction techniques is enormous, with works ranging from mesh quality [12], [34], [37], to performance [40] and accuracy analysis [33], [43]. In this section, we focus on methods that deal with topological issues that naturally appear in isosurfacing.

2.1 Topology-Aware Isosurfacing

Arguably the most popular isosurface extraction technique, Marching Cubes [23] (MC) processes one grid cell at a time and uses the *signs* of each grid node (whether the scalar field at the node is above or below the isovalue) to fit a triangular mesh that approximates the isosurface within the cell. As no information besides the signs is taken into account, Marching Cubes cannot guarantee any topological equivalence between the triangulated mesh and the original isosurface. In fact, the original Marching Cubes algorithm would produce surfaces with "cracks," caused by alternating vertex signs along a face boundary, which lead to contradicting triangulations in neighboring cells [31]. Disambiguation mechanisms can ensure crack-free surfaces, and many schemes have been proposed, such as the one by Montani et al. [26], domain tetrahedralization [4], preferred polarity [2], gradient-based method [41], and feature-based schemes [18]. The survey of Newman and Yi has a comprehensive account [29]. Although disambiguation prevents cracks in the output, it does not guarantee topological equivalence.

Topological equivalence between the resulting triangle mesh and the isosurface can only be achieved with additional information about the underlying scalar field. Since function values on grid nodes are typically the only information provided, a reconstruction kernel is assumed, of which trilinear reconstruction on regular hexahedral grids is most popular [30]. Nielson and Hamann, for example, use saddle points of the bilinear interpolant on grid cell faces [31]. Their method cannot always reproduce the topology of trilinear interpolation because there remain ambiguities internal to a grid cell: pairs of nonhomeomorphic isosurfaces could be homeomorphic when restricted to the grid cell faces. This problem has been recognized by Natarajan [28] and Chernyaev [8], leading to new classification and triangulation schemes. This line of work has inspired many other "topology-aware" triangulation methods, such as Cignoni et al.'s reconstruction technique [9]. Subsequent work by Lopes and Brodlie [22] and Lewiner et al. [21] has finally provided triangulation patterns covering all possible topological configurations of trilinear functions, implicitly promising a crack-free surface. The topology of the level sets generated by trilinear interpolation has been recently studied by Carr and Snoeyink [5], and Carr and Max [3]. A discussion about these can be found in Section 4.2.

2.2 Verifiable Visualization

Many of the false steps in the route from the original MC algorithm to the recent homeomorphic solutions could have been avoided with a systematic procedure to verify the algorithms and the corresponding implementations. Although the lack of verification of visualization techniques and the corresponding software implementations has been a long-term concern of the visualization community [16], [19], concrete proposals on verification are relatively recent. Etiene et al. [15] were among the first in scientific visualization to propose a practical verification framework for geometrical properties of isosurfacing. Their work is based on the method of MMS, a popular approach for assessing numerical software [1]. We are interested in *topological properties* of isosurfacing, and we also use MMS as a verification mechanism. As we will show in Section 6, our proposed technique discovered problems in popular software, supporting our assertion about the value of a broader culture of verification in scientific visualization.

There have been significant theoretical investigations in computational topology dealing with, for example, isosurface invariants, persistence, and stability [10], [13]. This body of work is concerned with how to define and compute topological properties of computational objects. We instead develop methods that stress topological properties of isosurfacing. These goals are complementary. Computational topology tools for data analysis might offer new properties which can be used for verification purposes, and verification tools can assess the correctness of the computational topology implementations. Although the mechanism we propose to compute topological invariants for piecewise smooth scalar fields is, to the best of our knowledge, novel (see Section 4.2), our primary goal is to present a method that developers can adapt to assess their own software.

3 Verifying Isosurface Topology

We now discuss strategies for verifying topological properties of isosurfacing techniques. We start by observing that simply stating the desired properties of the implementation is valuable. Consider a typical implementation of Marching Cubes. How would you debug it? Without a small set of desired properties, we are mostly limited to inspecting the output by explicitly exercising every case in the case table. The fifteen cases might not seem daunting, but what if we suspect a bug in symmetry reduction? We now have 256 cases to check. Even worse, what if the bug is in a combination of separate cases along neighboring cells? The verification would grow to be at

least as complicated as the original algorithm, and we would just as likely make a mistake during the verification as we would in the implementation. Therefore, we need properties that are simple to state, easy to check, and good at catching bugs.

Simple example. Although the previously mentioned problem with Marching Cubes [23] is well known, it is not immediately clear what topological properties fail to hold. For example, "the output of Marching Cubes cannot contain boundary curves" is not one such property, for two reasons. First, some valid surfaces generated by Marching Cubes—such as with the simple 2^3 case—do contain boundaries. Second, many incorrect outputs might not contain any boundaries at all. The following might appear to be a good candidate property: "given a positive vertex v_0 and a negative vertex v_1, any path through the scalar field should intersect the isosurface an odd number of times." This property *does* capture the fact that the triangle mesh should separate interior vertices from exterior vertices and seems to isolate the problem with the cracks. Checking this property, on the other hand, and even stating it precisely, is problematic. Geometrical algorithms for intersection tests are notoriously brittle; for example, some paths might intersect the isosurface in degenerate ways. A more promising approach comes from noticing that any such separating isosurface has to be a piecewise-linear (PL) manifold, whose boundary must be a subset of the boundary of the grid. This directly suggests that "the output of Marching Cubes must be a PL manifold whose boundaries are contained in the boundary of the grid." This property is simple to state and easy to test: the link of every interior vertex in a PL manifold is topologically a circle, and the link of every boundary vertex is a line. The term "consistency" has been used to describe problems with some algorithms [29]. In this paper, we say that the output of an algorithm is *consistent* if it obeys the PL manifold property above. By generating arbitrary grids and extracting isosurfaces with arbitrary isovalues, the inconsistency of the original case table becomes mechanically checkable and instantly apparent. Some modifications to the basic Marching Cubes table, such as using Nielson and Hamann's asymptotic decider [31], result in consistent implementations, and the outputs pass the PL manifold checks (as we will show in Section 6).

The example we have presented above is a complete instance of the method of manufactured solutions. We identify a property that the results should obey, run the implementations on inputs, and test whether the resulting outputs respect the properties. In the next sections, we develop a verification method for algorithms to reproduce the topology of the level sets of trilinear interpolation [8], [22], [30], thus completely eliminating any ambiguity. In this paper, we say the output is *correct* if it is homeomorphic to the corresponding level set of the scalar field. This correctness property is simple to state, but developing effective verification schemes that are powerful and simple to implement is more involved. We will turn to invariants of topological spaces, in particular to Betti numbers and the Euler characteristic, their relative strengths and weaknesses, and discuss how to robustly check their values. Fig. 1 shows our pipeline to assess topological correctness and also the paper organization.

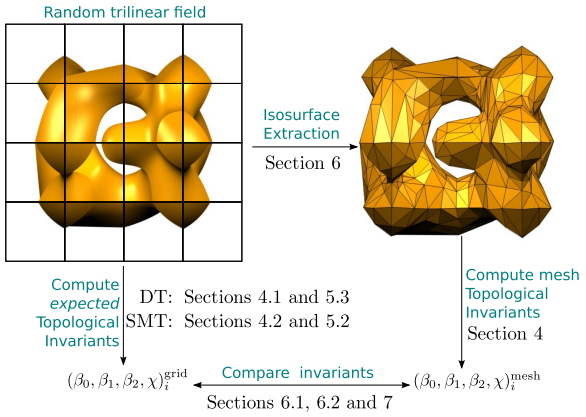


Fig. 1. Overview of our topology verification pipeline. First step, we generate a random trilinear field and extract a random isosurface using the implementation under verification. We then compute the *expected* topological invariants from the trilinear field and compare them against the invariants obtained from the mesh. We provide two simple ways to compute topological invariants from a trilinear field based on digital topology (DT) or stratified Morse theory.

4 Mathematical Tools

This section describes the mathematical machinery used to derive the topology verification tools. More specifically, we provide a summary of the results we need from digital topology and stratified Morse theory. A detailed discussion on digital topology can be found in Stelldinger et al.'s paper [39], and Goresky and MacPherson give a comprehensive presentation of stratified Morse theory [17].

In Section 4.1, we describe a method, based on digital topology, that operates on manifold surfaces without boundaries and determines the Euler characteristic and Betti numbers of the level sets. A more general setting of surfaces with boundaries is handled with tools derived from stratified Morse theory, detailed in Section 4.2. The latter method can only determine the Euler characteristic of the level set.

Let us start by recalling the definition and some properties of the Euler characteristic, which we denote by χ. For a compact 2-manifold $\mathcal{M}$, $\chi(\mathcal{M}) = V - E + F$, where V, E, and F are the number of vertices, edges, and faces of any finite cell decomposition of $\mathcal{M}$. If $\mathcal{M}$ is a connected orientable 2-manifold without boundary, $\chi(\mathcal{M}) = 2 - 2g(\mathcal{M})$, where $g(\mathcal{M})$ is the genus of $\mathcal{M}$. The Euler characteristic may also be written as $\chi(\mathcal{M}) = \sum_{i=0}^{n}(-1)^i\beta_i$, where β_i are the Betti numbers: the rank of the ith homology group of $\mathcal{M}$. Intuitively, for 2-manifolds, β_0, β_1, and β_2 correspond to the number of connected components, holes and voids (regions of the space enclosed by the surface), respectively. If $\mathcal{M}$ has many distinct connected components, that is, $\mathcal{M} = \bigcup_{i=1}^{n}\mathcal{M}^i$ and $\mathcal{M}^i \bigcap \mathcal{M}^j = \emptyset$ for $i \neq j$ then $\chi(\mathcal{M}) = \sum_i^n \chi(\mathcal{M}^i)$. More details about Betti numbers, the Euler characteristic, and homology groups can be found in Edelsbrunner and Harer's text [13]. The Euler characteristic and the Betti numbers are topological invariants: two homeomorphic topological spaces will have the same Euler characteristic and Betti numbers whenever these are well defined.

4.1 Digital Topology

Let $\mathcal{G}$ be an $n \times n \times n$ cubic regular grid with a scalar $e(s)$ assigned to each vertex s of $\mathcal{G}$ and $t : \mathbb{R}^3 \to \mathbb{R}$ be the

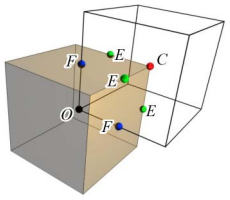


Fig. 2. The four distinct groups of vertices O, F, E, C, are depicted as black, blue, green, and red points. They are the "Old," "Face," "Edge," and "Corner" points of a voxel region $V_{\mathcal{G}}$ (semitransparent cube), respectively. For the sake of clarity, we only show a few points.

piecewise trilinear interpolation function in $\mathcal{G}$, that is, $t(x) = t_i(x)$, where t_i is the trilinear interpolant in the cubic cell c_i containing x. Given a scalar value α, the set of points satisfying $t(x) = \alpha$ is called the *isosurface* α of t. In what follows, $t(x) = \alpha$ will be considered a compact, orientable 2-manifold without boundary. We say that a cubic cell c_i of $\mathcal{G}$ is *unambiguous* if the following two conditions hold simultaneously:

1. Any two vertices s_a and s_b in c_i for which $e(s_a) < \alpha$ and $e(s_b) < \alpha$ are connected by *negative edges*, i.e., a sequence of edges $s_a s_1, s_1 s_2, \ldots, s_k s_b$ in c_i whose vertices satisfy $e(s_i) < \alpha$ for $i = 1, \ldots, k$.
2. Any two vertices s_c and s_d in c_i for which $e(s_c) > \alpha$ and $e(s_d) > \alpha$ are connected by *positive edges*, i.e., a sequence of edges $s_c s_1, s_1 s_2, \ldots, s_l s_d$ in c_i whose vertices satisfy $e(s_i) > \alpha$ for $i = 1, \ldots, l$.

In other words, a cell is unambiguous if all positive vertices form a single connected component via the positive edges and, conversely, all negative vertices form a single connected component by negative edges [41]. If either property fails to hold, c_i is called *ambiguous*. The top row in Fig. 3 shows all possible unambiguous cases.

The geometric dual of $\mathcal{G}$ is called the *voxel grid* associated with $\mathcal{G}$, denoted by $V_{\mathcal{G}}$. More specifically, each vertex s of $\mathcal{G}$ has a corresponding voxel v_s in $V_{\mathcal{G}}$, each edge of $\mathcal{G}$ corresponds to a face in $V_{\mathcal{G}}$ (and vice versa), and each cubic cell in $\mathcal{G}$ corresponds to a vertex in $V_{\mathcal{G}}$, as illustrated in Fig. 2. Each voxel v_s can also be seen as the Voronoi cell associated with s. Scalars defined in the vertices of $\mathcal{G}$ can naturally be extended to voxels, thus ensuring a single scalar value $e(v_s)$ to each voxel v_s in $V_{\mathcal{G}}$ defined as $e(s) = e(v_s)$. As we shall show, the voxel grid structure plays an important role when using digital topology to compute topological invariants of a given isosurface. Before showing that relation, though, we need a few more definitions.

Denote by $\mathcal{G}'$ the $(2n-1) \times (2n-1) \times (2n-1)$ regular grid is obtained from a refinement of $\mathcal{G}$. Vertices of $\mathcal{G}'$ can be grouped in four distinct sets, denoted by O, F, E, C. The set O contains the vertices of $\mathcal{G}'$ that are also vertices of $\mathcal{G}$. The sets F and E contain the vertices of $\mathcal{G}'$ lying on the center of faces and edges of the voxel grid $V_{\mathcal{G}}$, respectively. Finally, C contains all vertices of $V_{\mathcal{G}}$. Fig. 2 illustrates these sets.

Consider now the voxel grid $V_{\mathcal{G}'}$ dual to the refined grid $\mathcal{G}'$. Given a scalar value α, the *digital object* $\mathcal{O}_\alpha$ is the subset of voxels v in $V_{\mathcal{G}'}$ such that $v \in \mathcal{O}_\alpha$ if at least one of the criteria below are satisfied:

- $v \in O$ and $e(v) \leq \alpha$.
- $v \in F$ and both neighbors of v in O have scalars less than (or equal to) α.
- $v \in E$ and at least 4 of the 8 neighbors of v in $O \cup F$ have scalars less than (or equal) α.
- $v \in C$ and at least 12 of the 26 neighbors of v in $O \cup F \cup E$ have scalars less than (or equal) α.

The description above is called MI (Fig. 4), and it allows us to compute the voxels that belong to a digital object $\mathcal{O}_\alpha$. The middle row of Fig. 3 shows all possible cases for voxels picked by the MI algorithm (notice the correspondence with the top row of the same figure).

The importance of $\mathcal{O}_\alpha$ is two-fold. First, the boundary surface of the union of the voxels in $\mathcal{O}_\alpha$, denoted by $\partial\mathcal{O}_\alpha$ and called a *digital surface* (DS), is a 2-manifold (see the proof by Stelldinger et al. [39]). Second, the genus of $\partial\mathcal{O}_\alpha$ can be computed directly from $\mathcal{O}_\alpha$ using the algorithm proposed by Chen and Rong [7] (Fig. 5). As the connected components of $\mathcal{O}_\alpha$ can also be easily computed and isolated, one can calculate

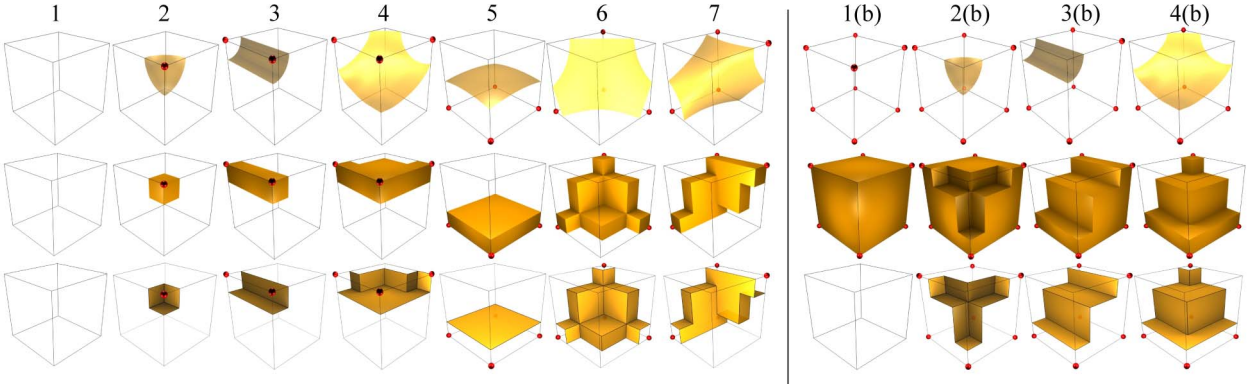


Fig. 3. An illustration of the relation between unambiguous isosurfaces of trilinear interpolants and the corresponding digital surfaces. The top row shows all possible configurations of the intersection of $t = \alpha$ with a cube c_j for unambiguous configurations [22]. Each red dot s_i denotes a vertex with $e(s_i) < \alpha$. Each image on the top right is the complement $\bar{c}_i$ of cases 1 to 4 on the left (cases 5 to 7 were omitted because the complement is identical to the original cube up to symmetry). The middle row shows the volume reconstructed by Majority Interpolation (MI) for configurations 1 to 7 (left) and the complements (right) depicted in the top row. Bottom row shows the boundary of the volume reconstructed by the MI algorithm (the role of faces that intersect c_i is explained in the proof of Theorem 4.1). Notice that all surfaces in the top and bottom rows are topological disks. For each cube configuration, the boundary of each digital reconstruction (bottom row) has the same set of positive/negative connected components as the unambiguous configurations (top row).

MAJORITYINTERPOLATION($\mathcal{G}, \alpha$)
1 ▷ Let O, F, E and C be the subset of vertices in $\mathcal{G}'$ as described in subsection 4.1.
2 ▷ Let $\mathcal{N}(s, \star)$ be the set of neighbors of $s \in \mathcal{G}'$ in the set $\star$, where $\star = \{O, F, E, C\}$, with associate scalar less than α
3 **for** $s \in \mathcal{G}'$
4 **do if** $s \in O$ **or**
5 $s \in$ F and $|\mathcal{N}(s, O)| = 2$ **or**
6 $s \in E$ and $|\mathcal{N}(s, O) + \mathcal{N}(s, F)| \geqslant 4$ **or**
7 $s \in$ C and $|\mathcal{N}(s, O) + \mathcal{N}(s, F) + \mathcal{N}(s, E)| \geqslant 12$
8 **then** Select voxel v_s
9 **return** $\mathcal{O}_\alpha$

Fig. 4. Voxel selection using Majority Interpolation.

the Euler characteristic of each connected component of $\mathcal{O}_\alpha$ from the formula $\chi = 2 - 2g$ and thus β_0, β_1, and β_2.

The voxel grid $V_{\mathcal{G}'}$ described above allows us to compute topological invariants for any digital surface $\partial\mathcal{O}_\alpha$. However, we so far do not have any result relating $\partial\mathcal{O}_\alpha$ to the isosurface $t(x) = \alpha$. The next theorem provides the connection.

Theorem 4.1. *Let $\mathcal{G}$ be an $n \times n \times n$ rectilinear grid with scalars associated with each vertex of $\mathcal{G}$ and t be the piecewise trilinear function defined on $\mathcal{G}$, such that the isosurface $t(x) = \alpha$ is a 2-manifold without boundary. If no cubic cell of $\mathcal{G}$ is ambiguous with respect to $t(x) = \alpha$, then $\partial\mathcal{O}_\alpha$ is homeomorphic to the isosurface $t(x) = \alpha$.*

Proof. Given a cube $c_i \subset \mathcal{G}$ and an isosurface $t = \{x \mid t(x) = \alpha\}$, let $t_i = t \cap c_i$. Similarly, denote

$$\partial\mathcal{O}_i = cl_{\mathbb{R}^3}((\partial\mathcal{O}_\alpha \cap c_i) - \partial c_i),$$

where $cl_{\mathbb{R}^3}$ denotes the closure operator. We note that $\partial\mathcal{O}_i$ is a 2-manifold for all i [35], [39]. There are two main parts to the proof presented here. For each i,

1. the 2-manifolds t_i and $\partial\mathcal{O}_i$ are homeomorphic; and
2. both t_i and $\partial\mathcal{O}_i$ cut the same edges and faces of c_i.

Since t is trilinear, no level-set of t can intersect an edge more than once. Hence, if c_i is not ambiguous, t_i is exactly one of the cases 1 to 7 in the top row of Fig. 3 [22], either a topological disk or the empty set. Each case in the top row of Fig. 3 is the unambiguous input for the MI algorithm to produce the voxel reconstruction shown in the middle row, where the boundaries of each of these voxel reconstructions are shown in the bottom row. By inspection, we can verify that the boundary of the digital reconstruction $\partial\mathcal{O}_i$ (bottom row of Fig. 3) is also a disk for all possible unambiguous cases and complement cases. Hence, for each i, the 2-manifolds $\partial\mathcal{O}_i$ and t_i are homeomorphic. Then, for each i, both $\partial\mathcal{O}_i$ and t_i cut the same set of edges and faces of c_i. Again, we can verify this for all possible i by inspecting the top and bottom rows in Fig. 3, respectively. Finally, we apply the Pasting Lemma [27] across neighboring surfaces $\partial\mathcal{O}_i$ and $\partial\mathcal{O}_j$ in order to establish the homeomorphism between $\partial\mathcal{O}_\alpha$ and t. □

GENUSFROMDS($\partial\mathcal{O}_\alpha$)
1 ▷ Let $\partial\mathcal{O}_\alpha$ be a 2-manifold without boundary
2 ▷ Let $|\mathcal{N}_i|$ be the number of surface points with exactly i neighbors.
3 ▷ Let g be the surface genus
4 $g = 1 + (|\mathcal{N}_5| + 2|\mathcal{N}_6| - |\mathcal{N}_3|)/8$
5 **return** g

Fig. 5. A simple formula for genus computation.

This proof provides a main ingredient for the verification method in Section 5. Crucially, we will show how to manufacture a complex solution that unambiguously crosses every cubic cell of the grid. Since we have shown the conditions for which the digital surfaces and the level sets are homeomorphic, any topological invariant will have to be the same for both surfaces.

4.2 Stratified Morse Theory

The mathematical developments presented above allow us to compute the Betti numbers of any isosurface of the piecewise trilinear interpolant. However, they require isosurfaces without boundaries. In this section, we provide a mechanism to compute the Euler characteristic of any regular isosurface of the piecewise trilinear interpolant through an analysis based on critical points, which can be used to verify properties of isosurfaces with boundary components. We will use some basic machinery from stratified Morse theory, following the presentation of Goresky and MacPherson's monograph [17].

Let f for now be a smooth function with isolated critical points p, where $\nabla f(p) = 0$. From classical Morse theory, the topology of two isosurfaces $f(x) = \alpha$ and $f(x) = \alpha + \epsilon$ differs only if the interval $[\alpha, \alpha + \epsilon]$ contains a critical value ($f(p)$ is a critical value iff p is a critical point). Moreover, if ε_p is a small neighborhood around p and $L^-(p)$ and $L^+(p)$ are the subset of points on the boundary of ε_p satisfying $f(x) < f(p)$ and $f(x) > f(p)$, respectively, then the topological change from the isosurface $f(x) = f(p) - \epsilon$ to $f(x) = f(p) + \epsilon$ is characterized by removing $L^-(p)$ and attaching $L^+(p)$. Thus, changes in the Euler characteristic, denoted by $\Delta\chi(p)$, are given by

$$\Delta\chi(p) = \chi(L^+(p)) - \chi(L^-(p)). \tag{4.1}$$

For a smooth function f, the number of negative eigenvalues of the Hessian matrix determines the index of a critical point p, and the four cases give the following values for $\chi(L^-(p))$ and $\chi(L^+(p))$.

	min	saddle-1	saddle-2	max
$\chi(L^-(p))$	0	2	0	2
$\chi(L^+(p))$	2	0	2	0

The above formulation is straightforward but unfortunately cannot be directly applied to functions appearing in either piecewise trilinear interpolations or isosurfaces with boundary, both of which appear in some of the isosurfacing algorithms with guaranteed topology. Trilinear interpolants are not smooth across the faces of grid cells, so the gradient is not well defined there. Identifying the critical points using smooth Morse theory is then problematic. Although

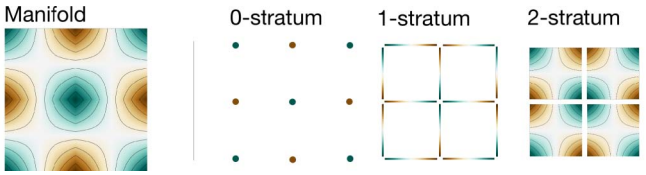


Fig. 6. An illustration of a piecewise-smooth immersed 2-manifold. The colormap illustrates the value of each point of the scalar field. Notice that although the manifold itself is not everywhere differentiable, each stratum is itself an open manifold that is differentiable.

arguments based on smooth Morse theory have appeared in the literature [42], there are complications. For example, the scalar field in a node of the regular grid might not have *any* partial derivatives. Although one can still argue about the intuitive concepts of minima and maxima around a nondifferentiable point, configurations such as saddles are more problematic, since their topological behavior is different depending on whether they are on the boundary of the domain. It is important, then, to have a mathematical tool which makes predictions regardless of the types of configurations, and SMT is one such theory.

Intuitively, a *stratification* is a partition of a piecewise-smooth manifold such that each subset, called a *stratum*, is either a set of discrete points or has a smooth structure. In a regular grid with cubic cells, the stratification we propose will be formed by four sets (the strata), each one a (possibly disconnected) manifold. The *vertex set* contains all vertices of the grid. The *edge set* contains all edge interiors, the *face set* contains all face interiors, and the *cell set* contains all cube interiors. We illustrate the concept for the 2D case in Fig. 6. The important property of the strata is that the level sets of f restricted to each stratum are smooth (or lack any differential structure, as in the vertex-set). In SMT, one applies standard Morse theory on each stratum, and then combines the partial results appropriately.

The set of points with zero gradient (computed on each stratum), which SMT assumes to be isolated, are called the *critical points* of the stratified Morse function. In addition, every point in the vertex set is considered critical as well. One major difference between SMT and the smooth theory is that some critical points do not actually change the topology of the level sets. This is why considering all grid vertices as critical does not introduce any practical problems: most grid vertices of typical scalar fields will be *virtual critical points*, i.e., points which do not change the Euler characteristic of the surface. Carr and Snoeyink use a related concept (which they call "potential critical points") in their state-machine description of the topology of interpolants [5].

Let $\mathcal{M}$ be the stratified grid described above. It can be shown that if p is a point in a d-dimensional stratum of $\mathcal{M}$, it is always possible to find a $(3-d)$-dimensional submanifold of $\mathcal{M}$ (which might straddle many strata) that meets transversely the stratum containing p, and whose intersection consists of only p (one way to think of this $(3-d)$-manifold is as a "topological orthogonal complement"). In this context, we can define a small neighborhood $T_\varepsilon(p)$ in the strata containing p and the *lower tangential* link $T_L^-(p)$ as the set of points in the boundary of $T_\varepsilon(p)$ with scalar values less than that in p.

Similarly, we can define the *upper tangential* link $T_L^+(p)$ as the set of points in the boundary of $T_\varepsilon(p)$ with scalar value higher than that at p. *Lower normal* $N_L^-(p)$ and *upper normal* $N_L^+(p)$ links are analogous notions, but the lower and upper links are taken to be subsets of $N_\varepsilon(p)$, itself a subset of the $(3-d)$-dimensional submanifold transverse to the stratum of p going through p. The definitions above are needed in order to define the *lower stratified link* and *upper stratified link*, as follows: given $T_\varepsilon(p)$, $T_L^-(p)$, $N_\varepsilon(p)$, and $N_L^-(p)$, the *lower stratified Morse link* (and similarly for upper stratified link) is given by

$$L^-(p) = (T_\varepsilon(p) \times N_L^-(p)) \cup (N_\varepsilon(p) \times T_L^-(p)). \tag{4.2}$$

These definitions allow us to classify critical points even in the nonsmooth scenario. They let us compute topological changes with the same methodology used in the smooth case. In other words, when a scalar value α crosses a critical value α_p in a critical point p, the topological change in the isosurface is characterized by removing $L^-(p)$ and attaching $L^+(p)$, affecting the Euler characteristic as defined in (4.1).

The remaining problem is how to determine $\chi(L^-(p))$ and $\chi(L^+(p))$. Recalling that $\chi(A \cup B) = \chi(A) + \chi(B) - \chi(A \cap B)$, $\chi(A \times B) = \chi(A)\chi(B)$, and $\chi(T_\varepsilon) = \chi(N_\varepsilon) = 1$ (we are omitting the point p) we have:

$$\begin{aligned} \chi(L^-) &= \chi(T_\varepsilon \times N_L^- \cup N_\varepsilon \times T_L^-) \\ &= \chi(N_L^-) + \chi(T_L^-) - \chi(T_\varepsilon \times N_L^- \cap N_\varepsilon \times T_L^-). \end{aligned} \tag{4.3}$$

Now, we can define $T_\varepsilon = T_L^- \cup T_r$, $T_L^- \cap T_r = \emptyset$ and similarly for N_ε and N_L^-. Then, expand the partitions and products, and distribute the intersections around the unions, noticing all but one of intersections will be empty:

$$\begin{aligned} T_\varepsilon \times N_L^- \cap N_\varepsilon \times T_L^- &= ((T_r \cup T_L^-) \times N_L^-) \cap ((N_r \cup N_L^-) \times T_L^-) \\ &= ((T_r \times N_L^-) \cup (T_L^- \times N_L^-)) \\ &\quad \cap ((N_r \times T_L^-) \cup (N_L^- \times T_L^-)) \\ &= N_L^- \times T_L^-. \end{aligned}$$

Therefore,

$$\chi(T_\varepsilon \times N_L^- \cap N_\varepsilon \times T_L^-) = \chi(N_L^- \times T_L^-) = \chi(N_L^-)\chi(T_L^-),$$

which gives the final result

$$\chi(L^-) = \chi(N_L^-) + \chi(T_L^-) - \chi(N_L^-)\chi(T_L^-). \tag{4.4}$$

The same result is valid for $\chi(L^+)$, if we replace the superscript "$-$" by "$+$" in (4.4). If T_L^- or T_L^+ are 1D, then we are done. If not, then we can recursively apply the same equation to T_L^- and T_L^+ and look at progressively lower-dimensional strata until we reach $T_\varepsilon(p)$ and $N_\varepsilon(p)$ given by 1-disks. The lower and upper links for these 1-disks will always be discrete spaces with zero, one, or two points, for which χ is simply the cardinality of the set.

In some cases, the Euler characteristic of the lower and upper link might be equal. Then, $\chi(L^-(p)) = \chi(L^+(p))$, and $\Delta\chi(p) = 0$. These cases correspond to the virtual critical points mentioned above. Critical points in the interior of cubic cells are handled by the smooth theory, since in that case the normal Morse data are 0 dimensional. This implies that the link will be an empty set with Euler characteristic zero. So, by (4.4), $\chi(L^-) = \chi(T_L^-)$. Because the restriction of the scalar field to a grid edge is a linear function, no critical point can appear there. As a result, the new cases are critical points occurring at vertices or in the interior of faces of the grid. For a critical point p in a vertex, stratification can be carried out recursively, using the edges of the cubes meeting in p as tangential and normal submanifolds. Denoting by n_{l1}, n_{l2}, n_{l3} the number of vertices adjacent to p with scalar value less than that of p in each Cartesian coordinate direction, (4.4) gives

$$\chi(L^-(p)) = n_{l1} + n_{l2} + n_{l3} - n_{l1}(n_{l2} + n_{l3}), \tag{4.5}$$

$\chi(L^+(p))$ can be computed similarly, but considering the number of neighbors of p in each Cartesian direction with scalars higher than that of p.

If p is a critical point lying in a face r of a cube, we consider the face itself as the tangential submanifold and the line segment $r^\perp$ orthogonal to r through p the normal submanifold. Recursively, the tangential submanifold can be further stratified in two 1-disks (tangential and normal). Denote by n_l the number of ends of $r^\perp$ with scalar value less than that of p. Also, recalling that the critical point lying in the face r is necessarily a saddle, thus having two face corners with scalar values less and two higher than that of p, (4.4) gives

$$\chi(L^-(p)) = n_l + 2 - 2\, n_l. \tag{4.6}$$

Analogously, we can compute $\chi(L^+(p)) = n_u + 2 - 2\, n_u$ where n_u is the number of ends of $r^\perp$ with scalar value higher than that of p.

A similar analysis can be carried out for every type of critical point, regardless of whether the point belongs to the interior of a grid cell (and so would yield equally well to a smooth Morse theory analysis), an interior face, a boundary face, or a vertex of any type. The Euler characteristic χ_α of any isosurface with isovalue α is simply given as

$$\chi_\alpha = \sum_{p_i \in C_\alpha} \Delta\chi(p_i), \tag{4.7}$$

where C_α is the set of critical points with critical values less than α.

It is worth mentioning once again that, to the best of our knowledge, no other work has presented a scheme which provides such a simple mechanism for computing the Euler characteristic of level sets of piecewise-smooth trilinear functions. Compare, for example, the case analyses and state machines performed separately by Nielson [30], by Carr and Snoeyink [5], and by Carr and Max [3]. In contrast, we can recover an (admittedly weaker) topological invariant by a much simpler argument. In addition, this argument already generalizes (trivially because of the stratification argument) to arbitrary dimensions, unlike the other arguments in the literature.

```
MMS-SMT(𝒢)
 1  ▷ Let the input 𝒢 be n × n × n rectilinear grid
 2  for i ← 1 to #tests
 3       do 𝒢 ← randomly sampled n × n × n grid
 4          CPs ← ComputeCriticalPoints(𝒢)
 5          if p ∈ CPs is degenerate or
 6          p is an internal saddle close to edges or faces
 7            then GoTo 3
 8            else K ← Isosurfacing(𝒢)
 9                 (χ^v)_i ← InvariantFromCPs(𝒢)
10                 (χ^k)_i ← InvariantFromMesh(K)
11          Compare (χ^v)_i and (χ^k)_i
```

Fig. 7. Overview of the method of manufactured solutions using stratified Morse theory. InvariantFromCPs is computed using (4.7). The method either fails to match the expected topology, in which case $\mathcal{G}$ is provided as a counterexample, or succeeds otherwise.

5 Manufactured Solution Pipeline

We now put the pieces together and build a pipeline for topology verification using the results presented in Section 4. In the following sections, the procedure called Isosurfacing refers to the isosurface extraction technique under verification. InvariantFromMesh computes topological invariants of a simplicial complex.

5.1 Consistency

As previously mentioned, MC-like algorithms which use disambiguation techniques are expected to generate PL manifold isosurfaces no matter how complex the function sampled in the vertices of the regular grid. In order to stress the consistency test, we generate a random scalar field with values in the interval $[-1, 1]$ and extract the isosurface with isovalue $\alpha = 0$ (which is all but guaranteed not to be a critical value) using a given isosurfacing technique, subjecting the resulting triangle mesh to the consistency verification. This process is repeated a large number of times, and if the implementation fails to produce PL manifolds for all cases, then the counterexample provides a documented starting point for debugging. If it passes the tests, we consider the implementation verified.

5.2 Verification Using Stratified Morse Theory

We can use the formulation described in Section 4.2 to verify isosurfacing programs which promise to match the topology of the trilinear interpolant. The SMT-based verification procedure is summarized in Fig. 7. The algorithm has four main steps. A random scalar field with node values in the interval $[-1, 1]$ is initially created. Representing the trilinear interpolation in a grid cell by $f(x, y, z) = axyz + bxy + cxz + dyz + ex + fy + gz + h$, the internal critical points are given by

$$t_x = (d\Delta_x \pm \sqrt{\Delta_x\Delta_y\Delta_z})/(a\Delta_x),$$
$$t_y = (c\Delta_y \pm \sqrt{\Delta_x\Delta_y\Delta_z})/(a\Delta_y),$$
$$t_z = (b\Delta_z \pm \sqrt{\Delta_x\Delta_y\Delta_z})/(a\Delta_z),$$

where $\Delta_x = bc - ae$, $\Delta_y = bd - af$, and $\Delta_z = cd - ag$ [32]. Critical points on faces of the cubes are found by setting x, y or z to either 0 or 1, and solving the quadratic equation. If the solutions lie outside the unit cube $[0, 1]^3$, they are not considered critical points, since they lie

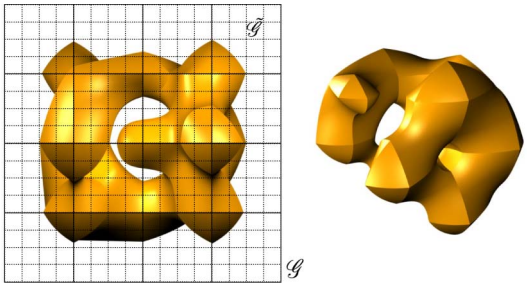


Fig. 8. Our manufactured solution is given by $t(x) = \alpha$. $\mathcal{G}$ is depicted in solid lines while $\tilde{\mathcal{G}}$ is shown in dashed lines. $\tilde{\mathcal{G}}$ is a uniform subdivision of $\mathcal{G}$. The trilinear surfaces t_i are defined for each cube $c_i \in \mathcal{G}$ and resampled in $c'_j \in \tilde{\mathcal{G}}$. The cubes in the center of $\mathcal{G}$ have four maxima each (left) and thus induce complicated topology. The final isosurface may have several tunnels and/or connected components even for coarse $\mathcal{G}$ (right).

outside the domain of the cell. The scalar field is regenerated if any degenerate critical point is detected (these can happen if either the random values in a cubic cell have, by chance, the same value or when Δ_x, Δ_y or Δ_z are zero). In order to avoid numerical instabilities, we also regenerate the scalar field locally if any internal critical point lies too close to the border of the domain (that is, to an edge or to a face of the cube).

The third step computes the Euler characteristic of a set of isosurfaces with random isovalues in the interval $[-1, 1]$ using the theory previously described, jointly with (4.7). In the final step, the triangle mesh M approximating the isosurfaces is extracted using the algorithm under verification, and $\chi(M) = V(M) - E(M) + F(M)$, where $V(M), E(M)$, and $F(M)$ are the number of vertices, edges, and triangles. If the Euler characteristic computed from the mesh does not match the one calculated via (4.7), the verification fails. We carry out the process a number of times, and implementations that pass the tests are less likely to contain bugs.

5.3 Verification Using Digital Topology

Fig. 9 shows the verification pipeline using the MI algorithm, and Fig. 8 depicts the refinement process. Once again a random scalar field, with potentially many ambiguous cubes, is initially generated in the vertices of a grid $\mathcal{G}$. The algorithm illustrated in Fig. 9 is applied to refine $\mathcal{G}$ so as to generate a new grid $\tilde{\mathcal{G}}$ which does not have ambiguous cells. If the maximum number of refinement is reached and ambiguous cells still remain, then the process is restarted from scratch. Notice that cube subdivision does not need to be uniform. For instance, each cube may be refined using a randomly placed new node point or using t_i's critical points, and the result of the verification process still holds. This is because Theorem 4.1 only requires c_i to be unambiguous. For simplicity, in this paper we refine $\mathcal{G}$ uniformly doubling the grid resolution in each dimension.

Scalars are assigned to the new vertices of $\tilde{\mathcal{G}}$ (the ones not in $\mathcal{G}$) by trilinearly interpolating from scalars in $\mathcal{G}$, thus ensuring that $\mathcal{G}$ and $\tilde{\mathcal{G}}$ have exactly the same scalar field [30]. As all cubic cells in $\tilde{\mathcal{G}}$ are unambiguous, Theorem 4.1 guarantees the topology of the digital surface $\partial\mathcal{O}_\alpha$ obtained from $\tilde{\mathcal{G}}$ is equivalent to that of $t(x) = \alpha$. Algorithm INVARIANTFROMDS computes topological invariants of $\partial\mathcal{O}_\alpha$ using the scheme discussed in Section 4.1. In this

MMS-DS($\mathcal{G}$)

```
1  ▷ Let the input 𝒢 be a n × n × n rectilinear grid
2  for i ← 1 to #tests
3       do 𝒢 ← randomly sampled n × n × n grid
4          𝒢̃ ← RefineAndResample(𝒢)
5          if 𝒢̃ has ambiguous cubes
6            then GoTo 3
7          𝒪 ← MajorityInterpolation(𝒢̃)
8          K ← Isosurfacing(𝒢)
9          (β0^v, β1^v, β2^v)_i ← InvariantFromDS(∂𝒪)
10         (β0^k, β1^k, β2^k)_i ← InvariantFromMesh(K)
11         Compare (β0^v, β1^v, β2^v)_i and (β0^k, β1^k, β2^k)_i
```

Fig. 9. Overview of the method of manufactured solutions using digital topology. The method either fails to match the expected topology, in which case $\mathcal{G}$ is provided as a counterexample, or succeeds otherwise.

context, INVARIANTFROMDS is the algorithm illustrated in Fig. 5. Surfaces with boundary are avoided by assigning the scalar value 1 to every vertex in the boundary of $\mathcal{G}$.

6 Experimental Results

In this section, we present the results of applying our topology verification methodology to a number of different isosurfacing techniques, three of them with topological guarantees with respect to trilinear interpolant. Specifically, the techniques are:

VTKMC [38] is the Visualization Toolkit (VTK) implementation of the Marching Cubes algorithm with the implicit disambiguation scheme proposed by Montani et al. [26]. Essentially, it separates positive vertices when a face saddle appears and assumes no tunnels exist inside a cube. The proposed scheme is topologically consistent, but it does not reproduce the topology of the trilinear interpolant.

Marching Cubes with Edge Transformations or MACET [12] is a Marching Cubes-based technique designed to generate triangle meshes with good quality. Quality is reached by displacing active edges of the grid (edges intersected by the isosurface), both in normal and tangential direction toward avoiding "sliver" intersections. Macet does not reproduce the topology of the trilinear interpolant.

AFRONT [37] is an advancing-front method for isosurface extraction, remeshing, and triangulation of point sets. It works by advancing triangles over an implicit surface. A sizing function that takes curvature into account is used to adapt the triangle mesh to features of the surface. AFRONT uses cubic spline reconstruction kernels to construct the scalar field from a regular grid. The algorithm produces high-quality triangle meshes with bounded Hausdorff error. As occurred with the VTK and Macet implementations, Afront produces consistent surfaces but, as expected, the results do not match the trilinear interpolant.

MATLAB [24] is a high-level language for building codes that requires intensive numerical computation. It has a number of features and among them an isosurface extraction routine for volume data visualization. Unfortunately, MATLAB documentation does not offer information on the particularities of the implemented isosurface extraction technique (e.g., Marching Cubes, Delaunay-based, etc.; consistent or correct).

SNAPMC [34] is a Marching Cubes variant which produces high-quality triangle meshes from regular grids. The central idea is to extend the original lookup table to account for cases where the isosurface passes exactly through the grid nodes. Specifically, a user-controlled parameter dictates maximum distance for "snapping" the isosurface into the grid node. The authors report an improvement in the minimum triangle angle when compared to previous techniques.

MC33 was introduced by Chernyaev [8] to solve ambiguities in the original MC. It extends Marching Cubes table from 15 to 33 cases to account for ambiguous cases and to reproduce the topology of the trilinear interpolant inside each cube. The original table was later modified to remove two redundant cases which leads to 31 unique configurations. Chernyaev's MC solves face ambiguity using Nielsen and Hamann's [31] asymptotic decider and internal ambiguity by evaluating the bilinear function over a plane parallel to a face. Additional points may be inserted to reproduce some configuration requiring subvoxel accuracy. We use Lewiner et al.'s implementation [21] of Chernyaev's algorithm.

DELISO [11] is a Delaunay-based approach for isosurface extraction. It uses the intersection of the 3D Voronoi diagram and the desired surface to define a restricted Delaunay triangulation. Moreover, it builds the restricted Delaunay triangulation without having to compute the whole 3D Voronoi structure. DELISO has theoretical guarantees of homeomorphism and mesh quality.

MCFLOW is a proof-of-concept implementation of the algorithm described in Scheidegger et al. [36]. It works by successive cube subdivision until it has a *simple edge flow*. A cube has a simple edge flow if it has only one *minima* and one *maxima*. A vertex $s \in c_i$ is a minimum if all vertices $s_j \in c_i$ connected to it has $t(s_j) > t(s_i)$. Similarly, a vertex is a maximum if $t(s_j) < t(s_i)$ for every neighbor vertex j. This property guarantees that the Marching Cubes method will generate a triangle mesh homeomorphic to the isosurface. After subdivision, the surfaces must be attached back together. The final mesh is topologically correct with respect to the trilinear interpolant.

We believe that the implementation of any of these algorithms in full detail is nontrivial. The results reported in the following section support this statement. They show that coding isosurfacing algorithms is complex and error-prone, and they reinforce the need for robust verification mechanisms. In what follows, we say that a *mismatch* occurs when invariants computed from a verification procedure disagree with the invariants computed from the isosurfacing technique. A mismatch does not necessarily mean an implementation is incorrect, as we shall see later in this section. After discussions with the developers, however, we did find that there were bugs in some of the implementations.

6.1 Topology Consistency

All implementations were subject to the consistency test (Section 5.1), resulting in the outputs reported in the first column of Table 1. We observed mismatches for DELISO, SNAPMC (with nonzero snap value), and MATLAB implementations. Now, we detail these results.

6.1.1 DELISO

We analyzed 50 cases where DELISO's output mismatched the ground truth produced by MMS, and we found that: 1) 28 cases had incorrect hole(s) in the mesh, 2) 15 cases had missing triangle(s), and 3) seven cases had duplicated vertices. These cases are illustrated in Fig. 11. The first problem is possibly due to the nonsmooth nature of the piecewise trilinear interpolant, since in all 28 cases the holes appeared in the faces of the cubic grid. It is important to recall that DELISO is designed to reproduce the topology of the trilinear interpolant inside each grid cube, but the underlying algorithm requires the isosurface to be C^2 continuous everywhere, which does not hold for the piecewise trilinear isosurface. In practice, real-world data sets such as medical images may induce "smoother" piecewise trilinear fields when compared to the extreme stressing from the random field, which should reduce the incidence of such cases. Missing triangles, however, occurred in the interior of cubic cells where the trilinear surface is smooth. Those problems deserve a deeper analysis, as one cannot say beforehand if the mismatches are caused by problems in the code or numerical instability associated with the initial sampling, ray-surface intersection, and the 3D Delaunay triangulation construction.

TABLE 1
Rate of Invariant Mismatches Using the PL Manifold Property, Digital Surfaces, and Stratified Morse Theory for 1,000 Randomly Generated Scalar Fields (the Lower the Rate the Better)

	Consistency (%)	Correctness (%)				
	Disk	Digital Surfaces				SMT
		β_0	β_1	β_2	χ	χ
AFRONT	0.0	35.9	22.8	35.9	47.5	25.5
MATLAB	19.7	32.2	18.9	20.5	49.3	70.3
VTKMC	0.0	27.6	23.2	27.6	43.5	70.7
MACET	0.0	54.3	20.9	54.3	64.0	100.0
SNAPMC[1]	0.0	45.0	25.4	45.0	57.3	72.0
SNAPMC[2]	53.7	41.6	17.3	23.1	87.1	74.0
MC33	0.0	2.4	1.1	2.4	3.4	5.4
DELISO	19.1	24.4	0.1	20.0	37.2	33.2
MCFLOW	0.0	0.0	0.0	0.0	0.0	0.0

The Invariants β_1 and β_2 are Computed Only if the Output Mesh is a 2-Manifold without Boundary. We run correctness tests in all algorithms for completeness and to test tightness of the theory: algorithms that are not topology-preserving should fail these tests. *The high number of DELISO, SNAPMC, and MATLAB mismatches are explained in Section 6.1.* [1] *indicates zero snap parameter and* [2] *indicates snap value of 0.3.*

6.1.2 SNAPMC

Table 1 shows that SNAPMC with nonzero snap value causes the mesh to be topologically inconsistent (Fig. 13a) in more the 50 percent of the performed tests. The reason for this behavior is in the heart of the technique: the snapping process causes geometrically close vertices to be merged together which may eliminate connected components, or loops, join connected components or even create nonmanifold surfaces. This is why there was an increase in the number of mismatches when compared with SNAPMC with zero snap value. Since nonmanifold meshes are not desirable in many applications, the authors suggest a postprocessing for fixing these topological issues, although no implementation or algorithm for this postprocessing is provided.

6.1.3 MATLAB

MATLAB documentation does not specify the properties of the implemented isosurface extraction technique. Consequently, it becomes hard to justify the results for the high

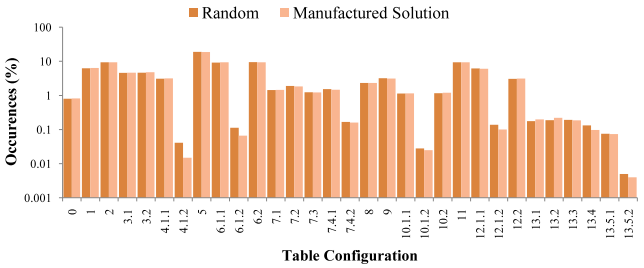


Fig. 10. The horizontal axis shows the case and subcase numbers for each of the 31 Marching Cubes configurations described by Lopes and Brodlie [22]. The dark bars show the percentage of random fields that fit a particular configuration. The light bars show the percentage of random fields which fit a particular configuration *and* do not violate the assumptions of our manufactured solution. Our manufactured solution hits all possible cube configurations.

number of mismatches we see in Table 1. For instance, Fig. 13b shows an example of a nonmanifold mesh extracted using MATLAB. In that figure, the two highlighted edges have more than two faces connected to them and the faces between these edges are coplanar. Since we do not have enough information to explain this behavior, this might be the actual expected behavior or an unexpected side effect. An advantage of our tests is the record of the observed behavior of mesh topologies generated by MATLAB.

6.1.4 MACET

In our first tests, MACET failed in all consistency tests for a $5 \times 5 \times 5$ grid. An inspection in the code revealed that the layer of cells in the boundary of the grid has not been traversed. Once that bug was fixed, MACET started to produce PL manifold meshes and was successful in the consistency test, as shown in Table 1.

6.2 Topology Correctness

The verification tests described in Sections 5.2 and 5.3 were applied to all algorithms, although only MC33, DELISO, and MCFLOW were expected to generate meshes with the same topology of the trilinear interpolant. Our tests consisted of one thousand random fields generated in a rectilinear $5 \times 5 \times 5$ grid $\mathcal{G}$. The verification test using Digital Surfaces demanded a compact, orientable, 2-manifold without boundary, so we set scalars equal to 1 for grid vertices in the boundary of the grid. As stratified Morse theory supports surfaces with boundary, no special treatment was employed in the boundary of $\mathcal{G}$. We decided to run these tests using all algorithms for completeness and also for testing the tightness of the theory, which says that if the algorithms do not preserve the topology of the trilinear interpolant, a mismatch should occur. Interestingly, with this test, we were able to find another code mistake in MACET that prevented it from terminating safely when the SMT procedure was applied. By the time of the submission of this paper, the problem was not fixed. For all nontopology-preserving algorithms, there was a high number of mismatches as expected.

One might think that the algorithms described in Figs. 7 and 9 do not cover all possible topology configurations because some scalar fields are eventually discarded (lines 7 and 6, respectively). This could happen due to the presence of ambiguous cells after refining the input grid to the maximum tolerance (digital topology test) or critical points falling too close to edges/faces of the cubic cells (SMT test). However, we can ensure that all possible configurations for the trilinear interpolation were considered in the tests. Fig. 10 shows the incidence of each possible configuration (including all ambiguous cases) for the trilinear interpolation in the generated random fields. Dark bars correspond to the number of times a specific case happens in the random field, and the light bars show how many of those cases are accepted by our verification methodology, that is, the random field is not discarded. Notice that no significant differences can be observed, implying that our rejection-sampling method does not bias the case frequencies.

Some configurations, such as 13 or 0, have low incidence rates and therefore might not be sufficiently stressed during verification. While the trivial case 0 does not pose a challenge for topology-preserving implementations, configuration 13 has six subcases whose level-sets are fairly complicated [22], [30]. Fortunately, we can build random fields in a convenient fashion by forcing a few cubes to represent a particular instance of the table, such as case 13, which produces more focused tests.

Table 1 shows statistics for all implementations. For MC33, the tests revealed a problem with configuration 4, 6, and 13 of the table (ambiguous cases). Fig. 12 shows the

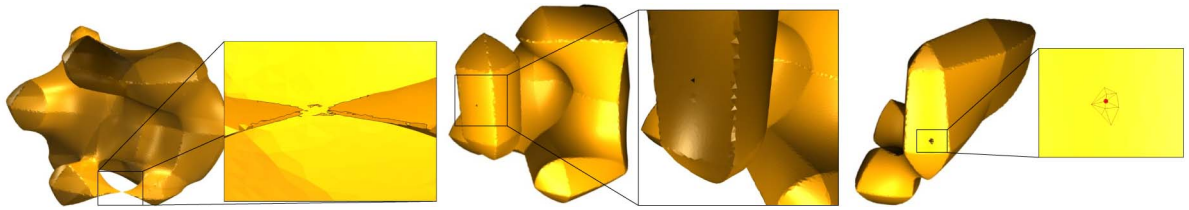


Fig. 11. DELISO mismatch example. From left to right: holes in C^0 regions; single missing triangle in a smooth region; duplicated vertex (the mesh around the duplicated vertex is shown). These behavior induce topology mismatches between the generated mesh and the expected topology.

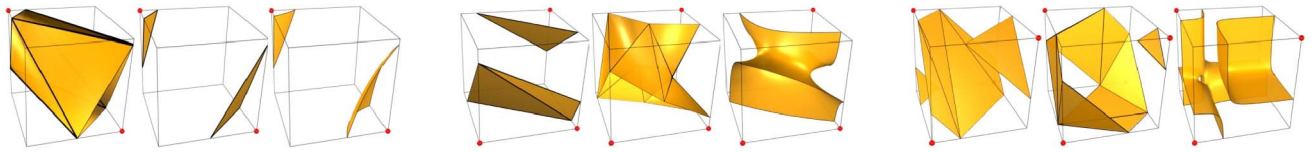
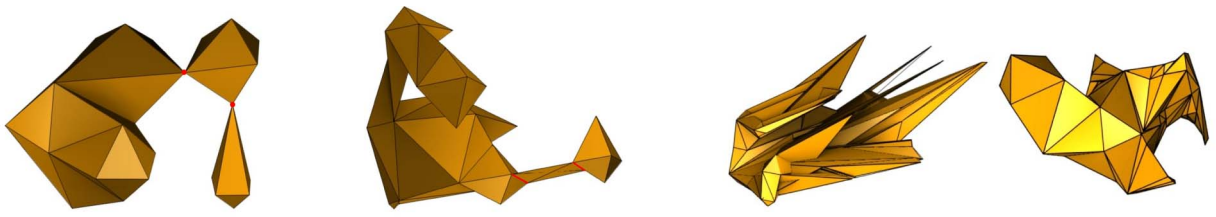


Fig. 12. MC33 mismatch example. From left to right: problem in the cases 4.1.2, 6.1.2, and 13.5.2 of marching cube table (all are ambiguous). Each group of three pictures shows the obtained, expected, and implicit surfaces. Our verification procedure can detect the topological differences between the obtained and expected topologies, even for ambiguous cases.



(a) SNAPMC (snap = 0.3) (b) MATLAB (c) MCFLOW

Fig. 13. Mismatches in topology and geometry. (a) SNAPMC generates nonmanifold surfaces due to the snap process. (b) MATLAB generates some edges (red) that are shared by more than two face. (c) MCFLOW before (left) and after (right) fixing a bug that causes the code to produce the expected topology, but the wrong geometry.

obtained and expected tiles for a cube. Contacting the author, we found that one of the mismatches was due to a mistake when coding configuration 13 of the MC table. A nonobvious algorithm detail that is not discussed in either Chernyaev's or Lewiner's work is the problem of orientation in some of the cube configurations [20]. The case 13.5.2 shown in Fig. 12 (right) is an example of one such configuration, where an additional criterion is required to decide the tunnel orientation that is lacking in the original implementation of MC33. This problem was easily detected by our framework, because the orientation changes the mesh invariants, and a mismatch occurs.

DELISO presented a high percentage of β_0 mismatches due to the mechanism used for tracking connected components. It uses ray-surface intersection to sample a few points over each connected component of the isosurface before extracting it. The number of rays is a user-controlled parameter and its initial position and direction are randomly assigned. DELISO is likely to extract the biggest connected component and, occasionally, it misses small components. It is important to say that the ray-sample-based scheme tends to work fine in practical applications where small surfaces are not present. The invariant mismatches for β_1 and β_2 are computed only if no consistency mismatch happens.

For MCFLOW, we applied the verification framework systematically during its implementation/development. Obviously, many bugs were uncovered and fixed over the course of its development. Since we are randomizing the piecewise trilinear field, we are likely to cover all possible Marching Cubes entries and also different cube combinations. As verification tests have been applied since the very beginning, all detectable bugs were removed, resulting in no mismatches. The downside of MCFLOW, though, is that typical bad quality triangles appearing in Marching Cubes become even worse in MCFLOW, because cubes of different sizes are glued together. MCFLOW geometrical convergence is presented in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.109 [36].

7 Discussion and Limitations

7.1 Quality of Manufactured Solutions

In any use of MMS, one very important question is that of the quality of the manufactured solutions, since it reflects directly on the quality of the verification process. Using random solutions, for which we compute the necessary invariants, naturally seems to yield good results. However, our random solutions will almost always have nonidentical values. This raises the issue of detecting and handling degenerate inputs, such as the ones arising from quantization. We note that most implementations use techniques such as Simulation of Simplicity [14] (for example, by arbitrarily breaking ties using node ordering) to effectively keep the facade of nondegeneracy. However, we note that developing manufactured solutions specifically to stress degeneracies is desirable when using verification tools during development. We decided against this since different implementations might employ different strategies to handle degeneracies and our goal was to keep the presentation sufficiently uniform.

7.2 Topology and Geometry

This paper extends the work by Etiene et al. [15] toward including topology in the loop of verification for isosurface techniques. The machinery presented herein combined with the methodology for verifying geometry comprises a solid battery of tests able to stress most of the existing isosurface extraction codes.

To illustrate this, we also submitted MC33 and MCFLOW techniques to the geometrical test proposed by Etiene, as these codes have not been geometrically verified. While MC33 has geometrical behavior in agreement with Etiene's approach, the results presented in Section 6 show it does not pass the topological tests. On the other hand, after ensuring that MCFLOW was successful regarding topological tests, we submitted it to the geometrical analysis, which revealed problems. Fig. 13c shows an example of an output generated in the early stages of development of MCFLOW before (left) and after (right) fixing the bug. The topology matches the expected one (a topological sphere); nevertheless, the geometry does not converge.

7.3 SMT versus DT

The verification approach using digital surfaces generates detailed information about the expected topology because it provides β_0, β_1, and β_2. However, verifying isosurfaces with boundaries would require additional theoretical results, as the theory supporting our verification algorithm is only valid for surfaces without boundary. In contrast, the verification methodology using stratified Morse theory can handle surfaces with boundary. However, SMT only provides information about the Euler characteristic, making it harder to determine when the topological verification process fails. Another issue with SMT is that if a code incorrectly introduces topological features so as to preserve χ, then no failure will be detected. For example, suppose the surface to be reconstructed is a torus, but the code produces a torus plus three triangles, each one sharing two vertices with the other triangles but not an edge. In this case, torus plus three

"cycling" triangles also has $\chi = 0$, exactly the Euler characteristic of the single torus. In that case, notice that the digital surface-based test would be able to detect the spurious three triangles by comparing β_0. Despite being less sensitive in theory, SMT-based verification revealed similar problems as the digital topology tests have. We believe this effectiveness comes in part from the randomized nature of our tests.

7.4 Implementation of SMT and DT

Verification tools should be as simple as possible while still revealing unexpected behavior. The pipeline for geometric convergence is straightforward and thus much less error-prone. This is mostly because Etiene et al.'s approach uses analytical manufactured solutions to provide information about function value, gradients, area, and curvature. In topology, on the other hand, we can manufacture only simple analytical solutions (e.g., a sphere, torus, double-torus, etc.) for which we know topological invariants. There are no guarantees that these solutions will cover all cases of a trilinear interpolant inside a cube. For this reason, we employ a random manufactured solution and must then compute explicitly the topological invariants. A point which naturally arises in verification settings is that the verification code is another program. How do we verify the verifier?

First, note that the implementation of either verifier is simpler than the isosurfacing techniques under scrutiny. This reduces the chances of a bug impacting the original verification. In addition, we can use the same strategy to check if the verification tools are implemented correctly. For SMT, one may compute χ for an isovalue that is greater than any other in the grid. In such case, the verification tool should result in $\chi = 0$. For DT, we can use the fact that Majority Interpolation always produces a 2-manifold. Fortunately, this test reduces to check for two invalid cube configurations as described by Stelldinger et al. [39]. Obviously, there might remain bugs in the verification code. As we have stated before, a mismatch between the expected invariants and the computed ones indicates a problem *somewhere* in the pipeline; our experiments are empirical evidence of the technique's effectiveness in detecting implementation problems.

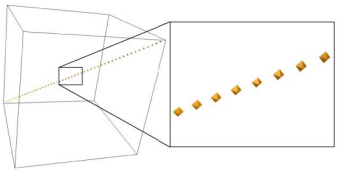
Another concern is the performance of the verification tools. In our experiments, the invariant computation via SMT and DS is faster than any isosurface extraction presented in this paper, for most of the random grids. In some scenarios, DS might experience a slowdown because it refines the grid in order to eliminate ambiguous cubes (the maximum number of refinement is set to 4). Thus, both SMT and DS (after grid refinement) need to perform a constant number of operations for each grid cube to determine the DS or critical points (SMT). In this particular context, we highlight the recent developments on certifying algorithms, which produce both the output and an *efficiently checkable certificate of correctness* [25].

7.5 Contour Trees

Contour trees [6] are powerful structures to describe the evolution of level-sets of simply connected domains. It normally assumes a simplicial complex as input, but there are extensions to handle regular grids [32]. Contour trees naturally provide β_0, and they can be extended to report β_1 and β_2. Hence, for any isovalue, we have information about all Betti numbers, even for surfaces with boundaries. This fact renders contour trees a good candidate for verification purposes. In fact, if an implementation is available, we encourage its use so as to increase confidence in the algorithm's behavior. However, the implementation of a contour tree is more complicated than the techniques presented here. For regular grids, a divide-and-conquer approach can be used along with oracles representing the split and join trees in the deepest level of the recursion, which is nontrivial. Also, implementing the merging of the two trees to obtain the final contour tree is still involving and error-prone. Our approach, on the other hand, is based on regular grid refinement and voxel selection for the DT method and critical point computation and classification for the SMT method. There are other tools, including contour trees, that could be used to assess topology correctness of isosurface extraction algorithms, and an interesting experiment would be to compare the number of mismatches found by each of these tools. Nevertheless, in this paper, we have focused on the approaches using SMT and DT because of their simplicity and effectiveness in finding code mistakes in publicly available implementations. We believe that the simpler methodologies we have presented here are more likely to be adopted during development of visualization isosurfacing tools.

7.6 Topology of the Underlying Object

In this paper, we are interested in how to effectively verify topological properties of codes which employ trilinear interpolation. In particular, this means that our verification tools will work for implementations other than marching methods (for example, DelIso is based on Delaunay refinement). Nevertheless, in practice, the original scalar field will not be trilinear, and algorithms which assume a trilinearly interpolated scalar field might not provide any topological guarantee regarding the reconstructed object. Consider, for example, a piecewise linear curve γ built by walking through diagonals of adjacent cubes $c_i \in \mathcal{G}$ and define the distance field $d(x) = \min\{\|x - x'\| \text{ such that } x' \in \gamma\}$. The isosurface $d(x) = \alpha$ for any $\alpha > 0$ is a single tube around γ. However, none of the implementations tested could successfully reproduce the tubular structure for all $\alpha > 0$. This is not particularly surprising, since the trilinear interpolation from samples of d is quite different from the d. The inline figure shows a typical output produced by VTK Marching Cubes for the distance field $d = \alpha$. Notice, however, that this is not only an issue of sampling rate because if the tube keeps going through the diagonals of cubic cells, VTK will not be able reproduce $d = \alpha$ yet. Also recall that some structures cannot even be reproduced by trilinear interpolants, as when γ crosses diagonals of two parallel faces of a cubic cell, as described in [8], [32]. The aspects above are not errors in the codes but reflect software design choices that should be clearly expressed to users of those visualization techniques.



7.7 Limitations

The theoretical guarantees supporting our manufactured solution rely on the trilinear interpolant. If an interpolant other than trilinear is employed, then new results ensuring

homeomorphism (Theorem 4.1) should be derived. The basic infrastructure we have described here, however, should be appropriate as a starting point for the process.

8 Conclusion and Future Work

We extended the framework presented by Etiene et al. [15] by including topology into the verification cycle. We used machinery from digital topology and stratified Morse theory to derive two verification tools that are simple and yet capable of finding unexpected behavior and coding mistakes. We argue that researchers and developers should consider adopting verification as an integral part of the investigation and development of scientific visualization techniques. Topological properties are as important as geometric ones, and they deserve the same amount of attention. It is telling that the only algorithm that passed all verification tests proposed here is the one that used the verification procedures *during* its development. We believe this happened because topological properties are particularly subtle and require an unusually large amount of care.

The idea of verification through manufactured solutions is clearly problem-dependent and mathematical tools must be tailored accordingly. Still, we expect the framework to enjoy a similar effectiveness in many areas of scientific visualization, including volume rendering, streamline computation, and mesh simplification. We hope that the results of this paper further motivate the visualization community to develop a culture of verification.

Acknowledgments

The authors thank Thomas Lewiner and Joshua Levine for help with MC33 and DELISO codes, respectively. 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, CMMI 1053077, IIP 0810023, CCF 0429477), 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 Eng.*, vol. 193, nos. 36-38, pp. 4057-4066, 2004.

[2] J. Bloomenthal, "Polygonization of Implicit Surfaces," *Computer Aided Geometric Design*, vol. 5, no. 4, pp. 341-355, 1988.

[3] H. Carr and N. Max, "Subdivision Analysis of the Trilinear Interpolant," *IEEE Trans. Visualization and Computer Graphics*, vol. 16, no. 4, pp. 533-547, July/Aug. 2010.

[4] H. Carr, T. Möller, and J. Snoeyink, "Artifacts Caused by Simplicial Subdivision," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 2, pp. 231-242, Mar./Apr. 2006.

[5] H. Carr and J. Snoeyink, "Representing Interpolant Topology for Contour Tree Computation," *Proc. Topology-Based Methods in Visualization II*, pp. 59-73, 2009.

[6] H. Carr, J. Snoeyink, and U. Axen, "Computing Contour Trees in all Dimensions," *Computational Geometry: Theory and Applications*, vol. 24, no. 2, pp. 75-94, 2003.

[7] L. Chen and Y. Rong, "Digital Topological Method for Computing Genus and the Betti Numbers," *Topology and its Applications*, vol. 157, no. 12, pp. 1931-1936 2010, http://dx.doi.org/10.1016/j.topol.2010.04.006.

[8] E.V. Chernyaev, "Marching Cubes 33: Construction of Topologically Correct Isosurfaces," Technical Report CN/95-17, 1995.

[9] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno, "Reconstruction of Topologically Correct and Adaptive Trilinear Isosurfaces," *Computers and Graphics*, vol. 24, pp. 399-418, 2000.

[10] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of Persistence Diagrams," *J. Discrete and Computational Geometry*, vol. 37, no. 1, pp. 103-120, 2007.

[11] T.K. Dey and J.A. Levine, "Delaunay Meshing of Isosurfaces," *Proc. IEEE Int'l Conf. Shape Modeling and Applications (SMI '07)*, pp. 241-250, 2007.

[12] C. Dietrich, C. Scheidegger, J. Schreiner, J. Comba, L. Nedel, and C. Silva, "Edge Transformations for Improving Mesh Quality of Marching Cubes," *IEEE Trans. Visualization and Computer Graphics*, vol. 15, no. 1, pp. 150-159, Jan./Feb. 2008.

[13] H. Edelsbrunner and J.L. Harer, *Computational Topology*. American Mathematical Soc., 2010.

[14] H. Edelsbrunner and E.P. Mücke, "Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms," *ACM Trans. Graphics*, vol. 9, pp. 66-104, 1990.

[15] T. Etiene, C. Scheidegger, L.G. Nonato, R.M. Kirby, and C. Silva, "Verifiable Visualization for Isosurface Extraction," *IEEE Trans. Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1227-1234, Jan./Feb. 2009.

[16] A. Globus and S. Uselton, "Evaluation of Visualization Software," *ACM SIGGRAPH Computer Graphics*, vol. 29, no. 2, pp. 41-44, 1995.

[17] M. Goresky and R. MacPherson, *Stratified Morse Theory*. Springer, 1988.

[18] C.-C. Ho, F.-C. Wu, B.-Y. Chen, Y.-Y. Chuangs, and M. Ouhyoungs, "Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data," *Computer Graphics Forum*, vol. 24, no. 3, pp. 537-545, 2005.

[19] R. Kirby and C. Silva, "The Need for Verifiable Visualization," *IEEE Computer Graphics and Applications*, vol. 28, no. 5, pp. 78-83, Sept./Oct. 2008.

[20] T. Lewiner, "Personal Communication," Mar. 2010.

[21] T. Lewiner, H. Lopes, A.W. Vieira, and G. Tavares, "Efficient Implementation of Marching Cubes' Cases with Topological Guarantees," *J. Graphics Tools*, vol. 8, no. 2, pp. 1-15, 2003.

[22] A. Lopes and K. Brodlie, "Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing," *IEEE Trans. Visualization and Computer Graphics*, vol. 9, no. 1, pp. 16-29, Jan.-Mar. 2003.

[23] W. Lorensen and H. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, pp. 163-169, 1987.

[24] MathWorks "Matlab," http://www.mathworks.com/products/matlab/, Mar. 2011.

[25] R. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer, "Certifying Algorithms," *Computer Science Rev.*, 2010.

[26] C. Montani, R. Scateni, and R. Scopigno, "A Modified Look-Up Table for Implicit Disambiguation of Marching Cubes," *The Visual Computer*, vol. 10, no. 6, pp. 353-355, Dec. 1994.

[27] J.R. Munkres, *Topology, A First Course*. Prentice-Hall, Inc., 1975.

[28] B.K. Natarajan, "On Generating Topologically Consistent Isosurfaces from Uniform Samples," *Visual Computer: Int'l J. Computer Graphics*, vol. 11, no. 1, pp. 52-62, 1994.

[29] T.S. Newman and H. Yi, "A Survey of the Marching Cubes Algorithm," *Computers and Graphics*, vol. 30, no. 5, pp. 854-879, 2006.

[30] G.M. Nielson, "On Marching Cubes," *IEEE Trans. Visualization and Computer Graphics*, vol. 9, no. 3, pp. 283-297, July-Sept. 2003.

[31] G.M. Nielson and B. Hamann, "The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes," *Proc. IEEE Second Conf. Visualization*, pp. 83-91, 1991.

[32] V. Pascucci and K. Cole-McLaughlin, "Parallel Computation of the Topology of Level Sets," *Algorithmica*, vol. 38, no. 1, pp. 249-268, 2003.

[33] J. Patera and V. Skala, "A Comparison of Fundamental Methods for ISO Surface Extraction," *Machine Graphics and Vision Int'l J.*, vol. 13, no. 4, pp. 329-343, 2004.

[34] S. Raman and R. Wenger, "Quality Isosurface Mesh Generation Using an Extended Marching Cubes Lookup Table," *Computer Graphics Forum*, vol. 27, no. 3, pp. 791-798, 2008.

[35] T. Sakkalis, T.J. Peters, and J. Bisceglio, "Isotopic Approximations and Interval Solids," *Computer-Aided Design*, vol. 36, no. 11, pp. 1089-1100, 2004.

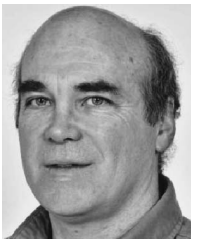
[36] C. Scheidegger, T. Etiene, L.G. Nonato, and C. Silva, "Edge Flows: Stratified Morse Theory for Simple, Correct Isosurface Extraction," technical report, Univ. of Utah, 2010.
[37] J. Schreiner, C. Scheidegger, and C. Silva, "High-Quality Extraction of Isosurfaces from Regular and Irregular Grids," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1205-1212, Sept./Oct. 2006.
[38] W. Schroeder, K. Martin, and W. Lorensen, "An Object-Oriented Approach to 3D Graphics," *Visualization Toolkit,* second ed. Prentice-Hall, 1998.
[39] P. Stelldinger, L.J. Latecki, and M. Siqueira, "Topological Equivalence between a 3D Object and the Reconstruction of Its Digital Image," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 1, pp. 126-140, Jan. 2007.
[40] P. Sutton, C. Hansen, H.-W. Shen, and D. Schikore, "A Case Study of Isosurface Extraction Algorithm Performance," *Proc. Data Visualization,* pp. 259-268, 2000.
[41] A. van Gelder and J. Wilhelms, "Topological Considerations in Isosurface Generation," *ACM Trans. Graphics,* vol. 13, no. 4, pp. 337-375, 1994.
[42] G.H. Weber, G. Scheuermann, H. Hagen, and B. Hamann, "Exploring Scalar Fields Using Critical Isovalues," *Proc. IEEE Visualization,* pp. 171-178, 2002.
[43] L. Zhou and A. Pang, "Metrics and Visualization Tools for Surface Mesh Comparison," *Proc. SPIE: Visual Data Exploration and Analysis VIII,* vol. 4302, pp. 99-110, 2001.

Tiago Etiene received both the BS and MS degrees in computer science from Universidade de São Paulo, Brazil. Currently, he is working toward PhD degree at Scientific Computing and Imaging Institute, University of Utah. His research interests include scientific visualization and computer graphics.

L. Gustavo Nonato received the PhD degree from the Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, in 1998. He joined the Universidade de São Paulo in 1999 and has been an associate professor since 2006. His research interests include visualization and geometry processing. His teaching activities include computer graphics, numerical analysis, and geometric modeling.

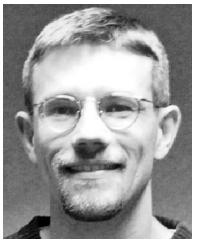
Carlos Scheidegger received the PhD degree in computing from the University of Utah. He is a researcher at AT&T Labs-Research. He has won best paper awards at *IEEE Visualization* in 2007, and *Shape Modeling International* in 2008. His research interests include data visualization and analysis, geometry processing and computer graphics.

Julien Tierny received the PhD degree in computer science from Lille 1 University in October 2008. He is currently a CNRS research associate at Telecom ParisTech, Paris. Prior to his CNRS tenure, he held a Fulbright fellowship (US Department of State) and was a postdoctoral research associate at the Scientific Computing and Imaging Institute at the University of Utah. His research interests include topological and geometrical spatial data analysis for scientific visualization and computer graphics.



Thomas J. Peters received the PhD degree in mathematics from Wesleyan University, with a specialization in topology. He has been a pioneer in establishing the contemporary field of computational topology. His publications span mathematical, computer science, and engineering venues. He holds joint appointments as a professor of computer science and engineering and as a professor of mathematics at the University of Connecticut. He is a cofounder of Kerner Graphics, Inc., for topological algorithms for digital special effects. He has been a senior technical leader at Charles Stark Draper Laboratory, Cambridge, Massachusetts and at Computervision Corporation.

Valerio Pascucci is an associate professor of the School of Computing and Scientific Computing and Imaging Institute at the University of Utah since 2008. Before joining SCI, he served as the data analysis group leader and as a project leader at the Lawrence Livermore National Laboratory, Center for Applied Scientific Computing (from May 2000) and an adjunct professor at the Computer Science Department of the University of California Davis (from July 2005). Prior to his CASC tenure, he was a senior research associate at the University of Texas at Austin, Center for Computational Visualization, CS and TICAM Departments. His research interests include: large data management and analysis, topological methods for image segmentation, progressive and multiresolution techniques for scientific visualization, combinatorial topology, geometric compression, computer graphics, applied computational geometry, and solid modeling. He is a member of the IEEE.



Robert M. Kirby (M'04) received the MS degree in applied mathematics, the MS degree in computer science, and the PhD degree in applied mathematics from Brown University, Providence, RI, in 1999, 2001, and 2002, respectively. He is currently an associate professor of computer science with the School of Computing, University of Utah, Salt Lake City, where he is also an adjunct associate professor in the Departments of Bioengineering and Mathematics and a member of the Scientific Computing and Imaging Institute. His current research interests include scientific computing and visualization. He is a member of the IEEE.

Cláudio T. Silva received the PhD degree in computer science from SUNY-Stony Brook in 1996. He is a full professor of computer science and a faculty member of the Scientific Computing and Imaging Institute at the University of Utah where he has been since 2003. In 2011, he joined NYU's Polytechnic Institute. He coauthored more than 160 technical papers and eight US patents, primarily in visualization, geometric computing, scientific data management, and related areas. He has served on more than 80 program committees, and he is currently in the editorial board of Computing in Science and Engineering, Computer and Graphics, The Visual Computer, and Graphical Models. He was the general cochair of *IEEE Visualization* 2010, and the papers cochair of VIS 2005 and 2006. He received IBM Faculty Awards in 2005, 2006, and 2007, and best paper awards at *IEEE Visualization* 2007, *IEEE Shape Modeling International* 2008 and the 2010 Eurographics Educator Program. His work is funded by grants from the NSF, NIH, DOE, IBM, and ExxonMobil. He is a senior member of the IEEE.