Topology Verification for Isosurface Extraction

Journal:	*Transactions on Visualization and Computer Graphics*
Manuscript ID:	TVCG-2010-07-0166.R1
Manuscript Type:	Regular
Keywords:	verifiable visualization, Isosurface, topology

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 upon 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 even 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. Users of such scientific software are not typically visualization experts. These users 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.

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*. Even when verification 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 the expectations.

Etiene et al. have recently used this method to verify geometrical properties of isosurfacing codes [15], and topological verification follows naturally. 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.

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

- *Etiene, Tierny, Pascucci, Kirby and Silva are with the School of Computing and SCI Institute, University of Utah, USA. E-mail: {tetiene,tierny,pascucci,kirby,csilva}@cs.utah.edu.*
- *Nonato is with ICMC, Universidade de São Paulo, Brazil. E-mail: gnonato@icmc.usp.br.*
- *Scheidegger is with AT&T Labs – Research. E-mail: cscheid@research.att.com.*
- *Peters is with the University of Connecticut, USA. E-mail: tpeters@cse.uconn.edu.*

with non-trivial topological properties, and show that this simple mechanism effectively stresses isosurfacing programs. We assume as input 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 report errors in two publicly-available isosurface extraction codes.

2 Related Work

The literature comparing and evaluating 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 which deal with topological issues that naturally appear in isosurfacing.

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 the 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 remains ambiguities internal to a grid cell: pairs of non-homeomorphic isosurfaces could be homeomorphic when restricted to the grid cell faces. That 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.

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 manufactured solutions (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 popularly used 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 which 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 be used to 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 too 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. What we need are properties which are simple to state, easy to check, and good at catching bugs.

Simple example. Although the previously mentioned problem with Marching Cubes [23] and cracks 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 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 piecewise-linear (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, discuss their relative strengths and weaknesses, and how to robustly check their values. Figure 1 shows our pipeline to assess topological correctness and also the paper organization.

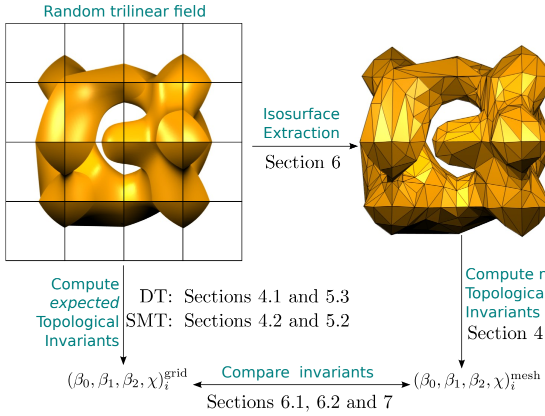


Fig. 1. Overview of our topology verification pipeline. First step, we generate a random trilinear field and extract a random isosurface using the implementation under verification. We then compute the *expected* topological invariants from the trilinear field and compare them against the invariants 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 (SMT).

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, which 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 $\mathscr{M}$, $\chi(\mathscr{M}) = V - E + F$, where V, E and F are the number of vertices, edges and faces of any finite cell decomposition of $\mathscr{M}$. If $\mathscr{M}$ is a connected orientable 2-manifold without boundary, $\chi(\mathscr{M}) = 2 - 2g(\mathscr{M})$, where $g(\mathscr{M})$ is the genus of $\mathscr{M}$. The Euler characteristic may also be written as $\chi(\mathscr{M}) = \sum_{i=0}^{n}(-1)^i \beta_i$, where β_i are the Betti numbers: the rank of the i-th homology group of $\mathscr{M}$. 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 $\mathscr{M}$ has many distinct connected components, that is, $\mathscr{M} = \bigcup_{i=1}^{n} \mathscr{M}^i$ and $\mathscr{M}^i \bigcap \mathscr{M}^j = \emptyset$ for $i \neq j$ then $\chi(\mathscr{M}) = \sum_i^n \chi(\mathscr{M}^i)$. 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

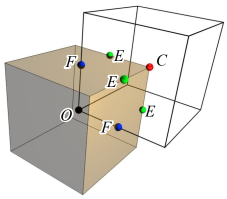


Fig. 2. The four distinct groups of vertices O, F, E, C, are depicted as black, blue, green and red points. They are the "Old", "Face", "Edge" and "Corner" points of a voxel region $V_{\mathscr{G}}$ (semitransparent cube) respectively. For the sake of clarity, we only show a few points.

numbers whenever these are well-defined.

4.1 Digital topology

Let $\mathscr{G}$ be an $n \times n \times n$ cubic regular grid with a scalar $e(s)$ assigned to each vertex s of $\mathscr{G}$ and $t : \mathbb{R}^3 \to \mathbb{R}$ be the piecewise trilinear interpolation function in $\mathscr{G}$, 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 $\mathscr{G}$ 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 $i = 1, \ldots, k$ and
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 Figure 3 shows all possible unambiguous cases.

The geometric dual of $\mathscr{G}$ is called the *voxel grid* associated with $\mathscr{G}$, denoted by $V_{\mathscr{G}}$. More specifically, each vertex s of $\mathscr{G}$ has a corresponding voxel v_s in $V_{\mathscr{G}}$, each edge of $\mathscr{G}$ corresponds to a face in $V_{\mathscr{G}}$ (and vice versa), and each cubic cell in $\mathscr{G}$ corresponds to a vertex in $V_{\mathscr{G}}$, as illustrated in Figure 2. Each voxel v_s can also be seen as the Voronoi cell associated to s. Scalars defined in the vertices of $\mathscr{G}$ can naturally be extended to voxels, thus ensuring a single scalar value $e(v_s)$ to each voxel v_s in $V_{\mathscr{G}}$ defined as $e(s) = e(v_s)$. As we shall show in the following, 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 $\mathscr{G}'$ the $2n \times 2n \times 2n$ regular grid obtained from a refinement of $\mathscr{G}$. Vertices of $\mathscr{G}'$ can be grouped in four distinct sets, denoted by O, F, E, C. The set O contains the vertices of $\mathscr{G}'$ that are also vertices of $\mathscr{G}$. The sets F and E contain the vertices of $\mathscr{G}'$ lying on the center of faces and edges of the voxel grid $V_{\mathscr{G}}$, respectively. Finally, C contains all vertices of $V_{\mathscr{G}}$. Figure 2 illustrates these sets.

Consider now the voxel grid $V_{\mathscr{G}'}$ dual to the refined grid $\mathscr{G}'$. Given a scalar value α, the *digital object* $\mathscr{O}_\alpha$ is the subset of voxels v in $V_{\mathscr{G}'}$ such that $v \in \mathscr{O}_\alpha$ 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 Majority Interpolation (MI) (Figure 5) and it allows us to compute the voxels that belong to a digital object $\mathscr{O}_\alpha$. The middle row of Figure 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 $\mathscr{O}_\alpha$ is two-fold. First, the boundary surface of the union of the voxels in $\mathscr{O}_\alpha$, denoted by $\partial\mathscr{O}_\alpha$ and called a *digital surface*, is a 2-manifold (See the proof by Stelldinger et al. [39]). Second, the genus of $\partial\mathscr{O}_\alpha$ can be computed directly from $\mathscr{O}_\alpha$ using the algorithm proposed by Chen and Rong [7] (Figure 4). As the connected components of $\mathscr{O}_\alpha$ can also be easily computed and isolated, one can calculate the Euler characteristic of each connected component of $\mathscr{O}_\alpha$ from the formula $\chi = 2 - 2g$ and thus β_0, β_1, and β_2.

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

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

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

$$\partial\mathscr{O}_i = cl_{\mathbb{R}^3}\left((\partial\mathscr{O}_\alpha \cap c_i) - \partial c_i\right).$$

We note that $\partial\mathscr{O}$ is a 2-manifold [35], [39]. There are two main parts to the proof presented here. For each i,

1) the 2-manifolds t_i and $\partial\mathscr{O}_i$ are homeomorphic; and
2) both t_i and $\partial\mathscr{O}_i$ 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 Figure 3 [22], either a topological disk or the empty set. Each case in the top row of Figure 3 is the unambiguous input for the MI algorithm to produce the voxel reconstruction show 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

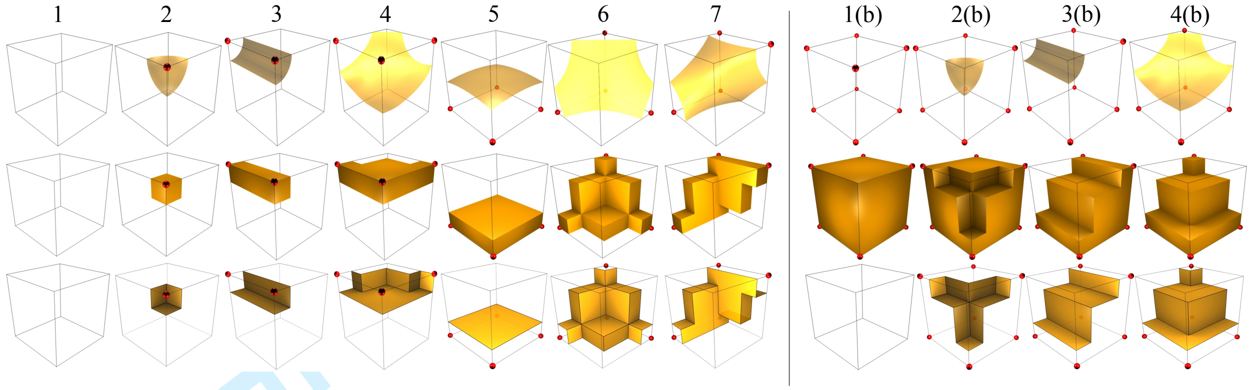


Fig. 3. An illustration of the relation between unambiguous isosurfaces of trilinear interpolants and the corresponding digital surfaces. The top row shows all possible configurations of the intersection of $t = \alpha$ with a cube c_j for unambiguous configurations [22]. Each red dot s_i denotes a vertex withs $e(s_i) < \alpha$. 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 components as the unambiguous configurations (top row).

$\partial\mathcal{O}_i$ (bottom row of Figure 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 Figure 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. □

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.

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

Fig. 4. A simple formula for genus computation.

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

MAJORITYINTERPOLATION($\mathcal{G}, \alpha$)
▷ Let O, F, E and C be the subset of vertices in $\mathcal{G}'$ as described in subsection 4.1.
▷ 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
for $s \in \mathcal{G}'$
  do if $s \in O$ **or**
    $s \in$ F and $|\mathcal{N}(s,O)| = 2$ **or**
    $s \in E$ and $|\mathcal{N}(s,O) + \mathcal{N}(s,F)| \geqslant 4$ **or**
    $s \in$ C and $|\mathcal{N}(s,O) + \mathcal{N}(s,F) + \mathcal{N}(s,E)| \geqslant 12$
      then Select voxel v_s
return $\mathcal{O}_\alpha$

Fig. 5. Voxel selection using Majority Interpolation (MI).

verify properties of isosurfaces with boundary components. We will use some basic machinery from stratified Morse theory (SMT), following the presentation of Goresky'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 $f(x) = \alpha + \varepsilon$ differs only if the interval $[\alpha, \alpha + \varepsilon]$ 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 in the boundary of ε_p satisfying $f(x) < f(p)$ and $f(x) > f(p)$ respectively, then the topological change from the isosurface $f(x) = f(p) - \varepsilon$ to $f(x) = f(p) + \varepsilon$ 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

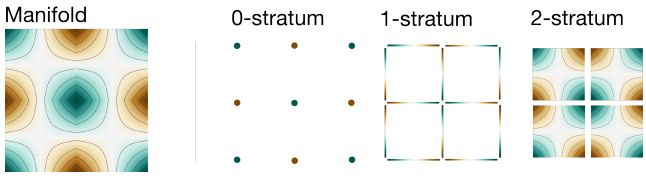


Fig. 6. An illustration of a piecewise-smooth immersed 2-manifold. The colormap illustrates the value of each point of the scalar field. Notice that although the manifold itself is not everywhere differentiable, each stratum is itself an open manifold that is differentiable.

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 $\chi(L^+(p))$:

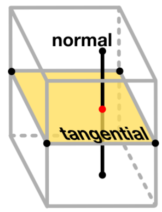
	min	saddle-1	saddle-2	max
$\chi(L^-(p))$	0	2	0	2
$\chi(L^+(p))$	2	0	2	0

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 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 non-differentiable 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 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 Figure 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 $\mathscr{M}$ be the stratified grid described above. It can be shown that if p is a point in a d-dimensional stratum of $\mathscr{M}$, it is always possible to find a $(3-d)$-dimensional submanifold of $\mathscr{M}$ (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 need 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



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

These definitions allow us to classify critical points even in the non-smooth 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 Equation 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:

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

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:

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

Therefore:

$$\begin{aligned}\chi(T_\varepsilon \times N_L^- \cap N_\varepsilon \times T_L^-) &= \chi(N_L^- \times T_L^-)\\ &= \chi(N_L^-)\chi(T_L^-)\end{aligned}$$

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 Equation 4.4. If T_L^- or T_L^+ are one-dimensional, 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 that the normal Morse data is 0-dimensional. This implies that the link will be an empty set with Euler characteristic zero. So, by Equation 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, Equation (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, Equation (4.4) gives:

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

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

A similar analysis can be 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.

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. 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 Figure 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:

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

MMS-SMT($\mathcal{G}$)
1 ▷ Let the input $\mathcal{G}$ be $n \times n \times n$ rectilinear grid
2 **for** $i \leftarrow 1$ **to** #tests
3 **do** $\mathcal{G} \leftarrow$ randomly sampled $n \times n \times n$ grid
4 $CPs \leftarrow$ COMPUTECRITICALPOINTS($\mathcal{G}$)
5 **if** $p \in CPs$ is degenerate **or**
6 p is an internal saddle close to edges or faces
7 **then** GOTO 3
8 **else** $K \leftarrow$ ISOSURFACING($\mathcal{G}$)
9 $(\chi^v)_i \leftarrow$ INVARIANTFROMCPS($\mathcal{G}$)
10 $(\chi^k)_i \leftarrow$ INVARIANTFROMMESH(K)
11 Compare $(\chi^v)_i$ and $(\chi^k)_i$

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

where $\Delta_x = bc - ae$, $\Delta_y = bd - af$, and $\Delta_z = cd - ag$. 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 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 Equation 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 Equation 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

Figure 9 shows the verification pipeline using the MI algorithm, and Figure 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 Figure 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.

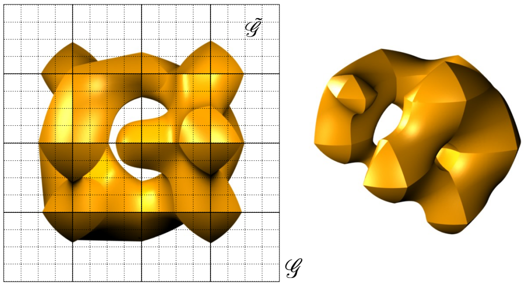


Fig. 8. Our manufactured solution is given by $t(x) = \alpha$. $\mathcal{G}$ is depicted in solid lines while $\tilde{\mathcal{G}}$ is shown in dashed lines. $\tilde{\mathcal{G}}$ is a uniform subdivision of $\mathcal{G}$. The trilinear surfaces t_i are defined for each cube $c_i \in \mathcal{G}$ and 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).

MMS-DS($\mathcal{G}$)
1 ▷ Let the input $\mathcal{G}$ be a $n \times n \times n$ rectilinear grid
2 **for** $i \leftarrow 1$ **to** #tests
3 **do** $\mathcal{G} \leftarrow$ randomly sampled $n \times n \times n$ grid
4 $\tilde{\mathcal{G}} \leftarrow$ REFINEANDRESAMPLE($\mathcal{G}$)
5 **if** $\tilde{\mathcal{G}}$ has ambiguous cubes
6 **then** GOTO 3
7 $\mathcal{O} \leftarrow$ MAJORITYINTERPOLATION($\tilde{\mathcal{G}}$)
8 $K \leftarrow$ ISOSURFACING($\mathcal{G}$)
9 $(\beta_0^v, \beta_1^v, \beta_2^v)_i \leftarrow$ INVARIANTFROMDS($\partial\mathcal{O}$)
10 $(\beta_0^k, \beta_1^k, \beta_2^k)_i \leftarrow$ INVARIANTFROMMESH(K)
11 Compare $(\beta_0^v, \beta_1^v, \beta_2^v)_i$ and $(\beta_0^k, \beta_1^k, \beta_2^k)_i$

Fig. 9. Overview of the method of manufactured solutions (MMS) 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.

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 context, INVARIANTFROMDS is the algorithm illustrated in Figure 4. 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 exists 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, an user-controlled parameter dictates maximum distance for "snap" 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 non-trivial. The results reported in the following section support this statement, showing how complex and error-prone is the coding of isosurfacing algorithms, and reinforcing 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 non-zero 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) 7 cases had duplicated vertices. These cases are illustrated in Figure 11. The first problem is possibly due to the non-smooth 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 datasets 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.

6.1.2 SNAPMC

Table 1 shows that SNAPMC with non-zero snap value causes the mesh to be topologically inconsistent (Figure 13(a)) in more the 50% 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 non-manifold surfaces. This is why there was an increase in the number of mismatches when compared with SNAPMC with zero snap value. Since non-manifold meshes are not desirable in many applications, the authors suggest a post-processing for fixing these topological issues, although no implementation or algorithm for this post-processing 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 number of mismatches we see in Table 1. For instance, Figure 13(b) shows an example of a non-manifold 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 meshes 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 was 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 Section 5.2 and 5.3 were applied to all algorithms although only MC33, DELISO and MCFLOW are expected to generate meshes with the same topology of the trilinear interpolant. Our tests consists of one thousand random fields generated in a rectilinear $5 \times 5 \times 5$ grid $\mathscr{G}$. The verification test using Digital Surfaces demands a compact, orientable, 2-manifold without boundary, so we set scalars equal 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 $\mathscr{G}$. We decide to run these tests using all

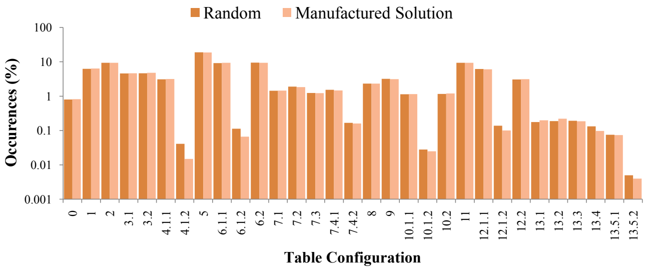


Fig. 10. The horizontal axis shows the case and subcase numbers for each of the 31 Marching Cubes configurations described by Lopes and Brodlie [22]. The dark bars show the percentage of random fields that fits a particular configuration. The light bars show the percentage of random fields which fit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

algorithms for completeness and also for testing the tightness of the theory which says that if the algorithms does 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 SMT procedure is applied. By the time of the submission of this paper, the problem was not fixed. For all non topology-preserving algorithms, there was a high number of mismatches as expected.

One might think that the algorithms described in Figures 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 are still being considered in the tests. Figure 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 rate 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 6 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, producing 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). Figure 12 shows the 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 non-obvious algorithm detail which is not discussed in either Chernyaev's or Lewiner's work is the problem of orientation of some of the cube configurations [20]. The case 13.5.2 shown if Figure 12 (right) is an example of one such configuration where an additional criterion is required to decide the tunnel orientation which 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 an 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 becomes even worse in MCFLOW, because cubes of different sizes are glued together. MCFLOW geometrical convergence is presented in the supplementary material [36].

7 DISCUSSION AND LIMITATIONS

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.

	Consistency (%)	Correctness (%)				
		Digital Surfaces				SMT
	Disk	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

TABLE 1

Rate of invariant mismatches using the PL manifold property, digital surfaces, and stratified Morse theory for 1000 randomly generated scalar fields (the lower the rate the better). 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.

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 submit 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 shows it does not pass in 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. Figure 13(c) 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.

SMT vs. 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 isosurface 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

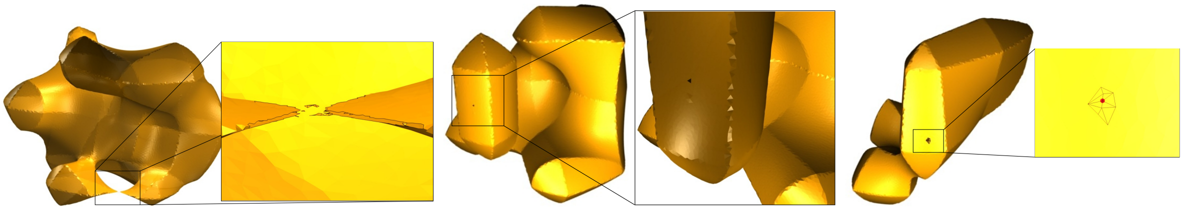


Fig. 11. DELISO mismatch example. From left to right: holes in C^0 regions; single missing triangle in a smooth region; duplicated vertex (the mesh around the duplicated vertex is shown). These behavior induce topology mismatches between the generated mesh and the expected topology.

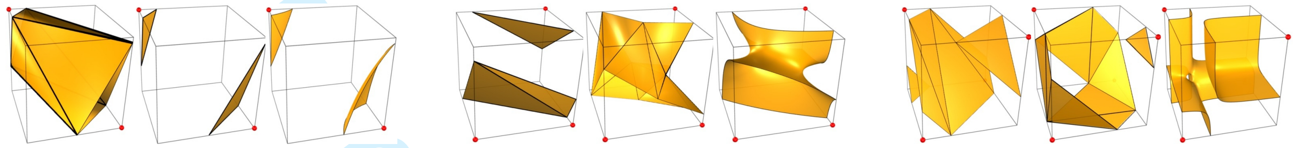
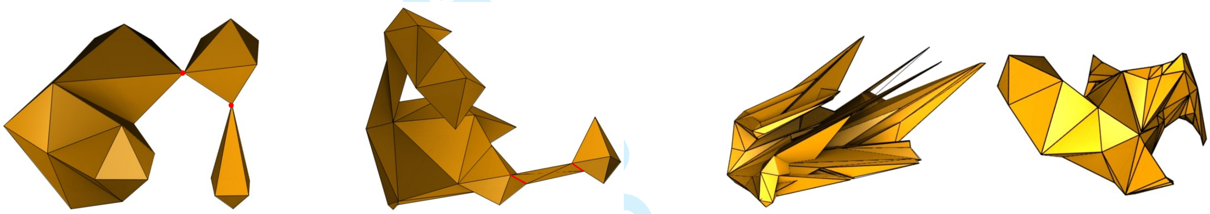


Fig. 12. MC33 mismatch example. From left to right: problem in the case 4.1.2, 6.1.2 and 13.5.2 of marching cube table (all are ambiguous). Each group of three pictures shows the obtained, expected and implicit surfaces. Our verification procedure can detect the topological differences between the obtained 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 non-manifold surfaces due to the snap process. (b) MATLAB generates some edges (red) that are shared by more than two face. (c) MCFLOWbefore (left) and after (right) fixing a bug that causes the code to produce the expected topology but the wrong geometry.

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. Notice that in that case, digital surface based test would be able to detect the spurious three triangles just comparing β_0. Despite being less sensitive in theory, SMT-based verification revealed problems as well as the digital topology tests have. We believe this effectiveness comes in part from the randomized nature of our tests.

Implementation of SMT and DT

Verification tools should be as simple as possible while still effective to reveal 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.

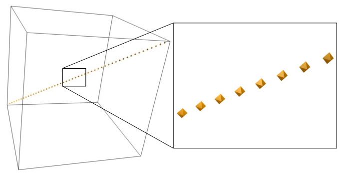
Another concern is the performance of the verification tools. In our experiments, the invariant computation via SMT and DS is faster than any isosurface extraction presented in this paper for most of the random grids. In some scenarios, DS might experience a slowdown because it refines the grid in order to eliminate ambiguous cubes (the maximum number of refinement is set to 4). Thus, for SMT and DS (after grid refinement), both need to perform a constant number of operation for each grid cube to determine the digital surface (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].

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 grid [32]. Contour trees naturally provides 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 good candidate for verification purposes. In fact, if an implementation is available, we encourage its use so as to increase confidence in the algorithms 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 non-trivial. 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 DT method and critical point computation and classification for 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 are focused on the approaches using SMT and DT because of their simplicity and effectiveness as we were able to find 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.

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 $c_i \in \mathscr{G}$ 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 on the right 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 can not even be reproduced by trilinear interpolants, as for example 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.



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 even code 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 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 happens 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 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 motivates the visualization community to develop a culture of verification.

ACKNOWLEDGMENTS

We 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 Engineering*, 193(36-38):4057–4066, 2004.
[2] J. Bloomenthal. Polygonization of implicit surfaces. *Comput. Aided Geom. Des.*, 5(4):341–355, 1988.
[3] H. Carr and N. Max. Subdivision analysis of the trilinear interpolant. *IEEE TVCG*, 16(4), 2010.
[4] H. Carr, T. Moller, and J. Snoeyink. Artifacts caused by simplicial subdivision. *IEEE TVCG*, 12(2):231–242, 2006.
[5] H. Carr and J. Snoeyink. Representing interpolant topology for contour tree computation. In *Topology-Based Methods in Visualization II*. Springer-Verlag, 2008.
[6] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Comput. Geom. Theory Appl.*, 24(2):75–94, 2003.
[7] L. Chen and Y. Rong. Linear time recognition algorithms for topological invariants in 3d. *CoRR*, abs/0804.1982, 2008.
[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 & Graphics*, 24:399–418, 2000.
[10] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37(1):103–120, 2007.
[11] 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.
[12] C. Dietrich, C. Scheidegger, J. Schreiner, J. Comba, L. Nedel, and C. Silva. Edge transformations for improving mesh quality of marching cubes. *IEEE TVCG*, 15(1):150–159, 2008.
[13] H. Edelsbrunner and J. L. Harer. *Computational Topology*. American Mathematical Society, 2010.
[14] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM TOG*, 9:66–104, 1990.
[15] T. Etiene, C. Scheidegger, L. G. Nonato, R. M. Kirby, and C. Silva. Verifiable visualization for isosurface extraction. *IEEE TVCG*, 15(6):1227–1234, 2009.
[16] A. Globus and S. Uselton. Evaluation of visualization software. *SIGGRAPH Comput. Graph.*, 29(2):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*, 24(3):537–545, 2005.
[19] R. Kirby and C. Silva. The need for verifiable visualization. *IEEE Computer Graphics and Applications*, 28(5):78–83, 2008.
[20] T. Lewiner. Personal communication.
[21] 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):38366, december 2003.
[22] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE TVCG*, 9(1):16–29, 2003.
[23] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comp. Graph.*, 21:163–169, 1987.
[24] MathWorks. Matlab. http://www.mathworks.com/products/matlab/.
[25] R. M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. Certifying algorithms. *Computer Science Review*, 2010. to appear.
[26] C. Montani, R. Scateni, and R. Scopigno. A modified look-up table for implicit disambiguation of marching cubes. *The Visual Computer*, 10(6):353–355, December 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. *Vis. Comput.*, 11(1):52–62, 1994.
[29] T. S. Newman and H. Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, 2006.
[30] G. M. Nielson. On marching cubes. *IEEE TVCG.*, 9(3):283–297, 2003.
[31] G. M. Nielson and B. Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd conference on Visualization '91*, pages 83–91, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.
[32] V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38(1):249–268, 2003.
[33] J. Patera and V. Skala. A comparison of fundamental methods for iso surface extraction. *Machine Graphics & Vision International Journal*, 13(4):329–343, 2004.
[34] S. Raman and R. Wenger. Quality isosurface mesh generation using an extended marching cubes lookup table. *Comput. Graph. Forum*, 27(3):791–798, 2008.
[35] T. Sakkalis, T. J. Peters, and J. Bisceglio. Isotopic approximations and interval solids. *Computer-Aided Design*, 36(11):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, University of Utah, 2010.
[37] J. Schreiner, C. Scheidegger, and C. Silva. High-quality extraction of isosurfaces from regular and irregular grids. *IEEE TVCG*, 12(5):1205–1212, 2006.
[38] W. Schroeder, K. Martin, and W. Lorensen. *Visualization Toolkit, An Object-Oriented Approach to 3D Graphics - 2nd 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 Anal. Mach. Intel.*, 29(1):126–140, 2007.
[40] 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.
[41] A. van Gelder and J. Wilhelms. Topological considerations in isosurface generation. *ACM TOG*, 13(4):337–375, 1994.
[42] G. H. Weber, G. Scheuermann, H. Hagen, and B. Hamann. Exploring scalar fields using critical isovalues. In *Proceedings of the conference on Visualization '02*, pages 171–178, Washington, DC, USA, 2002. IEEE Computer Society.
[43] 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.

Topology Verification for Isosurface Extraction

Response to reviewer's suggestions

We would like to thank the reviewers for their insightful reviews. We have addressed the issues that the reviewers brought up by clarifying the text and adding additional discussion on key topics. After a short summary of the changes, we enumerate each suggestion and specify the changes that we made to improve the manuscript.

We would like to start by saying that we do not propose a probabilistic path to the problem of topology verification. Furthermore, there are no formal guarantees on the type of errors that can be found, or on how many errors can be found or the absence of errors. Our approach follows the one proposed by Babuska and Oden [1], and its goal is to increase user confidence in the correctness of the code. Our experiments clearly demonstrate that this approach can uncover subtle bugs.

Another major point raised by one of the reviewers concerns the similarities between our use of Stratified Morse Theory (SMT) and the work of [CS08] and [CM10]. While there is overlap, we have used SMT since it provides for a straightforward way to compute the Euler characteristic.

Reviewer 1:

- **The authors have described the performance on many techniques highlighted in Section 6. While I haven't checked the references in detail, do any of them handle adaptive grids? One of the challenges in handling adaptive grids is the "patching problem"? In that regard, many authors have developed dual methods (e.g. dual contouring) which work well in terms of solving these patching problems. I would like to see some more discussion and evaluation of these methods (esp. what kind of isosurface extraction methods can be handled by current methods). See the useful comparison in: "A Topological Comparison of Surface Extraction ALgorithms", by Andujar et al., CAGD 2005.**

We do not address the address adaptive grids. The focus of our paper is on topology verification of piecewise trilinear cells in a regular grid. This has been clarified in the introduction.

- **There is extensive work in geometric processing literature, which tends to preserve topology of extracted surfaces. It may be useful to compare and contrast the proposed methods with those techniques. Some prominent work includes:**
  (i) "Topology Preserving and Controlled Topology Simplifying Multiresolution Isosurface Extraction", Gerstner and Pajarola, IEEE Vis. 2000.
  (ii) "Global Topology Preservation in Isosurface Extraction of Volumetric Data", Yang and Zhang, Advances in Visual Cmoputing, 2006.
  (iii) "Topology preserving surface extraction using adaptive subdivision", Varadhan et al., 2004, Geometric Processing.

The comparison of different techniques is out of the scope of our work. We focus on the *verification* of the techniques. We will release our verification framework to the community so that anyone can use it for verifying the geometric and topological properties of isosurfacing codes.

Reviewer 2:

- **Abstract improvements.**

We thank the reviewer 2 for the helpful suggestions. The pointed references were indeed insightful. We have made a number of adjustments to improve the communication of our ideas. In particular, we have rewritten the abstract keeping these advices in mind.

- **Other examples of context first for managing reader expectations. intro: parag 2: Increasing confidence in visualization tools is the main goal behind verifiable visualization [17], which aims at developing... --> The main goal of verifiable visualization [17] is to increase confidence in visualization tools by developing... Note how a reader will otherwise read this first as ``confidence is increasing'', then need to backtrack as they learn that this is a goal, not a happening. The rewrite conserves the reader's energy for understanding the ideas rather than disambiguating the prose.**

Fixed.

- **parag 3: By starting with ``simple and effective. Simple methods'' the reader expects next to hear about effective methods, but the word does not appear. Use parallels that you establish as context.**

Fixed.

- **parag 4: ... are concerned with verifying isosurfacing implementations. Specifically, we are interested in their topological properties. Note that the implementation does not have topological properties -- the reader is misdirected to think that ``implementation'' is the context. The non-specific verb phrases ``are concerned with'' and ``are interested in'' allow this sort of imprecision. Perhaps: We want to verify that an implementation computes isosurfaces with desired topological properties.**

Fixed.

- **p.2, point 4): stressing topological properties of the software being tested. [it is not the topological properties but their handling that is stressed. software being tested is context. Perhaps: stress testing software handling of topological properties. OR generating test cases of difficult topological properties for software to handle.]**

Fixed.

- **next parag: ``improves the applicability of the technique under verification'' this cumbersome phrase made me realize that the context for the whole paragraph should be the technique under verification, so that this doesn't have to be said in multiple ways. Also, stress was just used in a different way above. Perhaps: Finally, the work of verification benefits the technique being verified by producing a comprehensive record of the desired properties of its results, along with**

an objective assessment of whether these properties are satisfied. We argue that this record improves the applicability of the technique and increases the value of visualization for the computational science community.

Fixed.

- **sec 2, parag 2: I would prefer ``signs'' to ``polarities'', as no poles are involved, but will defer to the authors' knowledge of the related literature. ``topological matching'' is not a common term, and later ``topological equivalence'' is used. Why not say (and possibly define) isomorphism?**

Fixed.

- **p. 2, col 2, parag 2: assumptions are made on the reconstruction kernel used, [by whom?] --> a specific reconstruction kernel is assumed [changes the noun form ``assumptions'' to a more precise verb than ``made'', even if you stick with passive.]**

Fixed.

- **Still, that approach does not reproduce the topology of trilinear functions as it cannot deal with ambiguities internal to a grid cell. This happens because some non-homeomorphic isosurfaces, when restricted to the cube faces, are in fact homeomorphic. [First, don't change terms: method or methodology becomes approach, and grid cell becomes cube here. Second, word carefully to avoid the seeming self-contradiction.] Their method cannot always reproduce the topology of a trilinear function because there remains potential ambiguity internal to a grid cell: non-homeomorphic isosurfaces can have isomorphic intersections with all grid cell faces.**

Fixed.

- **In fig 2, labeling the vertices with the letters would avoid the mapping to colors, which is an unnecessary additional conceptual step and may be lost in b/w reproduction.**

Fixed.

- **col 2, point 1) separate math with words, not just a comma.**

Fixed.

- **p 5, proof of 4.1: t is reused as the isosurface name and trilinear function.**

The isosurface is $t(x) = \alpha$, which is defined at the first paragraph of section 4.1. No changes were made.

- **p 6, sec 4.2: if we continuously move [we don't move] along a scalar value \alpha, [establish alpha as context, not ``we''] the topology of two isosurface[s] ..mumble.. are different [plural verb with singular subject ``topology''] only when [time is not involved] there is ... [note how critical value comes well before its definition.] --> for a scalar value alpha, the topologies of two**

isosurfaces ..mumble.. differ only if the interval [alpha, alpha+epsilon] contains a critical value. (f(p) is a critical ... [Again, identifying context helps pinpoint prose to repair.]

Fixed.

- **Fig 6 caption should tell the dimension that the manifold has to help the reader interpret the figure.**

Fixed.

- **p.6, table at 43: the text immediately above the number of negative eigenvalues, so the reader is likely to first think that those are the numbers in the table. Perhaps reorder: In a smooth function f, the number of negative eigenvalues of the Hessian matrix determines the index of a critical point p, and the four types give the following the Euler characteristics, chi(L-(p)) and chi(L+(p)):**

Fixed.

- **line 57: intuitively argue about the concept -> argue about the intuitive concepts?**

Fixed.

- **p. 6, col. 2, line 51: point -> points**

Fixed.

- **p. 7, line 54 \delta -> \Delta. Check for others.**

Fixed.

- **Fig 7, line 3: what is the result of random sampling? Are you changing values in the grid, or choosing a point p? The caption should also say what the goal of the method is by specifying input and desired output: is script-G an input or output parameter, or both?**

Fixed. The input parameter G is a nxnxn grid. The random sampling assigns random values to the grid nodes of G.

- **p. 8, line 52: don't you set x,y, OR z and solve a quadratic?**

Fixed.

- **p.9 DelIso description should drop one of the words `` approach'' or ``technique.'' p.9,col 2,line 57: mistakes -> makes mistakes OR is mistaken line 59: delete ``of some implementations under verification''**

Fixed.

- **p.10, line 3: submitted to -> subjected to line 60: topology behavior of meshes --> behavior of mesh topologies sec 6.2: these were not previously called a ``methodology''; perhaps say, ``verification tests'' instead? (change was->were, if you do)**

Fixed.

- **col 2, parag 2: Although our methodology eventually discards scalar fields because either ambiguous cells are still present ... Please be specific of what part of ``our methodology'' since this word is overused in the paper, and please reword so it no longer sounds like all scalar fields are discarded. Perhaps: Although we discard test sets that are found to contain ambiguous cells, or...**

Fixed.

- **p. 11, line 23 were->was line 29 criteria -> criterion**

Fixed.

- **Table 1 should align numbers along the decimal points.**

Fixed.

- **p.12, line 40: presented -> discovered?**

Fixed. presented -> revealed.

- **You have to handle monkey saddles in SMT that are not handled in smooth Morse theory. Below, we prove that saddles do not appear with trilinear interpolants and the scalar fields that we use for verification purposes.**

Since there are many different definitions of monkey saddles, we will take that to mean "degenerate saddles with zero Hessian".

A critical point must appear in one of the four strata of the manifold we are examining, since the strata form a partition.

If the critical point lies in the 3-dimensional stratum, then smooth Morse theory suffices (and smooth Morse theory is what stratified Morse theory becomes when the stratum has ambient dimension) can be employed, since by assumption the Hessian has derivative nonzero (as we describe in the first paragraph of section 5.2, we only verify implementations using non-degenerate scalar fields) and so all critical points are non-degenerate. If the critical point does not lie in the 3-dimensional stratum, then stratification is necessary.

No critical point will appear in the 1-dimensional stratum, since the restriction of the scalar field is linear on all points. The zero-dimensional stratum has no smooth component, and so it does not make sense to talk about Hessians there (and, in any case, equations 4.1 through 4.6 provide the tools to describe the local topology to the necessary degree).

The only remaining case, that of a point in the 2-dimensional stratum, is treated the same way as a point in the 3-dimensional stratum: for the Hessian to be zero inside the face, the quadratic equation of the bilinear interpolant must have a discriminant of zero. We simply check that case and discard the resulting field if it occurs

- **Kurt Mehlhorn and the LEDA team have some nice work on verifying geometric algorithms that could be referenced.**

We have included a reference to Mehlhorn's work in our updated manuscript.

Reviewer 3:

- **Significant portions of it duplicate existing work [CS08][CM10], albeit with a different formalism.**
  In particular, while Stratified Morse Theory (SMT) antedates [CS08], the mechanism discussed in [CS08] was applied to isosurfacing, and should be discussed, especially since it relies on the same identification of critical points in subfaces.

We cite [CS08] work in section 4.2. The reason we have chosen Stratified Morse Theory is that it gives a very straightforward analysis of the change in Euler Characteristic of any given critical point, regardless of the dimensionality of the subface in which the critical point is located and the local configuration of the manifold.

Although the theory in [CS08] and [CM10] does provide a thorough analysis of the topology of the trilinear interpolant, we believe that our approach leads to simple algorithms for verification. Computing $\Chi$ for a lower or upper link literally reduces to 10 lines of code in our approach; a single recursive loop through the cardinalities of the 1-dimensional links.

- **Moreover, a number of the assertions about the topology of isosurfaces in a trilinear cell are insufficiently supported, and either [Nie03] or [CM10] should be referred to.**

We have made those changes based on the recommendations on the attached document.

- **Finally, the premise is that by randomly sampling the isovalues, the probability of discovering errors is high. However, given that topological errors tend to clump around particular features, this is not necessarily true.**

We did not quite understand the remark regarding the clumping of topological errors and features.

In any case, we can not make a statement about the probability of error detection. This is in no small part because we do not have a model for the types of errors which would be effective to cover the wide variety of implementations and possible types of errors.

- **Its weakest aspect is that the authors make insufficiently clear that this is a probabilistic claim about the verification, rather than a strict guarantee.**

The outlined framework does not come with bounded probabilities or formal proofs of correctness. Whether this is possible or not is out of the scope of our paper.

Our approach is based on the framework proposed by Babuska and Oden [1]. We highlight in the third paragraph of Section 1 that verification is a process and the next bug might always be there. The goal of verifiable visualization "is to increase confidence in visualization tools." This is fundamentally different from verification approaches which strive for a proof of correctness.

Extra from reviewer 3:

- **"The discussion of Stratified Morse Theory and the notion of the critical points of the stratified Morse function parallel the definition of 'potential critical points' used in [CS08]: this should be cited and discussed".**

We have added a discussion in the main text.

We argue that using stratified morse theory here is more appropriate. To quote [CS08]:

> 'Define vertices and Morse critical points of a cell to be potential critical points: potential because they may not be global critical points.'

The definition of a potential critical point and the critical points of SMT are in fact related to one another. We argue that our exposition through Stratified Morse Theory compares favorable in simplicity, generality and calculation convenience (the latter one obviously restricted to the computation of the Euler characteristic). Stratified Morse Theory gives a tool which predicts the kind of topological change *any* critical point will have on the level set by a completely local analysis of cardinality of the Morse data, as we describe in the text. The analysis using the state diagrams and machines previously presented is, in our opinion, more complicated. For example it is not clear how can we use the proposed state machines to predict the total final topology of an entire scalar field, a central issue in our work, composed of many such cubes? How should we account for the interaction between the topologies of adjacent cubes? In contrast, the Stratified Morse Theory monograph we cite contains a full description of the machinery necessary for a rigorous proof of our assertions.

- **"data is" vs "data are":**

In accordance to the IEEE style guide, we choose to use "Morse data" as a mass noun.

data: Follow author preference for use as singular or plural, but maintain consistency within an article (unless context clearly demands inconsistency).

- **"verified" is perhaps too strong a term here, as the most that can be claimed is a high probability of consistency. And even that *ought* to come with a proof of bounded probability of incorrect verification.**

This is already discussed previously.

- **\Delta_x, \Delta_y, \Delta_z - given that [Nie03] & [CM10] have already used a different notation for these terms, it would be preferable if the authors used notation consistent with the previous work.**

We adopt the notation from Pascucci and Cole-McLaughlin 2003.

- **"For simplicity, in this paper we refine" - one of the consequences of the analysis in [CM10] is that ambiguity can only be removed by dividing at the isovalues of the critical points. Thus, uniformly doubling the grid resolution should be expected to fail. If it does not, then this bears consideration.**

We agree with the reviewer and, as described in Section 5.3, other subdivision criteria can be used, including subdivisions at critical points. In order to devise an algorithm which is simple to implement, a highly desirable property for verification purposes, we opted for uniform subdivision of grid cubes. As pointed out in the manuscript, a direct consequence of this choice is that sometimes the process fails (as expected), i.e., the maximum allowed number of subdivision is reached and not all resultant cubes are unambiguous, as required by our digital topology framework. At this point, we resampled the scalar field once again and restart the subdivision process from scratch. In our experiments we observed eliminating certain scalar fields does not introduce bias in the frequency of cases and all cases are eventually covered by the verification procedure, as explained in Section 6.2. Although a subdivision scheme based on critical points could lead to fewer subdivisions, and perhaps faster algorithms, our main goal is to show the benefits of this verification framework.