Practical considerations on Marching Cubes 33 topological correctness

Lis Custodio[1], Tiago Etiene[2], Sinesio Pesco[1], Claudio Silva[3]

[1]*Department of Mathematics - Pontifical Catholic University of Rio de Janeiro*

[2]*SCI Institute - University of Utah*

[3]*Polytechnic Institute of NYU*

Abstract

Chernyaev's *Marching Cubes 33* is one of the first algorithms intended to preserve the topology of the trilinear interpolant. In this work, we address three issues with the Marching Cubes 33 algorithm, two of which are related to its original description and one that is related to its variant. In particular, we solve a problem with the core disambiguation procedure of Marching Cubes 33 that prevents the extraction of topologically correct isosurfaces for the ambiguous configuration 13.5. This work closes an existing gap in the topological correctness of Marching Cubes 33. Furthermore, we make our results reproducible, meaning that examples provided in this work can be easily explored and studied. Finally, as part of the philosophy of reproducibility, we provide a corrected version of the Marching Cubes 33 open-source implementation and access to datasets that can be used to verify the correctness of any available topologically correct isosurface extraction implementation that preserves the topology of the trilinear interpolant.

Keywords: Marching cubes, isosurface extraction, trilinear interpolation, topological guarantees

1. Introduction

The Marching Cubes (MC) algorithm [17] is arguably the most popular isosurface extraction algorithm available and is an important component of the toolkits of many visualization experts and researchers. The popularity of the MC algorithm springs from its many advantages, which include simplicity, robustness, and speed. The widespread adoption of MC has resulted in a vast number of improvements in areas ranging from performance to triangle quality and topological correctness. The latter topic is of special interest in this work. We focus on the problem of MC interior ambiguity and, in particular, on the topological correctness of Chernyaev's Marching Cubes 33 (MC33) [2, 15].

Isosurface extraction techniques can be divided into two classes according to their topological guarantees, namely, consistency or correctness. Topologically *consistent* techniques produce surfaces that are piecewise-linear (PL) manifolds (*i.e.*, crack-free surfaces), except at the boundary of the domain. Topologically *correct* techniques produce a PL-manifold homeomorphic to the surface induced by a given interpolant, such as the trilinear interpolant. Although there are many topologically *consistent* MC-based techniques, only a handful are topologically *correct*. Marching Cubes 33 is one of the first MC-based algorithms that aim to preserve the topology of the trilinear interpolant.

Topological correctness increases the complexity of isosurface extraction algorithms. The many isosurface configurations possible for a given interpolant in a cubic grid makes both the algorithm and its implementation a challenging task. As algorithms and implementations become more complex, issues may be overlooked and remain hidden in the multitude of (pseudo-) lines of code. Throughout years of research, it has been shown that some supposedly topologically correct techniques, including MC33, have issues that prevent correctness [7, 16, 21]. In particular, the work of Etiene *et al.* [7] shows that the MC33 implementation by Lewiner *et al.* [14, 15], fails to produce topologically correct isosurfaces. Alas, the authors *do not* provide an explanation for the problem source, let alone fix the problem. They only provide cases that are mishandled by MC33 and a conjecture regarding the root of one of the observed flaws. As we studied the MC33 implementation, we realized that the source of the problem was not merely implementation bugs but the core ideas behind the implemented algorithm. In this work, we address issues with Chernyaev's original algorithm, its extension, and its implementation. Our work closes an existing gap in the topological correctness of Marching Cubes 33.

The subtleties involved in the correctness of isosurface extraction techniques are sometimes difficult to grasp in the ordinary paper medium. Both the geometry and topology inside grid voxels are often complex and challenging to understand, study and replicate (*e.g.*, see Figures 9 and 10 in [22]). As an attempt to bridge this gap, we build on recent efforts towards *executable papers* [13, 31]. Executable papers extend the traditional paper/digital counterpart by including tools that allow readers to interact, explore and verify experiments more easily. In this work, we use executable papers to increase the reproducibility of our results.

Our contributions, which have a practical nature, are the following:

- We explain and address three algorithmic issues and one

non-trivial implementation issue with Marching Cubes 33. In particular, we solve an issue with the core MC33 disambiguation procedure that, as far as we know, has not been addressed elsewhere. Hence, we close an existing gap in the MC33 literature.

- We make our results reproducible. CrowdLabs [31] and Vistrails [8] are used to create an executable paper that can reproduce the results shown in the following sections.
- We provide datasets that can be used to verify the correctness of any topologically correct isosurface extraction technique.

A by-product of this work is a thorough analysis of both the MC33 algorithm and its implementation that can be used by anyone interested in the use or development of correct isosurface extraction algorithms based on MC33. The results of our efforts are materialized into an extended version of the MC33 implementation [14], henceforth called Corrected-MC33 (C-MC33).

This work is organized as follows. Section 3 reviews key aspects related to the Marching Cubes 33 algorithm. Section 4 explains how experiments that uncovered problems in both MC33 algorithm and implementation were conducted. The details of the problems found are shown in Section 5 and the solutions are presented in Section 6. Section 7 shows the results of applying algorithm with different topological guarantees to real-world datasets.

2. Related Work

Soon after the publication of the MC algorithm, the quest for a topologically correct isosurface extraction technique began. A number of approaches were proposed for dealing with cracks, face ambiguity, and, lastly, interior ambiguity. Dürst [6] was the first to point out that some MC cases allow multiple triangulations. A consequence of this is that MC does not always generate topologically consistent surfaces. This problem arises due to the *ambiguity problem*; and the Asymptotic Decider [23] provides a simple and elegant solution to face ambiguity.

The ambiguity problem also occurs in the interior of a voxel. Natarajan [20] was the first to address this problem by adding four new cases to the standard MC triangulation table (subcases of cases 3, 4, 6, and 7). To find the correct subcases, the author proposed a disambiguation procedure based on both face and interior critical points. Nevertheless, the method misses the possibility of two interior critical points in case 7, consequently the proposed algorithm may generate a surface with the incorrect topology [1, 21].

Using a different approach, Chernyaev [2] extended the original MC table to 33 cases – hence MC33; this extension included all the subcases for each ambiguous case. He used the Asymptotic Decider and a new interior ambiguity test to discriminate among subcases. Lewiner *et al.* [15] provided a practical open-source implementation of the Chernyaev algorithm. It is worth noting that some of the configurations shown in Chernyaev's work [2] may have been inspired by personal communication with Nielson [22]. Matveyev [18] introduced an isosurface technique that is also based on an extended table and used the intersections of isosurfaces with cube diagonals to determine the correct case.

Lopes and Brodlie [16] extended the tests proposed by Natarajan. The goals of the work are threefold: i) extract topologically correct isosurfaces; ii) produce geometrically accurate isosurface; iii) allow continuity with respect to changes in threshold and data. Nevertheless, as in Natarajan's work, the method missed the possibility of two interior critical points in case 7 [16]. Cignoni *et al.* [3] also used the test proposed by Natarajan to reconstruct topologically correct isosurfaces. The work of Theisel [30] uses Bézier patches to build G^1 continuous isosurfaces that are topologically correct. Nielson [22] lists all possible cases of a trilinear interpolant inside a cubic grid and builds a topologically correct MC using a three stage algorithm for surface polygonization.

The past two decades have also produced a number of isosurface techniques that are not MC-based. Dual Contouring [12] (DC) is a robust, crack-free, isosurface extraction technique that works on the dual grid. Several improvements over Dual Contouring have been proposed: Schaefer *et al.* [24] address the issue of non-manifold surfaces generated by DC; Varadhan *et al.* [32] combine a signed distance field with DC to reconstruct details such as thin features; and Zhang *et al.* [33] use DC for topology-preserving simplification of isosurfaces. Note that none of these techniques are intended to preserve the topology of the trilinear interpolant. Dey and Levine [5] presented an algorithm that computes a Delaunay triangulation based on the intersection between the isosurface and the 3D Voronoi diagram. Another paradigm for isosurface extraction is the advancing front method. Advancing front algorithms build a triangulated surface by progressively adding triangles to an implicit surface [10], possibly creating several fronts that are simultaneously advanced one triangle at a time. A number of extensions have been proposed for advancing front techniques [25, 26, 29].

In the following sections, we focus on MC33. Note that, although many of the algorithms presented previously are topologically consistent, only a handful of them are topologically correct [2, 5]. Also, the implementation of a topologically correct isosurface extraction algorithm is non-trivial. Hence, once the algorithm is implemented, topological guarantees, both consistency and correctness, may be lost because of algorithm or implementation issues, as shown in the work of Etiene *et al.* [7]. Although it has been ten years since the publication of MC33, we believe it is important to correct a mistake in the algorithm that has gone unnoticed since Chernyaev published it almost 20 years ago. In this work we aim to close an existing gap in the MC33 literature. Furthermore, we aim not only to provide a correct algorithm but verify that our modified implementation is faithful to the correct algorithm. We explain the issues and propose solutions for both algorithm and implementation. We note that "MC33" may refer to either the Marching Cubes 33 *algorithm* presented in Chernyaev's work [2] or its *implementation*, as in Lewiner *et al.* [15] depending on the context.

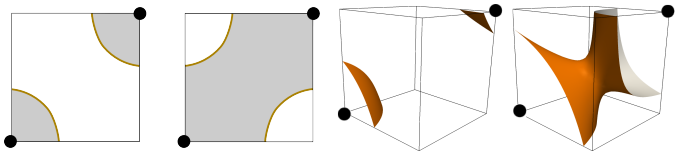


Figure 1: Left: case 4 ambiguity. The interior ambiguity test proposed by Chernyaev is used choose the correct configuration.Right: face ambiguity. The Asymptotic Decider is used to resolve the ambiguity.

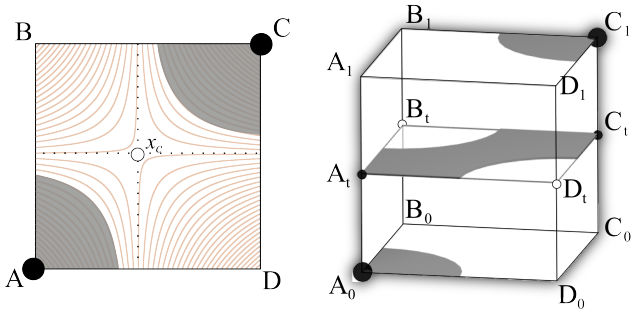


Figure 2: Asymptotic Decider (left) and MC33 interior ambiguity test for MC case 4 (right). The gray areas represent regions with positive scalar values, and the capital letters represent the scalar value at each vertex. In the left image, we observe that $f(x_c) < 0$, where x_c is the saddle point position. Positive areas will be connected if $f(x_c) > 0$. The orange squared plane shown in the right image represents the cutting-plane. The goal of the MC33 algorithm is to find a cutting-plane such that the gray areas in the top and bottom planes are joined in the interior.

3. Preliminaries

In this section we present the notation that will be used throughout the paper. We also briefly review the main concepts behind Chernyaev's algorithm and Lewiner *et al.*'s improvements to it. Let G be a rectilinear grid with scalar values associated with each vertex $x_j \in G$. Let $g : \mathbb{R}^3 \rightarrow \mathbb{R}$ be a piecewise-trilinear interpolation function defined on G. Given an isovalue λ, the isosurface S_λ is defined as the set of points for which $g(x) = \lambda$. For each voxel $v_i \subset G$, and $x \in v_i$, $g(x) = g_i(x)$ where g_i is the trilinear interpolant inside the cubic cell v_i.

The output of MC-based algorithms is a piecewise-linear mesh M_λ, and we say that an algorithm and its implementation are *topologically correct* if M_λ is homeomorphic to S_λ. Without loss of generality, we assume that $\lambda = 0$, and thus $S_\lambda = S_0 = S$. We say that a point x is *positive* (*negative*) if $g(x) > 0$ ($g(x) < 0$).

Given a voxel v_i, and a cutting-plane P parallel to one of v_i's faces, define $f_i : \mathbb{R}^2 \rightarrow \mathbb{R}$ as the bilinear interpolant along P. Note that $f_i(x) = g_i(x)$ for $x \in P$. Throughout the text, we deal with a single voxel v; thus, we omit the subscript i. We also assume that v and P are defined in the domains $[0, 1]^3$ and $[0, 1]^2$, respectively.

3.1. *Chernyaev's* MC33

The two pillars of Marching Cubes 33's topological correctness are Nielson and Hamann's Asymptotic Decider and Chernyaev's interior ambiguities test; together these solve the face ambiguity and interior ambiguity problems in the Marching Cubes 33 algorithm. A face ambiguity occurs when face vertices have alternating signs. That is, one face diagonal is positive (both vertices are positive) and the other is negative (both vertices are negative). In this case, the signs of the face vertices are insufficient to determine the correct way to triangulate the isosurface. Similarly, an interior ambiguity occurs when the signs of the cube vertices are insufficient to determine the correct surface triangulation, *i.e.*, when multiple triangulations are possible for the same cube configuration (see Figure 1).

The idea behind the Asymptotic Decider is to verify the face saddle sign and compare it to the sign on the face vertices. A positive saddle means that the positive face vertices are connected; consequently, the positive face vertices are separated if the face saddle point is negative (see Figure 2). To compute the face saddle sign, the saddle point position x_c must be computed [2]:

$$x_c = \left(\frac{A - D}{A + C - B - D}, \frac{A - B}{A + C - B - D}\right), \quad (1)$$

where A, B, C and D are the scalar values at the face vertices (see Figure 2). The sign of x_c can easily be checked by replacing Equation (1) into the bilinear interpolant:

$$f(x_c) = \frac{AC - BD}{A + C - B - D}. \quad (2)$$

For an ambiguous face, assuming A, C positive and B and D negative, the denominator of the Equation (2) is always positive (see Figure 2). Then, the face ambiguity is solved by evaluating the sign of the numerator of $f(x_c)$.

Due to the interior ambiguity, the Asymptotic Decider alone cannot solve the topological correctness problem. Chernyaev uses the idea behind the Asymptotic Decider to solve the interior ambiguity problem. The proposed test uses a sweeping cutting-plane to evaluate the behavior of the trilinear interpolant inside the cube.

Given a cube with an ambiguous configuration, define the scalar values at the base and top planes as A_0, B_0, C_0, D_0 and A_1, B_1, C_1, D_1, respectively (see Figure 2). Let A_0 and C_1, the vertices to be tested, be positive. Observe that, although A_0 and C_1 belong to opposite cube faces, they can be connected through the cube interior. In other words, there may exist a path from A_0 to C_1 passing through the voxel interior for which all points belonging to that path are positive. To determine whether A_0 and C_1 are connected, Chernyaev begins by observing that the saddle points at the top and base cube faces are negative, *i.e.*, Equation (2) is negative at the bottom and top faces. Since the denominator is positive, it follows that:

$$A_0C_0 - B_0D_0 < 0 \quad (3)$$

$$A_1C_1 - B_1D_1 < 0. \quad (4)$$

Then, if there is a plane cutting the cube such that its saddle point is positive, it means that there is a positive area crossing the cube, *i.e.* the positive vertices are connected inside the cube. In other words, the Chernyaev interior test searches for a t for

which:

$$F(t) = A_t C_t - B_t D_t \quad > \quad 0. \tag{5}$$

This can be achieved by solving a second order equation in t. Replacing $X_t = X_0 + (X_1 - X_0)t$, $X \in \{A, B, C, D\}$ and $t \in [0, 1]$ in Equation (5) one obtains a second order equation in t:

$$\begin{aligned} F(t) &= A_t C_t - B_t D_t & (6) \\ &= at^2 + bt + c, & (7) \end{aligned}$$

where a, b, and c are functions of A, B, C, and D (see Appendix A). Chernyaev concludes that positive vertices A_0 and C_1 are connected through the cube interior if:

i $a < 0$;
ii $t_{\max} = -b/2a \in (0, 1)$;
iii $F(t_{\max}) > 0$.

If one of the above conditions fails, the positive vertices are separated.

3.2. *Lewiner* et al.*'s* MC33

Lewiner *et al.* [15] proposed a modification of Chernyaev's interior test. In this modification, they use an alternative method for computing the height plane t for most ambiguous cases. For cases 6, 7, 12, and 13, the authors compute the height t based on the barycenter of the end vertices of an edge e (a cube edge intersected by the isosurface) weighted by the values of the scalar field on these vertices (see [15] for details). In practice, the implementation uses:

$$t_{\text{alt}} \quad = \quad \frac{V_0}{V_0 - V_1}, \tag{8}$$

where V_0 and V_1 are the scalar values at the vertices of e. Note that this is equivalent to finding the intersection point between the isosurface S and e. The authors keep the structure of the test proposed by Chernyaev, but condition (i) is not used, and condition (ii) is always true because e is an edge intersected by the isosurface; consequently, $t_{alt} \in (0, 1)$.

Section 5 explains why the algorithm proposed by Chernyaev and its modified version proposed by Lewiner *et al.* may fail to extract surfaces that are topologically correct. In the following section, we present the tools we use to detect, debug, and reproduce the issues found in the MC33 algorithm and its implementation.

The full Marching Cubes table can be found in the works of Chernyaev [2] and Lewiner et al. [15].

4. Experiments setup

We begin by investigating the source of topological problems in the MC33 implementation [7]. The topological issues described were obtained by systematically stress-testing the implementation over many topological configurations using the verification framework proposed in Etiene *et al.* [7]. These authors' algorithm can be summarized as follows. (I) A random scalar field G is built by uniformly sampling scalar values in the range $[-1, 1]$ for each $x_j \in G$. (II) The *expected topological invariants* are obtained directly from S, *i.e.*, without extracting the isosurface of interest. The topological invariants used are the Euler characteristic $\chi(S)$ and the Betti numbers $\beta_k(S)$. (III) The MC33 implementation is used to extract a piecewise linear mesh M, and its invariants $\chi(M)$ and $\beta_k(M)$ are computed. (IV) Lastly, the pairs of topological invariants $\{\chi(S), \chi(M)\}$ and $\{\beta_k(S), \beta_k(M)\}$ are compared. A mismatch indicates that a problem has occurred. Nevertheless, as the authors note, a match between invariants does not imply a bug-free code [7]. The verification process does not prove the absence of bugs but only increases one's confidence in its correctness. In this paper, we exploit the fact that when the expected and obtained surfaces are not homeomorphic, a counterexample is given in the form of a scalar field G and a mesh M. We use this information to find and correct errors in MC33.

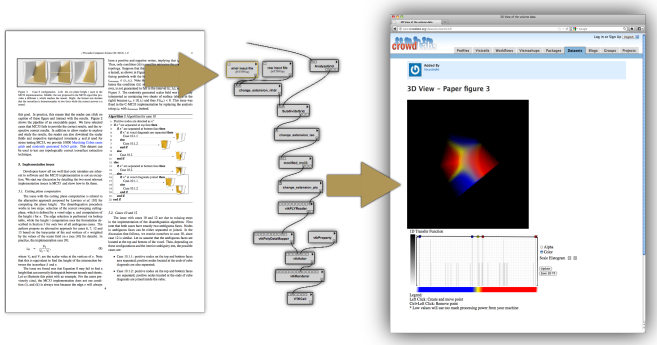


Figure 3: The executable paper pipeline. An image is made executable (left), meaning that the reader can launch a request to execute a pipeline in a remote server (middle) and interact with the result in a web browser (right) [4].

4.1. Reproducibility

As investigators in a mature field within the scientific visualization community, isosurface extraction researchers have developed ways to help other researchers and practitioners reproduce their results. Published journal articles offer a first approximation of reproducibility. Nevertheless, many details regarding implementation, source code, input data, and other types of information are often omitted. Many, but not all, published techniques make source code and input data freely available, and some are part of widely used visualization packages such as VTK [27]. This practice greatly increases the degree of reproducibility of the work. In this paper, we strive to increase the degree of reproducibility of the work presented by making the results shown in Figures 8 and 9 and in Algorithm 2 fully reproducible. We use CrowdLabs [31] and Vistrails [8, 28] as a platform to achieve this goal. To explore some of the results shown in this paper, the reader may click on individual figure captions and interact with the results via web browser. Figure 3 shows the pipeline of an executable paper. We have selected cases in which MC33 fails and have provided the respective correct results. In addition, to allow the reader to explore and study the results presented here, he or she can also download the scalar fields and respective topological invariants χ and β

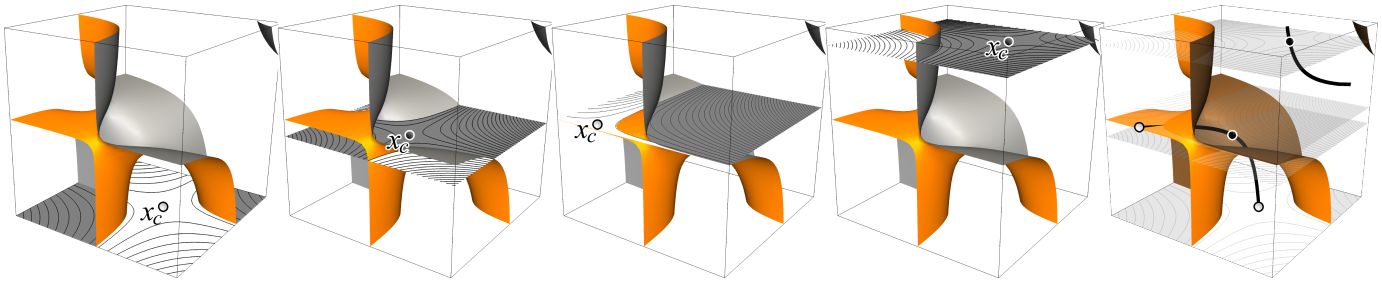


Figure 4: Sign changes of the cutting-plane saddle point as a function of the height t. The gray area depicts $f(x) > 0$. The black (resp. white) dots are face saddles with $f(x_c) > 0$ (resp. $f(x_c) < 0$). From left to right, the four leftmost images show the sign of the face saddle points changing from negative to positive to negative and to positive again, respectively. The rightmost image shows the hyperbolic trajectory of the face saddle position $x_c(t)$. The MC33 algorithm fails to track the saddle point sign because it ignores the influence of the hyperbolic trajectory shown here.

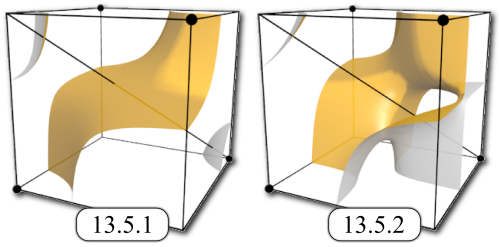


Figure 5: Challenging cases for Chernyave's interior test: voxel diagonal has vertices with opposite signs. Case 13.5.2 needs to be oriented correctly. One of the diagonal vertices is isolated from all other vertices in the cube, while the other is faced by the tunnel. In order to determine which vertex is isolated, we apply the same tool used for disambiguation of case 13.5. For case 13.5.1, the orientation of the isosurface have no influence on the topology.

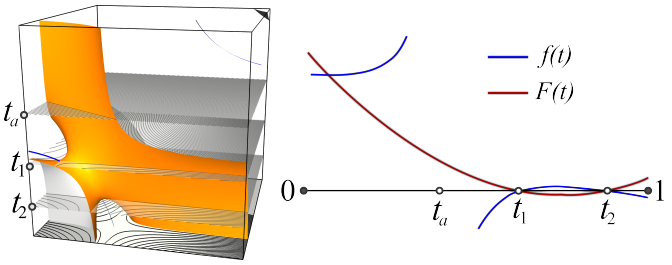


Figure 6: Counterexample to Chernyaev's core disambiguation algorithm. The MC33 algorithm incorrectly interprets case 13.5.2 as 13.5.1. The left image shows the zero-level set for case 13.5.2 and cutting-planes at heights t_1, t_2, and t_a, which correspond to both roots of $F(t)$ and the asymptote of $f(x_c(t))$, respectively. The blue ribbon shows the path of the face saddle $x_c(t)$. The right image shows the changes in $f(x_c(t))$ and $F(t)$. According to the three criteria of the MC33 algorithm described in Section 3, the upward-facing red parabola defines the absence of a tunnel (condition (i)), which is incorrect. The blue curve, on the other hand, shows the correct sign change.

used for stress testing MC33. We also provide 10000 Marching Cubes cases grids and randomly generated 5x5x5 grids [4]. This dataset can be used to test any topologically correct isosurface extraction technique.

5. Issues with the MC33 algorithm and its implementation

In this section, we discuss specific issues regarding both Chernyaev [2] and Lewiner *et al.*'s [15] work. Because Lewiner *et al.* extends Chernyaev's work, the issues presented in the latter are also part of the former. Specifically, we detail three *algorithmic* issues – two in Chernyaev's MC33 and one in Lewiner *et al.* – and one *implementation* issue. The solutions for the issues raised here will be presented in the next section.

This section is organized as follows. First, we explain an algorithmic problem with the MC33 core disambiguation procedure. This issue has not been discussed in the literature to date. We then discuss a second algorithmic problem related to the triangulation table and the extraction of non-manifold meshes. Although this problem has been discussed in the literature, we discuss it here for completeness and because we provide an alternative solution to the problem (see Section 6). Next, we show a third algorithmic problem related to the alternative approach proposed by Lewiner *et al.* for computing the height plane t. Lastly, we show a non-trivial problem with the open-source implementation of the MC33.

5.1. Issue I – Case 13.5

Here, we show a problem with the core disambiguation procedure described in Chernyaev's work. To our knowledge, this problem has not been exposed or addressed in the literature.

Case 13 is certainly the most complex table case; all faces are ambiguous, and six subcases are possible. Four of the subcases can be discriminated by using Asymptotic Decider. The remaining cases 13.5.1 and 13.5.2 require Chernyaev's MC33 interior ambiguity resolution method. Recall that the MC33 approach discriminates between tunnels and isolated sheets by finding a cutting-plane for which positive nodes in the cube diagonal are joined by points in the interior of the cubic cell (see Figure 2). Cases 13.5.1 and 13.5.2 differ precisely because the positive nodes in case 13.5.2 are connected to one another by interior points, which is not true for 13.5.1 (see Figure 5).

Although it seems that the MC33 methodology described in Section 3 fits naturally in this scenario, as it turns out this disambiguation procedure *cannot* be applied for 13.5. Let us illustrate this point with an example. Figure 4 shows the expected changes in the sign of the saddle point x_c as a function of the height t. Mathematically

$$x_c(t) = \left(\frac{A_t - D_t}{A_t + C_t - B_t - D_t}, \frac{A_t - B_t}{A_t + C_t - B_t - D_t} \right). \quad (9)$$

It follows that the face saddle value (and thus sign) is also defined as a function of t:

$$f(x_c(t)) = \frac{A_t C_t - B_t D_t}{A_t + C_t - B_t - D_t} \quad (10)$$

$$= \frac{at^2 + bt + c}{A_t + C_t - B_t - D_t}. \quad (11)$$

As can be seen in Figure 4, from left to right, as the plane height t changes, the value of the face saddle $f(x_c(t))$ changes from negative to positive to negative and to positive again. These changes occur at the roots t_1 and t_2 of $f(x_c(t))$ and the asymptote of $f(x_c(t))$, *i.e.*, the root t_a of the denominator of f (see left image in Figure 6). Thus, in total, three sign changes will occur. The rightmost image in Figure 4 shows the path traced by the face saddles $x_c(t)$; as t grows, there is a "jump" not only in the sign of $f(x_c(t))$ but also in the position of $x_c(t)$. The change occurs precisely when the height t passes through the asymptote of $f(x_c(t))$.

Nevertheless, contrary to what is expected, the polynomial $F(t)$ (Equation (7)), used by Chernyaev's MC33 algorithm for tracking the sign of the saddle point, is a second order equation in t and thus can only allow for two sign changes. Therefore, the sign tracked by the MC33 algorithm will not match the expected one at some point. Because the sign of the saddle points is embedded in all three conditions for verifying the presence or absence of tunnels, MC33 will eventually provide a wrong result.

The source of the problem can be tracked to Equations 3 and 4 and the assumption that the denominator of $f(x_c)$ (Equation (2)) is positive. These assumptions can easily be verified to be true for case 4, shown in Figure 2. However, for case 13, the saddle points at the top and bottom planes have opposite signs, which contradicts Equations (3) and (4). In addition, the denominator $A + C - B - D$ of $f(x_c)$ changes its sign at the asymptote of $f(x_c)$, contrary to the assumption that it is always positive. The consequence of incorrectly tracking sign changes is that the three rules used for resolving internal ambiguity will fail for some scalar fields. As an example, Figure 6 shows a case 13.5.2 that will mistakenly be taken as case 13.5.1 because $a > 0$ characterizes multiples surface sheets instead of a tunnel (see also Appendix A). The problem is not only related to the misclassification of case 13.5.2 as 13.5.1. We have also devised examples in which case 13.5.1 is mistakenly taken as case 13.5.2 because the three criteria shown in Section 3 hold. Thus, Chernyaev's interior ambiguity test does not always yield topologically correct isosurfaces.

5.1.1. Tunnel orientation

A second minor issue regarding case 13.5.2 is the tunnel orientation of configuration 13.5.2. Once case 13.5.2 is determined, one needs to properly orient the tunnel inside the voxel. The inline figures show the two possibilities. Both vertices at the voxel diagonal are separated from all other voxel vertices at the voxel faces (note that this is not the case for other vertices). Nevertheless, either the positive or the negative vertex of the cube diagonal will connect with vertices with the same sign through the voxel's interior. This will determine which vertex

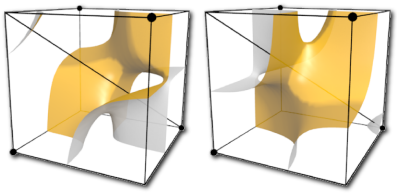


Figure 7: Two possible tunnel orientations for case 13.5.2. The difference between them is the location of the positive vertex.

is isolated and which is facing the tunnel. This problem with the tunnel orientation is not dealt with or mentioned in either Chernyaev or Lewiner *et al.*'s work. Nevertheless, it was briefly mentioned in Etiene *et al.* [7], but no solution to the problem was provided. As the authors observed, the isosurface topology changes if the tunnel orientation is incorrect; thus, it must be oriented correctly. The section 6.1.1 provides a solution for this issue.

5.2. Issue II – Non-manifold surfaces

The second algorithmic issue is related to the triangulation table used to build triangulated surfaces. The choice of the correct MC configuration is only part of the process of building an algorithm that preserves the topology of the piecewise-trilinear field. The voxel triangulation table is, in fact, the determinant of the final mesh topology. Chernyaev's original triangulation table contains cases that lead to topologically inconsistent non-manifold meshes in scenarios such as the one shown in Figure 8. This problem occurs because the MC33 triangulation table allows faces that are coplanar with the grid voxel faces. Hence, when neighbor voxels have "tunnels" in their interiors, and share an ambiguous, coplanar face, the end result will be non-manifold edges, as shown in Figure 8. Because this is an issue with the triangulation table, any topologically correct algorithm whose table is based on Chernyaev's triangulation table will build non-manifold surfaces whether or not the algorithm can correctly distinguish the voxel cases.

This problem with Chernyaev's work was pointed out by Lopes and Brodlie [16] (following earlier work by Van Gelder and Wilhelms [9]) and is one of the motivations of Lopes and Brodlie's work on topologically correct and geometrically accurate isosurface extraction algorithm [16]. Lopes and Brodlie aimed at improving the geometry quality of the trilinear surface patches and consequently solving the topology problem. They achieve this goal by adding points to the voxel faces as well as to the voxel interior. These extra points are placed on the trilinear patch which increases geometry accuracy. They are classified into three different classes and used for extending the contour of the trilinear patch with the voxel faces. The implementation of this technique becomes intricate and error-prone due to the additional steps required for voxel triangulation.

5.3. Issue III – Cutting-plane computation

The third algorithmic issue is related to an MC33 improvement proposed by Lewiner *et al.* [15] for computing the plane

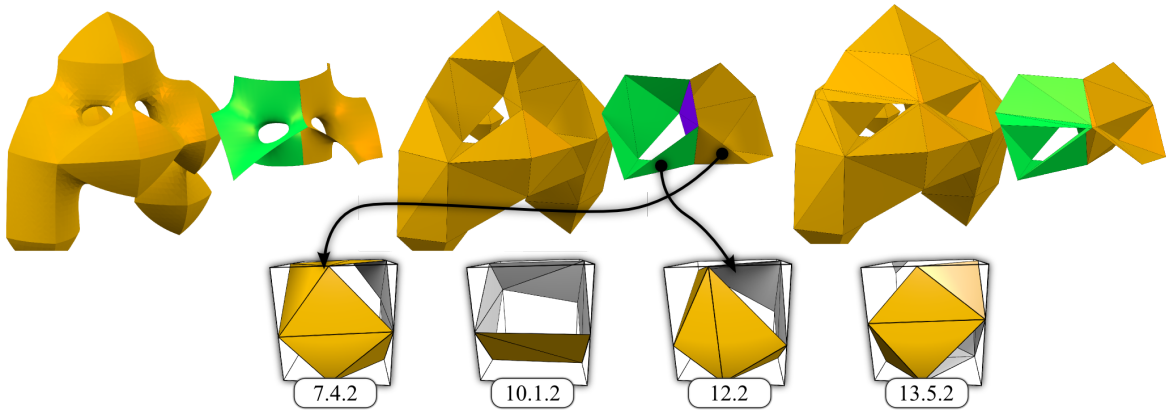


Figure 8: Top: Problem with Chernyaev's triangulation table. The figure shows the zero level-set of a $5 \times 5 \times 5$ randomly generated piecewise-trilinear scalar field G (left) and two meshes extracted using the MC33 (center) and C-MC33 (right) algorithms. The isolated voxel patches, shown in green and yellow, represent the two voxels at the center of G. The face shared by two consecutive tunnels, shown in purple, generates non-manifold edges. After one subdivision at the critical point of this case, the problem no longer occurs, and a valid manifold surface is obtained (right). Bottom: Triangulation for tunnels used by Lewiner *et al.* [15]. Each has a face that is coplanar to the voxel faces, which may lead to non-manifold surfaces. [4]

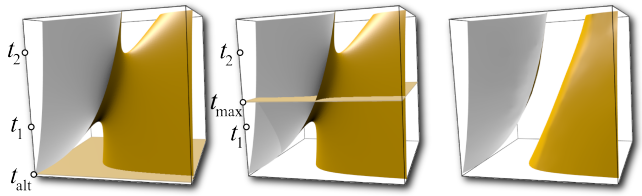


Figure 9: Case 6 configuration. Left: the cut plane height $t = t_{alt} > 0$ used in the MC33 *implementation*. Middle: the test proposed in the MC33 *algorithm* provides a different $t = t_{max} > 0$, which reaches the tunnel. Right: the former test decides that the isosurface is homeomorphic to two discs whereas the correct answer is a tunnel. [4]

height. The problem is that Equation (8) may fail to find an appropriate height that can correctly distinguish between tunnels and surface sheets. Let us illustrate this point with an example. For the cases previously cited, two of the conditions in the Chernyaev interior test described in Section 3 are not used. The MC33 implementation does not use condition (i), and (ii) is always true because the edge e will always have a positive and a negative vertex, implying that $t_{alt} \in (0, 1)$. Thus, only condition (iii) is used in retrieving the correct voxel topology. Suppose that the scalar field in a given voxel defines a tunnel, as shown in the left image in Figure 9. In this case, to retrieve the correct topology, $F(t)$ should be a downward-facing parabola with both roots $t_1, t_2 \in (0, 1)$, $t_1 < t_2$, and $t_{max} \in (t_1, t_2)$. In this case, $F(t) > 0$ only for $t \in (t_1, t_2)$; hence, $F(t_{max}) > 0$, and a tunnel is retrieved according to condition (iii). The problem with the alternative approach is that, as shown in Figure 9, the solution to Equation (8) is not guaranteed to fall within the (t_1, t_2) interval, which implies that the scalar field may be incorrectly interpreted as containing two sheets of surface (shown on the right). In other words, because $t_{alt} \in (0, t_1)$ and $F(t_{alt}) < 0$, condition (iii) verifies the absence of a tunnel.

5.4. Issue IV – Case 10

The last issue described in this work is related to the implementation of MC33. Developers know all too well that code mistakes are inherent to software and the MC33 implementation is not an exception.

Due to a missing step in the implementation of the disambiguation algorithm, MC33 fails to correctly resolve the ambiguity in cases 10 and 12. Note that both cases have exactly two ambiguous faces and the nodes in ambiguous faces can be either separated or joined. In the discussion that follows, we restrict ourselves to case 10; case 12 is similar.

Let us assume that the ambiguous faces are located at the top and bottom of the voxel. Then, following the algorithm proposed by Chernyaev [2], depending on the sign of the face saddles and the interior ambiguity test, one can identify the correct case (see also Algorithm 2):

- Case 10.1.1: the positive nodes on both faces are separated, and the positive nodes at cube diagonals are also separated;
- Case 10.1.2: the positive nodes on both faces are separated, and the positive nodes at the cube diagonals are not;
- Case 10.2: the positive nodes are separated on the top and connected on the bottom face.

The cases shown above assume that the positives nodes at the top face are separated. But a similar reasoning must be applied to cases in which the positives nodes at the top faces are joined. In the Lewiner *et al.*'s implementation the possibility that the positive nodes at the top faces are joined is missing.

6. Solutions

We present solutions for the four issues raised in the previous section.

6.1. Issue I – Case 13.5

The disambiguation of case 13.5 has been approached in different ways for different frameworks for isosurface extraction. For example, Nielson [22] presents an algorithm that is concerned with connectivity along edges, faces and the voxel interior. The author presents a detailed description of the behavior of the trilinear interpolant inside the cubic grid and uses these descriptions to solve the ambiguity problem in the interior. Lopes and Brodlie [16], on the other hand, use critical points in order to resolve some ambiguities. In this case, the sign of the critical point determines the correct configuration. Unfortunately, the above solutions do not seamlessly integrate with the MC33 algorithm. The core idea for solving interior ambiguity, namely, that tunnels can be detected by a sweeping plane through the voxel, is absent in both approaches. This motivated us to devise an alternative solution that we feel follows the idea presented in the original algorithm.

We solve this problem by proposing a new interior test that uses the fact that case 13.5.2 requires both *roots* t_1 and t_2 of $f(x_c(t))$ and the associated saddle points to be inside the voxel. First, recall that $x_c(t)$ tracks the path of the face saddle inside the voxel as a function of height plane at height t, and $f(x_c(t))$ tracks the value (and thus the sign) of that saddle. Both functions are illustrated in the rightmost image in Figure 4, in which the black hyperbolic curves represent the path of $x_c(t)$ and the color of the circles represents the sign of the face saddle at a given point (white and black circles are points with negative and positive values, respectively). For case 13.5.2, the path traced by the curve $x_c(t)$ must intersect the isosurface tunnel twice, once at each of the roots t_1 and t_2 of $f(x_c(t))$. This implies that both saddle points $x_c(t_1)$ and $x_c(t_2)$ must lie inside the voxel. This is not the case for 13.5.1 because the face saddle can cross the middle sheet at most once. Therefore, it suffices to verify that both roots of $f(x_c(t))$ and its saddle points are inside the voxel. Algorithm 1 illustrates our solution. Our algorithm is very simple, and does not require the computation of the critical points of the trilinear interpolant, or a detailed description of its behavior inside a voxel. Our algorithm uses the ideas proposed by Chernyaev in order to fix an algorithmic problem in his work. We have implemented and tested this solution on C-MC33 using over 10000 randomly generated instances of case 13.5.

Algorithm 1 A simple disambiguation procedure for Case 13.5

CASE 13.5(a, b, c)

▷ Let t_1 and t_2 be the roots of $at^2 + bt + c$ (Equation (7))
1 **if** $t_1, t_2 \in (0, 1)$ and $x_c(t_1), x_c(t_2) \in (0, 1)^2$
2 **then return** Case 13.5.2
3 **else return** Case 13.5.1

6.1.1. Issue II – Tunnel orientation

To find the correct tunnel orientation one can use the sign of any point between the roots t_1 and t_2. This is because any point in this range must have the same sign as the critical points of the trilinear interpolant for case 13.5.2. This can be seen in the black path shown in the rightmost image in Figure 4 and from the graph in Figure 6. All points between roots t_1 and t_2 will have the same sign, which is the sign of the "interior" of the tunnel. Thus, we compare the sign of $f((t_1 + t_2)/2)$ with the sign of both vertices of the voxel diagonal which is inside the tunnel. The tunnel will face the vertex with the same sign as $f((t_1 + t_2)/2)$, whereas the other vertex must be isolated from all cube vertices. Figure 10 illustrates this scenario. Note that Lopes and Brodlie [16] used the sign of the critical points of the trilinear interpolant to retrieve the correct tunnel orientation. We provide a different solution that fits nicely with Chernyaev's framework.

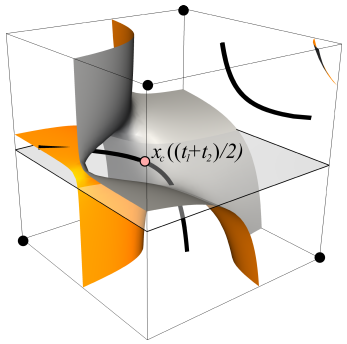


Figure 10: Solution to the orientation problem. The black dots represent regions with positive scalar values.The cutting-plane location is at $(t_1 + t_2)/2$. The sign of $f((t_1 + t_2)/2)$ determines the tunnel orientation.

6.2. Issue II – Non-manifold surfaces

A possible solution to this problem involves post-processing the mesh to remove non-manifold features. Although many works in the literature proposed methods for fixing meshes (see [11] for an excellent survey), these are mainly focused on retrieving a valid manifold mesh. Topologically correct algorithms, on the other hand, require that the topology of the trilinear interpolant be preserved. In addition, mesh repairing techniques may mask implementation issues by fixing them, which complicates the verification process.

We use an alternative approach that does not require any changes in the MC33 triangulation table. An interesting fact is that this problem has a low probability of being generated at random and an even lower probability of occurring in real-world datasets. For example, as shown in Figure 14 for the Skull dataset this problem appeared six times in total for 50 distinct isosurfaces. In our tests, it occurred only once in 10000 randomly generated $5 \times 5 \times 5$ scalar fields. Thus, instead of implementing the approach of Lopes and Brodlie's, we adopt a different solution that takes advantage of the fact that this is a rare event.

Non-manifold surfaces are created when two adjacent voxels that share an ambiguous face have tunnels in the voxel interior. By splitting both voxels at the critical point of that face, the face ambiguity is eliminated [1]. To simplify the algorithm, we split not only the voxels sharing the ambiguous face but

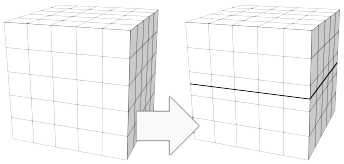


Figure 11: Grid refinement. The slice of voxels containing the offending configuration is splitted into two slices.

all faces in the volume slice that contains that face (see Figure 11). Assuming an input of size $n \times n \times n$, each subdivision will add n^2 voxels to the grid. Assuming that k subdivisions are required, kn^2 voxels will be added. In practice $k = O(1)$, and thus $kn^2 = O(1)O(n^2) = O(n^2)$. This implies that the asymptotic size of the dataset does not change. This subdivision adds the degree of freedom necessary to eliminate the problem, making this implementation of the Marching Cubes 33 topologically correct (see Figure 8).

6.3. Issue III – Cutting-plane computation

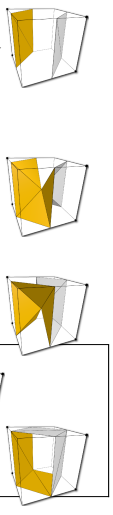
Because this is a problem with the alternative method used in Lewiner *et al.*, the issue can be avoided by replacing the use of t_{alt} with use of the originally proposed t_{max}.

6.4. Issue IV – Case 10

Algorithm 2 illustrates the required steps for disambiguation on case 10. We fixed the MC33 implementation by adding the lines 16-20, which in the original implementation were replaced by the result *case 10.1.1*.

Algorithm 2 Algorithm for case 10 [4]

```
1:  Positive nodes are denoted as n+
2:  if n+ are separated at top face then
3:      if n+ are separated at bottom face then
4:          if n+ at voxel diagonals are separated then
5:              Case 10.1.1
6:          else
7:              Case 10.1.2
8:          end if
9:      else
10:         Case 10.2
11:     end if
12: else
13:     if n+ are separated at bottom face then
14:         Case 10.2
15:     else
16:         if n+ at voxel diagonals joined then
17:             Case 10.1.1
18:         else
19:             Case 10.1.2
20:         end if
21:     end if
22: end if
```



7. Experiments with real-world datasets

We now turn our attention to the practical impact of the topological correctness of the trilinear interpolant. For real-world datasets, the vast majority of Marching Cubes cases match the non-ambiguous configurations, namely, 1, 2, 5, 8, and 9. This means that the standard Marching Cubes will match the topology generated by both MC33 and C-MC33. Nevertheless, for some voxels, there will be topological differences in the approaches, which may result in quite different meshes.

For the sake of completeness, in this section we provide a qualitative analysis of these differences. The aneurysm dataset shown in Figure 12 provides an example of the differences. From left to right, Figure 12 shows meshes extracted with VTK Marching Cubes, MC33, and C-MC33. The VTK implementation is based on the work of Montani *et al.* [19] and does not have topological guarantees aside from consistency. These three implementations can be viewed as three distinct ways of extracting the mesh topology. Although only a handful of voxels differ among the implementations, for the aneurysm dataset the consequence is that the (largest) main brain artery appears quite different in each interpretation. Because the dataset contains several thin features, subvoxel accuracy is required to connect the pieces of the blood vessels. As shown in the inset images in Figure 12, one voxel is sufficient to separate fairly large vessels.

VTK and MC33 generate more extra connected components (shown in purple) than does C-MC33. Figure 13 shows the difference in the number of connected components generated by VTK and C-MC33 (left) and by MC33 and C-MC33 (right) as a function of the isovalue for the aneurysm dataset. Clearly, VTK produces substantially more connected components than C-MC33 (up to 2400 more components). The differences between MC33 and C-MC33 are not as large, although they are sufficient to disconnect important artery segments. In this example, MC33 generates more connected components than C-MC33 for most isovalues. The aneurysm dataset shows that changes in the topology of some voxels can impact the final surface. In this particular example, it is reasonable to assume that the blood vessels form a single connected component and thus that the dataset contains as few connected components as possible. Using this criterion, C-MC33 shows the best performance for most isovalues. We emphasize that the "importance" of the differences in the number of connected components ought to be measured. For instance, although in general C-MC33 produced fewer connected components, for some isovalues the number of components extracted with C-MC33 was greater than the number extracted using MC33. As it turns out, this is due to the presence of pieces of small components disconnected from the main artery. However, because small isolated components do not disconnect large portions of the datasets, contrary to what is shown in Figure 12, MC33 and C-MC33 could be considered only "slightly" different. A thorough study of impact of the different approaches for extracting mesh topology is desirable but is beyond the scope of this work.

The second problem is due to the extraction of non-manifold features. The issue explained in Section 5.2 also pertains to

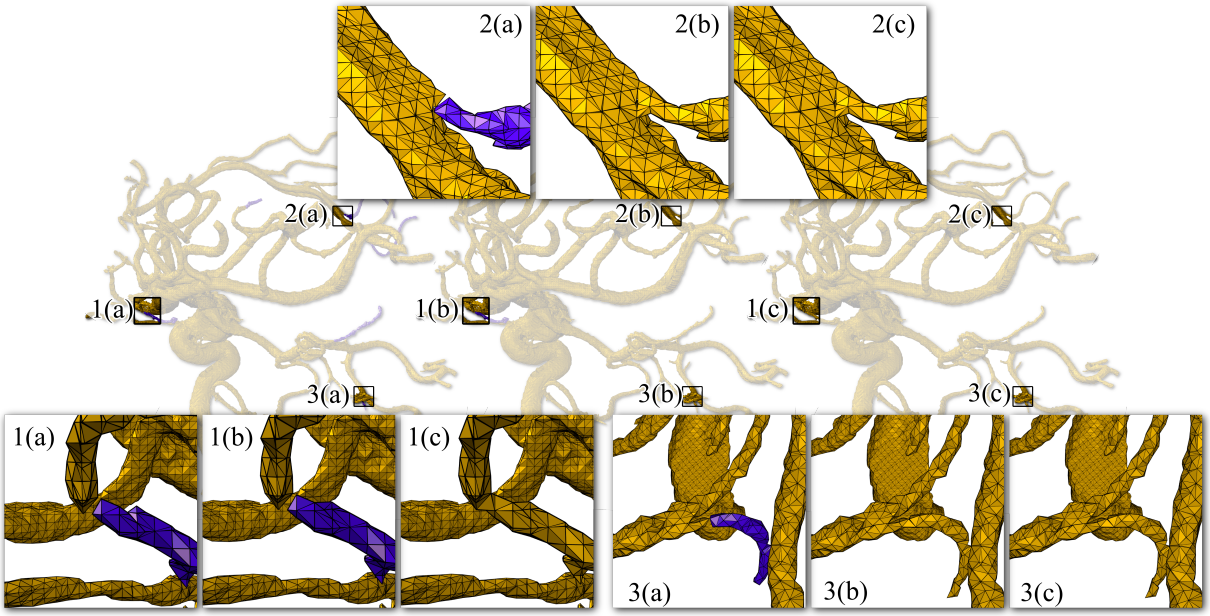


Figure 12: Aneurysm dataset. From left to right, the displayed isosurfaces were extracted using VTK, MC33, and C-MC33, respectively. We show the main brain artery component in yellow and the extra connected components in purple. From the images shown, it is clear that the purple components should be part of the main branch. Nevertheless, due to the implicit disambiguation in VTK and the issues in MC33, the final isosurface contains multiple components (left and middle figures). The isosurface generated using C-MC33 is shown on the right. [4]

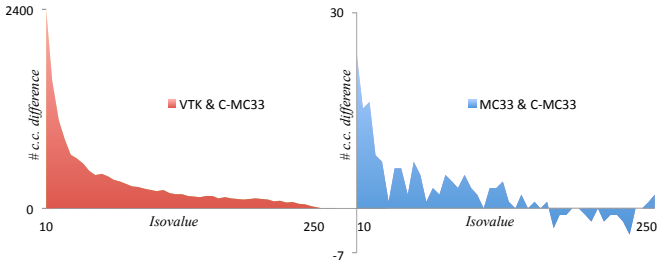


Figure 13: The left plot shows the difference between the number of connected components extracted by VTK implementation of Marching Cubes and the number of connected components extracted by our C-MC33 implementation. The right plot shows the difference in the number of connected components but between the MC33 and C-MC33 implementations. Negative values indicate that the C-MC33 implementation generated more connected components. Clearly, VTK generates more components that C-MC33. MC33 generates more components for most of the isovalues.

real-world datasets. Figure 14 shows an example of a medical dataset in which the output of MC33 implementation is a non-manifold surface. We have observed the same problem for certain isovalues of other commonly used datasets, such as the backpack and bonsai datasets. Nevertheless, in our experiments, this problem occurred rarely in the datasets tested: on average, one case of non-manifold edges was found per 10^7 evaluated voxels.

8. Conclusion

In this paper, we discuss in detail three issues with the Marching Cubes 33 algorithm and one non-trivial issue with its implementation. We present solutions for the issues raised and implement them into C-MC33, a topologically correct version of MC33. In addition, we make the results of our paper reproducible so that the reader can easily study, explore, and use the results presented here for his or her own purpose.

Acknowledgment

The authors wish to thank Lewiner *et al.* for making their code available and Fernando Chirigati, Wendel Silva, Tommy Ellqvist, Harish Doraiswamy and Prof. Juliana Freire for their valuable help with the paper. The medical datasets used in this work can be download from the Volvis project webpage. This work was supported in part by grants from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq # 202045/2012-9), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, (CAPES # 9864/11-3), the U.S. Department of Energy (DOE), Ofce of Biological and Environmental Research (BER), the National Science Foundation, and Exxon-Mobil.

Supplemental material

The results of this paper are reproducible. Information on the executable figures in this paper, the datasets, and source code can be accessed and downloaded from the project page [4].

References

[1] H. Carr and N. Max. Subdivision analysis of the trilinear interpolant. *IEEE Transactions on Visualization and Computer Graphics*, 16:533–547, 2010.

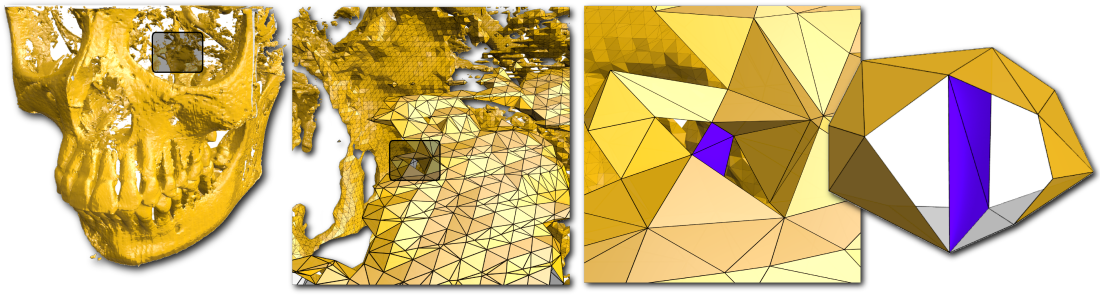


Figure 14: Skull dataset. The image shows a progressive zoom-in into the dataset in order to reveal non-manifold edges. The face containing the non-manifold edges is highlighted in purple. The rightmost image is an isolated version of the case shown in the dataset, with a slightly different geometry for the sake of clarity. A non-manifold edge appeared six times in total for 50 distinct isosurfaces.

[2] E. V. Chernyaev. Marching Cubes 33: Construction of topologically correct isosurfaces. Technical Report CN/95-17, Institute for High Energy Physics, 1995.

[3] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computers & Graphics*, 24(3):399 – 418, 2000.

[4] L. Custodio, T. Etiene, S. Pesco, and C. Silva. Practical considerations on Marching Cubes 33. `http://liscustodio.github.io/C_MC33`, April 2013.

[5] T. K. Dey and J. A. Levine. Delaunay meshing of isosurfaces. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, SMI '07, pages 241–250, Washington, DC, USA, 2007. IEEE Computer Society.

[6] M. Dürst. Letters: additional reference to marching cubes. *Computer Graphics*, 1988.

[7] T. Etiene, L. G. Nonato, C. Scheidegger, J. Tierny, T. J. Peters, V. Pascucci, R. M. Kirby, and C. T. Silva. Topology verification for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics*, 18:952–965, 2012.

[8] J. Freire, C. T, Silva, S. Callahan, E. Santos, C. E, Scheidegger, and H. T, Vo. Managing rapidly-evolving scientific workflows. In *Proceedings of the International Conference on Provenance and Annotation of Data*, pages 10–18, 2006.

[9] A. V. Gelder and J. Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13:337–375, 1994.

[10] E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14:95–108, 1998.

[11] T. Ju. Fixing geometric errors on polygonal models: A survey. *Journal of Computer Science and Technology*, 24:19–29, 2009. 10.1007/s11390-009-9206-7.

[12] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Trans. Graph.*, 21(3):339–346, July 2002.

[13] D. Koop, E. Santos, P. Mates, H. T. Vo, P. Bonnet, B. Bauer, B. Surer, M. Troyer, D. N, Williams, J. E, Tohline, J. Freire, and C. T, Silva. A provenance-based infrastructure to support the life cycle of executable papers. *Procedia Computer Science*, 2011.

[14] T. Lewiner. *http://www.matmidia.mat.puc-rio.br/tomlew*, 2012 (accessed July 24, 2012).

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

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

[17] W. Lorensen and H. Cline. Marching Cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Computer Graphics*, 21:163–169, 1987.

[18] S. V. Matveyev. Marching cubes: surface complexity measure. Technical report, Institute for High Energy Physics, 1999.

[19] C. Montani, R. Scateni, and R. Scopigno. A modified look-up table for implicit disambiguation of marching cubes. *The Visual Computer: International Journal of Computer Graphics*, 1994.

[20] B. K. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11(1):52–62, Jan. 1994.

[21] T. S. Newman and H. Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854 – 879, 2006.

[22] G. M. Nielson. On marching cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9:283–297, 2003.

[23] G. M. Nielson and B. Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd conference on Visualization*, pages 83–91, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

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

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

[26] J. Schreiner, C. E. Scheidegger, S. Fleishman, and C. T. Silva. Direct (re)meshing for efficient surface processing. *Computer Graphics Forum*, 25(3):527–536, 2006.

[27] W. Schroeder, K. Martin, and W. Lorensen. *Visualization Toolkit, An Object-Oriented Approach to 3D Graphics - 2nd ed.* Prentice-Hall, 1998.

[28] C. T. Silva, J. Freire, and S. P. Callahan. Provenance for visualizations: Reproducibility and beyond. *Computing in Science and Engineering.*, 9(5):82–89, Sept. 2007.

[29] C. T. Silva and J. S. B. Mitchell. Greedy cuts: an advancing front terrain triangulation algorithm. In *Proceedings of the 6th ACM international symposium on Advances in geographic information systems*, GIS '98, pages 137–144, New York, NY, USA, 1998. ACM.

[30] H. Theisel. Exact isosurfaces for marching cubes. *Computer Graphics Forum*, 21(1):19–32, 2002.

[31] J. E. Tohline and E. Santos. Visualizing a Journal that Serves the Computational Sciences Community. *Computing in Science & Engineering*, 12(3):78–81, 2010.

[32] G. Varadhan, S. Krishnan, Y. J. Kim, and D. Manocha. Feature-sensitive subdivision and isosurface reconstruction. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 14–, Washington, DC, USA, 2003. IEEE Computer Society.

[33] N. Zhang, W. Hong, and A. Kaufman. Dual contouring with topology-preserving simplification using enhanced cell representation. In *Proceedings of the conference on Visualization '04*, VIS '04, pages 505–512, Washington, DC, USA, 2004. IEEE Computer Society.

Appendix A. The counterexample in numbers

In this section we provide the data necessary for reproducing the counterexample shown in Figure 6. The isosurface of interest is homeomorphic to configuration 13.5.2 of the extended Marching Cubes table. This example can be used to show that both the original and modified versions of the MC33 algorithm will fail to retrieve the correct case. Following the interior test proposed by Chernyaev, let

$$\begin{aligned} A_0 &= +0.2864 \qquad A_1 = -0.2384 \\ B_0 &= -0.0639 \qquad B_1 = +0.9486 \\ C_0 &= +0.6568 \qquad C_1 = -0.5049 \\ D_0 &= -0.1692 \qquad D_1 = +0.1075. \end{aligned}$$

The coefficient a, b, and c in $F(t)$ are given by

$$\begin{aligned} a &= +(A_1 - A_0)(C_1 - C_0) \\ &\quad -(B_1 - B_0)(D_1 - D_0) = 0.3296 \\ b &= +C_0(A_1 - A_0) \\ &\quad +A_0(C_1 - C_0) \\ &\quad -D_0(B_1 - B_0 \\ &\quad -B_0(D_1 - D_0) = -0.4886 \\ c &= A_0C_0 - B_0D_0 = 0.1701. \end{aligned}$$

Condition (i) does not hold because $a > 0$, which means that a tunnel is absent. Therefore, under Chernyaev's conditions, case 13.5.2 is incorrectly interpreted as 13.5.1.

Now, following the Lewiner's implementation, for the same scalar field, let

$$\begin{aligned} A_0 &= +0.1075 \qquad A_1 = -0.5049 \\ B_0 &= -0.1692 \qquad B_1 = +0.6568 \\ C_0 &= +0.2864 \qquad C_1 = -0.0639 \\ D_0 &= -0.2384 \qquad D_1 = +0.9486. \end{aligned}$$

The proposed alternative t is given by

$$t_{\text{alt}} = \frac{A_0}{A_0 - A_1} = 0.1756,$$

and:

$$F(t_{\text{alt}}) = -0.0007 < 0.$$

Thus condition (iii) fails, which means that case 13.5.2 is again incorrectly interpreted as 13.5.1.