Efficient implementation of Marching Cubes' cases with topological guarantees

Thomas Lewiner[1,2], Hélio Lopes[1], Antônio Wilson Vieira[1,3] and Geovan Tavares[1]

[1] Department of Mathematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
[2] Géométrica Project — INRIA – Sophia Antipolis — France
[3] CCET — Universidade de Montes Claros — Brazil
{tomlew, lopes, awilson, tavares}@mat.puc--rio.br.

Abstract. Marching Cubes' methods first offered visual access to experimental and theoretical data. The implementation of this method usually relies on a small lookup table. Many enhancements and optimizations of Marching Cubes still use it. However, this lookup table can lead to cracks and inconsistent topology. This paper introduces a full implementation of Chernyaev's technique to ensure a topologically correct result, i.e. a manifold mesh, for any input data. It completes the original paper for the ambiguity resolution and for the feasibility of the implementation. Moreover, the cube interpolation provided here can be used in a wider range of methods. The source code is available online.

Keywords: *Marching Cubes. Isosurface extraction. Implicit surface tiler. Topological guarantees.*
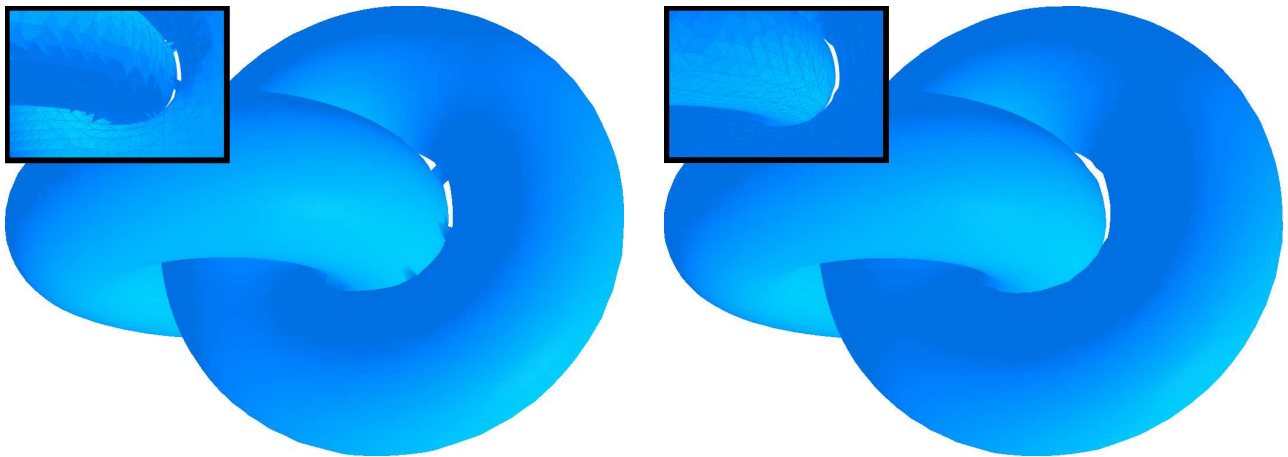
Figure 1: *Implicit surface of linked tori generated by the classical Marching Cubes algorithm, and ours.*

1 Introduction

Isosurface extractors and implicit surface tilers opened up visual access to experimental and theoretical data, such as medical images, mechanical pieces, sculpture scans, mathematical surfaces, and physical simulation by finite elements methods. Among those techniques, the Marching Cubes [5] produces a surface out of a sampling of a scalar field $f : \mathbb{R}^3 \rightarrow \mathbb{R}$. It has been enhanced to a wide range of applications, from geological reconstruction [10], medical images to 3D scanning (see [4] for an original use in the Digital Michelangelo Project). Although this paper focuses on surface reconstruction from sampled data, the tilings of cubes introduced here can be used in simple reconstruction meth-

ods for synthetic data [2, 13] in order to guarantee the topological consistency of the result when the precision of the result is limited.

Marching Cubes [5] has become the reference method when the sampled scalar field is structured on a cuberille grid. It classifies vertices as positive or negative, according to their comparison with a given isovalue. Then, it uses a lookup table to tile the surface inside the cube. This method has been enhanced and generalized in various directions, especially to reduce the number of cubes to be evaluated. However, most of those modern techniques still use a simple lookup table, which does not ensure the topological consistency of the result.

Prior work. The main obstacles of the Marching Cubes' derived methods are the ambiguities inherent to data sampling. Those ambiguities can appear on the faces of a cube,
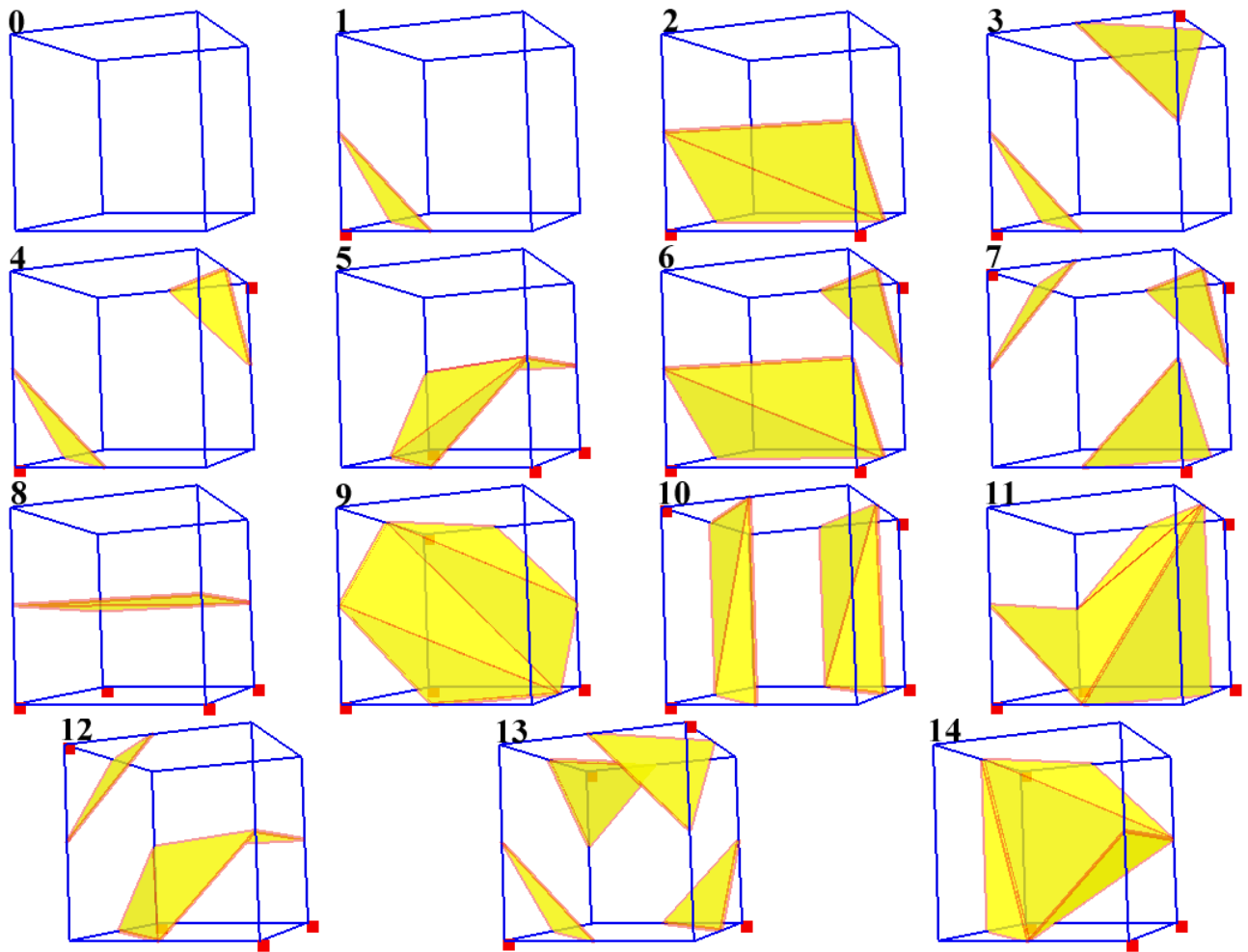
Figure 2: *Original Marching Cubes' lookup table.*

or inside the cube. The ambiguities on faces have been re-solved in [8], supposing the scalar field f is trilinear over each cube, which gave a modified lookup table [6]. Within the same hypothesis, it is possible to resolve internal ambigu-ity as done in [7, 3]. Further approaches [12, 1, 14] computes the topology of f as a volume, giving rise to more complex algorithms that need, at the end, to tile each cube.

Contribution. In this paper, we describe an efficient and robust implementation of Chernyaev's Marching Cubes 33 algorithm [3] (see Figure 1). We needed to complete Chernyaev's paper on the internal ambiguity resolution. We computed and tested the 730 subcases of the enhanced lookup table. This table can be used *as is* into Marching Cubes' improvements (in particular, those who avoid empty cell tests). Our result is guaranteed to be a manifold surface, with no crack, with the topology of the trilinear interpola-tion of the scalar field over each cube. The complete source code is available online at the address listed at the end of this paper.

Cube vs. tetrahedron. Another range of techniques for iso-surface generation is based on tetrahedra [11], as opposed to cubes. Those methods guarantee the topological consis-tency, and have a small lookup table. However, they have many drawbacks. They generate much more triangles, with a weaker geometrical accuracy of the result: the cubes' tilings are segmented even in obvious configuration, and the vertex position cannot be adjusted to fit the geometrical trilinear ap-proximation as we do with cubes. Moreover, the ambiguity resolution that is hidden in those methods leads to slower algorithms, which are more difficult to speed up with hard-ware implementation. Our technique uses a complex lookup table, that only needs to be stored, which enable our algo-rithm to be efficiently hardware accelerated. This technique allows a **topologically correct** result with a single entry cu-bical lookup table, by a low complexity algorithm. This is a significant practical improvement compared to the former state of the art of isosurface tilers [9].

2 Marching Cubes with topological guarantees

Marching Cubes. The Marching Cubes method produces a triangle mesh of the preimage $f^{-1}(\alpha)$ of an isovalue α by a scalar function $f : \mathbb{R}^3 \to \mathbb{R}$. We will consider $\alpha = 0$ for the rest of the paper (considering $f - \alpha$). This scalar field is given by samples over a cuberille grid. The original method sweeps the grid, and tiles the surface cube per cube. Each vertex v of the cube is classified into positive and negative vertices, depending whether $f(v)$ is greater than α or not. Thus, there are $2^8 = 256$ possible configurations of a cube. The usual implementation stores those 256 in a lookup table that encodes the tiling of the cube in each case (see Figure 2).

Correct topology. However, this simple algorithm can leads to cracks, as shown on Figure 3. The same configuration can be tiled in various ways, and the 256– entries lookup table does not distinguish between those. Among the different tilings, some approximate a trilinear interpolation of the scalar field f over the cube. We will say a resulting mesh has the *correct topology* if it is homeomorphic to $F^{-1}(\alpha)$, where F is equal to f at the sample vertices, and trilinear over each cube of the grid. This allows avoiding cracks, by applying topological test on ambiguous faces of a cube. The same test will be done on the adjacent cube, allowing a coherent transition from one cube to the other one. Nielson and Hamann [8] introduced the usual face test to resolve those face ambiguities.
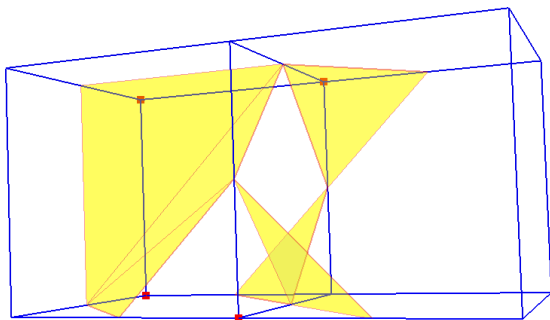
Figure 3: *A crack occurring on an ambiguous face in-between cases 12 and 3 with the 256–lookup table.*

By resolving face ambiguities, we avoid cracks. Nevertheless, this does not guarantee the correct topology, as with the same cube configuration and the same resolution of ambiguous faces, there are topologically different trilinear interpolations (see Figure 4). Therefore, we also need to resolve internal ambiguity to guarantee the topology. The technique described in this paper guarantees the topology by providing an extended lookup table and an enhanced analysis of each cube.

Marching Cubes 33. Chernyaev described, with the Marching Cubes 33 [3], the different possible topologies of a trilinear function over a cube. He gave a tiling for each case, adding some extra points for better geometrical approximation if necessary. He also proposed a method for
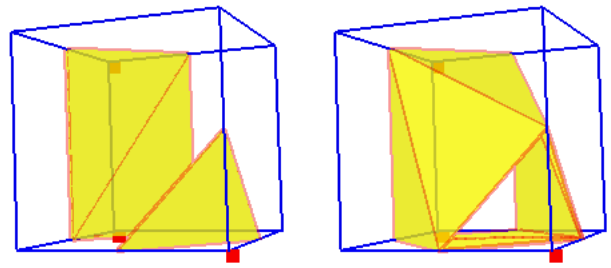
Figure 4: *Two trilinear tilings of the 6th case, with the same resolution of faces' ambiguity.*

resolving internal ambiguity, although it was not complete. We completed and enhanced this method, adding some tricks to avoid useless tests. We computed and tested the complete lookup table described by Chernyaev (see Figure 5).

3 Algorithm and implementation

The algorithm goes through the following 4 steps:

1. determine the case number and configuration (section 3(a) *Determining the configuration*).

2. lookup which faces are to be tested for this configuration (section 3(b) *Performing the tests*).

3. determine the subcase based on the result of the face tests (section 3(c) *Determining the subcase*).

4. lookup the tiling of the cube for this subcase (section 3(d) *Tiling each cube*).

noindent The simplicity of the algorithm relies on the lookup table, which is actually split into three tables:

— The *case table* maps each of the 256 possible configurations of a cube to one of the 15 cases of Figure 2, and to a specific number designating this configuration.

— The *test table* stores, for each configuration, the tests to be performed to resolve topological ambiguity.

— The *tiling table* stores the tiling for each configuration and subcase (there is no need for computing any geometrical transformation).

(a) Determining the configuration

The classical Marching Cubes lookup table has 256 entries (represented on Figure 2 by the 15 geometrically different cases, i.e. that cannot be deduced by solid transformations.). Each entry represents a different vertex configuration, given that a vertex is simply described by its sign (positive or negative).

Each entry is identified by an 8–bit word b, as for the classical Marching Cubes: its ith bit is set to 1 (resp. 0) if the ith vertex of the cube is positive (resp. negative). Figure 6 details the label of the vertices. The *case table* maps this 8-bit word b to the corresponding configuration of one of the 15 cases (see Figure 2). The configuration numbering is arbitrary.
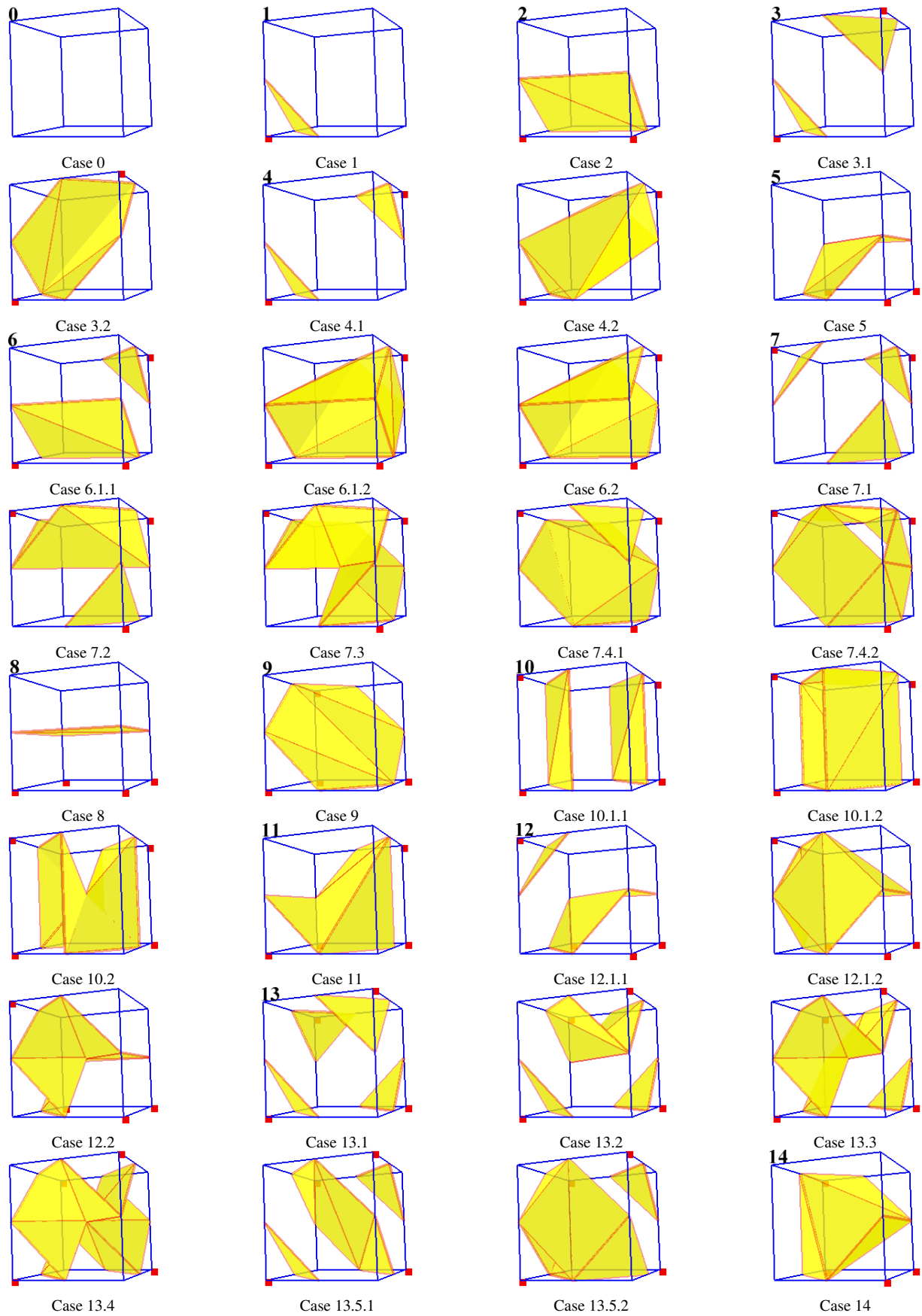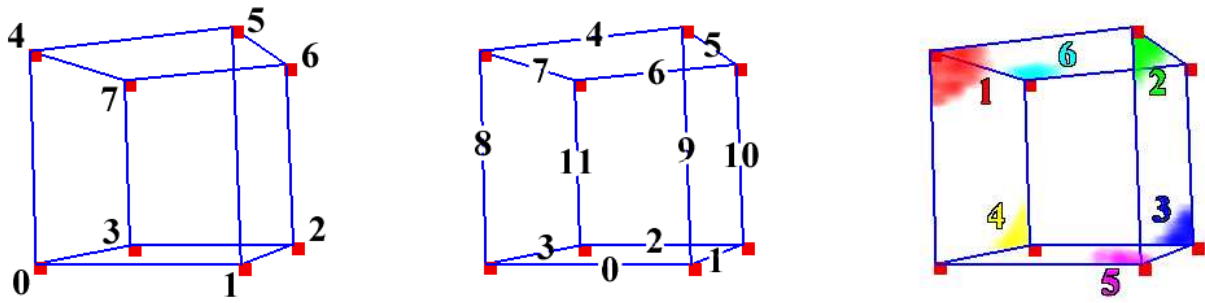
**Figure 5:** *Chernyaev's lookup table.*

Figure 6: *Labeling of vertices, edges and faces: vertex 0 has the lowest x,y,z coordinates, and vertex 6 the highest.*

For example, $b = 129$ means that only vertices 0 and 7 are positive, which is numbered in the case table as the 3rd configuration of the case 3 (see Figure 7).

(b) Performing the tests

To determine the topology of the graph of f inside each cube, it is not sufficient to know the sign of f over each vertex, even when f is trilinear. For example, Figure 7 shows two trilinear tilings of a cube with the same sign of the vertices. When two or more adjacent vertices of a face have the same sign, the topology of the isosurface on that face is obvious. Otherwise, this ambiguity can be resolved using the tests described in section 4(a) *Resolution of faces ambiguities*. When the positive vertices of a cube are connected by edges or through the faces, and also are the negative vertices, the topology of the interior of the cube is obvious. Otherwise, this ambiguity can be resolved using the method described in section 4(b) *Resolution of internal ambiguities*.

To know which test are necessary to perform, a *test table* stores for each configuration of each ambiguous case (cases 3,4,6,7,10,12 and 13) the label of the faces to be tested. The interior test sometimes requires an edge code further describing the configuration, which are stored at specific position in the tiling table (see section 4(b) *Resolution of internal ambiguities*). The labels of the edges and faces are detailed on Figure 6, face 7 stands for the interior.
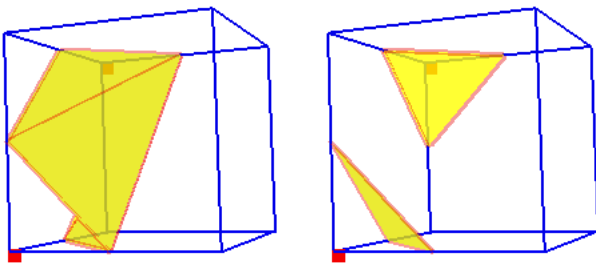
Figure 7: *Case 3, configuration 3: two different tilings of the ambiguous face 4.*

For example, with $b = 129$, the 3rd entry of the *test table* of case 3 is just '4', which means that face 4 is ambiguous (there is no adjacent vertices with the same sign) and has to be tested (see Figure 7).

(c) Determining the subcase

The branching of those tests determines the subcase to be tiled (represented on Figure 5 by the 32 geometrically different cases). This branching is hard–coded for the simple cases (see table 1). For complex cases (cases 7 and 13), other small tables are used to map the result of the tests to their corresponding subcase.

For example, with $b = 129$, the subcases of case 3 are determined by only one test, and is thus hard–coded. The cube will be tiled according to subcase 3.1 if the test is negative, and to subcase 3.2 otherwise (see Figure 5 and Figure 7).

(d) Tiling each cube

Each cube is tiled with triangles according to the *tiling table*. Each code of the tiling sequences identifies an edge of the cube (see Figure 6) on which a vertex of the cube's tiling will be computed. The 12th code means interior vertex. The vertices of the final triangulation are computed by barycentric interpolations on the edges of the cube. Each group of 3 consecutive edge codes in the tiling table corresponds to a triangle.

For example, with $b = 129$, the 3rd configuration of case 3 corresponds in the tiling table to

```
{ 3,0,8,11,7,6,
            7,8,6,8,0,6,3,11,6,3,6,0 }
```

Subcase 3.1 corresponds to the first sequence of the tiling table (2 triangles), and subcase 3.2 to the second one (4 triangles), as shown on Figure 7. As each tiling of the same subcase has the same number of triangles, those sequences are easily distinguished even if stored in the same table.

4 Ambiguity resolution

This section describes the methods used to test the faces and interior of a cube, when the values at the vertices yield an ambiguous configuration. Those tests are used in step 2 of the algorithm, to resolve those ambiguities on a face (see section 4(a) *Resolution of faces ambiguities*) or on the interior of a cube (see section 4(b) *Resolution of internal ambiguities*).

Case	Face tests			Interior Test	Subcase	# triangles
	1st	2nd	3rd			
0						0
1						1
2						2
3	-				3.1	2
	+				3.2	4
4	-				4.1	2
	+				4.2	6
5						3
6	-			-	6.1.1	3
	-			+	6.1.2	7
	+				6.2	5
7	-	-	-		7.1	3
	+	-	-		7.2	5
	-	+	-		7.2	5
	-	-	+		7.2	5
	+	+	-		7.3	9
	+	-	+		7.3	9
	-	+	+		7.3	9
	+	+	+	+	7.4.1	9
	+	+	+	-	7.4.2	9
8						2
9						4
10	+	+			10.1.1	4
	-	-		-	10.1.1	4
	-	-		+	10.1.2	8
	+	-			10.2	8
	-	+			10.2	8
11						4
12	+	+			12.1.1	4
	-	-		-	12.1.1	4
	-	-		+	12.1.2	8
	+	-			12.2	8
	-	+			12.2	8
13	45 subcases, testing all the 6 faces and eventually the interior					
14						4

Table 1: *A reduced representation of the* test table*. Case 13 has 45 entries to map the results of all the possible tests to the right subcase.*
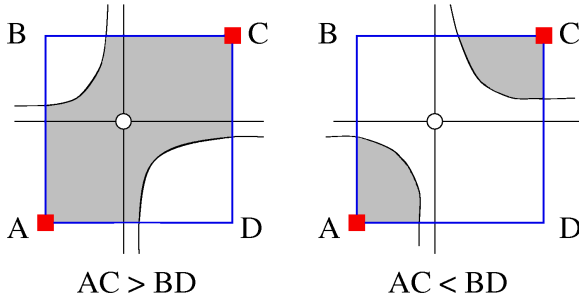
$$AC > BD \qquad\qquad AC < BD$$

Figure 8: *Face ambiguity test (Chernyaev's illustration [3]).*

(a) Resolution of faces ambiguities

Face ambiguity arise when two opposite vertices A and C of the face are positive, and the two others B and D are negatives (see Figure 7 and Figure 8). We supposed the scalar field F is trilinear on each cube, thus, bilinear on each face. Therefore, $F^{-1}(\alpha)$ restricted to the face is a hyperbola (see Figure 8). Testing whether the center of the hyperbola is positive or negative (i.e. whether the positive vertices are connected inside the face) reduces (for $\alpha = 0$) to testing the sign of $F(A) \cdot F(C) - F(B) \cdot F(D)$.

The sign of the center of the hyperbola is the sign of the above expression if A is positive, and the opposite if A is negative. In addition, for some configuration, we want the opposite of the result. This is encoded by a negative face label in the test table. Therefore, the implemented test returns the sign of: $sign\,(face\_label \cdot F(A) \cdot (F(A) \cdot F(C) - F(B) \cdot F(D)))$.

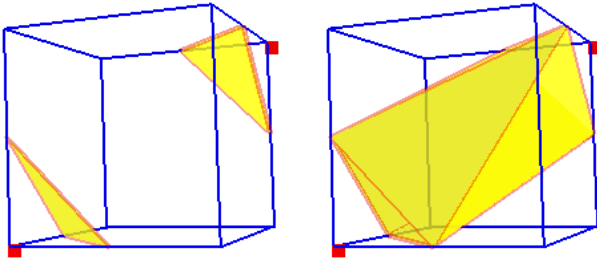### (b) Resolution of internal ambiguities

Figure 9: *Case 4, configuration 0: two different tilings of the interior of a cube.*

An internal ambiguity arises when two diagonally opposite vertices A_0 and C_1 of a cube can be connected through the interior of the cube, creating a kind of tunnel (see Figure 4 and Figure 9). Let say those two vertices A_0 and C_1 are positive (the description holds for negative vertices also). We first resolve face ambiguity, according to section 4(a) *Resolution of faces ambiguities*. If there is a chain of positive vertices joining A_0 to C_1, connected by edges or crossing ambiguous faces resolved as positive, then there is no internal ambigu-

ity. Otherwise, we have to test if A_0 and C_1 are connected only through the cube.

Suppose A_0 and C_1 are connected through the interior of the cube. For $X = A, B, C, D$, let $X_t = t \cdot X_0 + (1-t) \cdot X_1$. As F is trilinear, F cannot change sign more than once along segment. Thus, there is a plane $P = (A_t, B_t, C_t, D_t)$, where the two edges A_0A_1 and C_1C_0 are connected inside the square A_t, B_t, C_t, D_t (see Figure 10). For each case, we must find the direction of P, compute the height t of P along the edge, and resolve the ambiguity inside the square A_t, B_t, C_t, D_t.
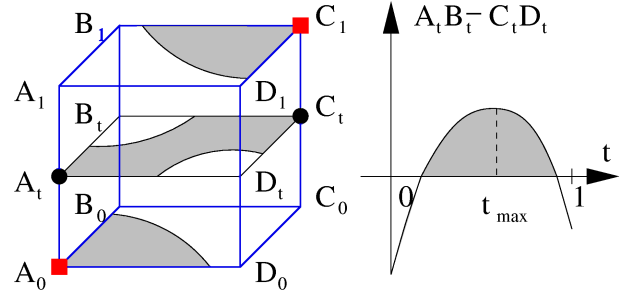

Figure 10: *Internal ambiguity test for cases 4 and 10 (Chernyaev's illustration [3]).*

Cases 4 and 10 have enough symmetry to choose always the same direction, independently of the configuration. The height $t \in [0, 1]$ is the solution of Chernyaev's second order equation : $a \cdot t^2 + b \cdot t + c$, with :

$$
\begin{aligned}
a &= (A_1 - A_0) \cdot (C_1 - C_0) - (B_1 - B_0) \cdot (D_1 - D_0) \\
b &= C_0 \cdot (A_1 - A_0) + A_0 \cdot (C_1 - C_0) \\
&\quad - D_0 \cdot (B_1 - B_0) - B_0 \cdot (D_1 - D_0) \\
c &= A_0 \cdot C_0 - B_0 \cdot D_0
\end{aligned}
$$

For the other cases, the direction of P is encoded as an edge e inside one particular sequence of the tiling table: the 17th edge for cases 6, 7, and 12; the 2nd edge for the case 13.5. In that case, the height of the plane is the barycenter of the end vertices of e, weighted by F.

In both situations, the intersection of P with the cube is a square, with vertices A_t, B_t, C_t, D_t. There is no ambiguity inside this square if 0 or 1 vertex is positive (negative vertices are connected through the cube), if 3 or 4 vertices are positive (positive vertices are connected through the cube), or if 2 consecutive vertices are positive (no diagonal connection). In the other case, the square is ambiguous, and we resolve it in the same way as for face ambiguity.

5 Tips and tricks

The vertices of the final mesh are interpolated along an edge. To avoid computing them more than once, they can be all computed first. To store them, we used 3 arrays, which assign respectively to each grid vertex an eventual index to the mesh vertex on the edge parallel to the x, y and z axis.

The normal coordinates at each grid vertex $\overrightarrow{p}$ can be computed as $F\left(\overrightarrow{p} + \overrightarrow{\delta}\right) - F\left(\overrightarrow{p} - \overrightarrow{\delta}\right)$, where $\overrightarrow{\delta}$ is the grid step along that coordinate. The normal at each point is then interpolated linearly.

A low resolution extraction can be obtained by considering a lower resolution grid, i.e. taking into account every other vertex or every n–th vertex.

This algorithm is guaranteed to produce manifold meshes for any sample data, which allows working on previews with the same tools as on the final mesh.

This algorithm can be used for implicit surface tiling to construct fixed precision or exact result. In the latter case, it allows an economic use of exact arithmetic: when an evaluation of the surface inside a cube is ambiguous, the cube needs to be subdivided and the implicit function is evaluated again on the subdivision cubes. The tests we provided here can be substituted to avoid subdividing cubes guaranteeing a manifold result. For example, an exact evaluation of the contour graph of F [1] gives the number of connected components inside a cube. This topological information is a powerful test to distinguish between subcases.

Web information

A C++ implementation together with the tables are available online at `http://www.acm.org/jgt/papers/ LewinerEtAl03`. A small interface allows examining each entry of those tables.

References

[1] C. L. Bajaj, V. Pascucci and D. Schikore. Visualization of scalar topology for structural enhancement. In D. Ebert, H. Hagen and H. Rushmeier, editors, *Visualization*, pages 51–58. IEEE, 1998.

[2] J. Bloomenthal. An implicit surface polygonizer. In P. Heckbert, editor, *Graphics Gems IV*, pages 324–349. Academic Press, Boston, 1994.

[3] E. V. Chernyaev. Marching Cubes 33: construction of topologically correct isosurfaces. Technical Report CERN CN 95–17, CERN, 1995.

[4] J. Davis, S. Marschner, M. Garr and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Symposium on 3D Data Processing, Visualization, and Transmission*, 2002.

[5] W. E. Lorensen and H. E. Cline. Marching Cubes: a high resolution 3D surface construction algorithm. In *Siggraph*, volume 21, pages 163–169. ACM, 1987.

[6] C. Montani, R. Scateni and R. Scopigno. A modified lookup table for implicit disambiguation of Marching Cubes. *The Visual Computer*, 10(6):353–355, 1994.

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

[8] G. M. Nielson and B. Hamann. The asymptotic decider: resolving the ambiguity in Marching Cubes. *Visualization*, pages 29–38, 1991.

[9] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *Computer Graphics and Applications*, 13(6):33–41, 1993.

[10] G. Tavares, R. Santos, H. Lopes, T. Lewiner and A. W. Vieira. Topological reconstruction of oil reservoirs from seismic surfaces. In *International Association for Mathematical Geology*, Portsmouth, UK, 2003. Terra Nostra.

[11] G. Treece, R. Prager and A. Gee. Regularised Marching Tetrahedra: improved iso–surface extraction. *Computers & Graphics*, 23(4):583–598, 1999.

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

[13] L. Velho. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools*, 1(2):5–25, 1996.

[14] G. H. Weber, G. Scheuermann, H. Hagen and B. Hamann. Exploring scalar fields using critical isovalues. In *Visualization*. IEEE, 2002.