Robust adaptive meshes for implicit surfaces

Afonso Paiva Hélio Lopes Thomas Lewiner Luiz Henrique de Figueiredo
Department of Mathematics, PUC–Rio, Brazil IMPA, Rio de Janeiro, Brazil

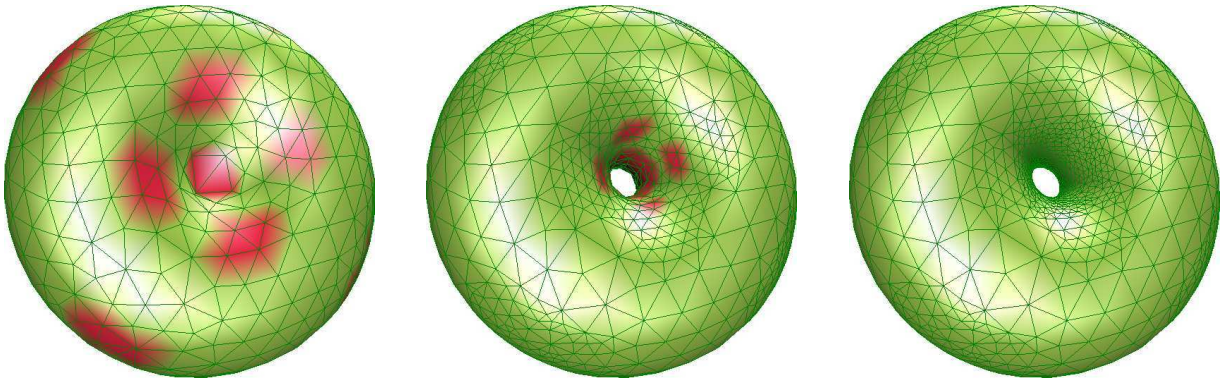

Figure 1. Toric isosurface extraction: our algorithm extracts a valid surface with adaptive triangulation. It guarantees the green parts of the surface, and the ambiguity of the other parts is resolved with a small number of refinements.

Abstract

This work introduces a robust algorithm for computing good polygonal approximations of implicit surfaces, where robustness entails recovering the exact topology of the implicit surface. Furthermore, the approximate triangle mesh adapts to the geometry and to the topology of the real implicit surface. This method generates an octree subdivided according to the interval evaluation of the implicit function in order to guarantee the robustness, and to the interval automatic differentiation in order to adapt the octree to the geometry of the implicit surface. The triangle mesh is then generated from that octree through an enhanced dual marching.

Keywords: *Implicit Surface, Dual Marching Cubes, Robust Algorithms, Geometric Modelling.*

1 Introduction

Implicit surfaces provide powerful primitives for geometric modelling. However, computing good polygonal approximations remains an important problem. An *implicit surface* is the set of solutions of an equation $f(x, y, z) = 0$, where $f\colon \Omega \subseteq \mathbf{R}^3 \to \mathbf{R}$. For well-behaved functions f, this set is indeed a manifold surface.

The simplest and most flexible polygonal approximation abides triangle meshes, since they are easy to represent efficiently and they suit also well for rendering with current graphics hardware. The criteria for good approximations involve robustness and adaptation. Robustness means that the mesh captures exactly the topology of the surface, guaranteeing the representation of each connected component in Ω and their genus. Adaptation means that, with a reduced number of triangles, the geometry of the surface is described efficiently. In particular, the mesh should place large triangles in regions of low surface curvature and smaller triangles in regions of high surface curvature. Moreover, the triangles should have a good aspect ratio since thin triangles (also known as *slivers*) induce numerical instability for geometric processing, in particular for rendering and derivative estimations.

In this paper, we describe an algorithm that computes a robust and adaptive triangular approximation for an implicit surface given by a formula. The algorithm combines interval arithmetic and automatic differentiation to ensure both robustness and adaptation. The algorithm first explores the domain Ω adaptively using an octree to locate the sur-

face. This exploration uses interval arithmetic to eliminate octree cells that are guaranteed *not* to intersect the surface, driving the octree towards identifying all connected components. Combined with automatic differentiation, interval arithmetic provides interval estimates for the gradient, which allows locating regions of low curvature and detecting topological ambiguities. A triangle mesh is extracted from the dual of the octree [21], and then smoothed to avoid slivers. This technique produces a good triangle mesh that carries topological and geometric guarantees. Figure 1 shows an example of the approximation of the implicit surface

$$(1.5 - \sqrt{(x^2 + y^2)})^2 + z^2 - (1.35)^2 = 0,$$

running our method in the cube $\Omega = [-3.1, 3.1]^3$. Further examples are given in Section 5.

This paper extends to surfaces our previous research on implicit curves [15]. Section 2 reviews other related works. A brief account of interval arithmetic and automatic differentiation is given in Section 3, followed by the details of the complete algorithm in Section 4.

2 Related work

There exist mainly two types of data for defining implicit surfaces: discrete or continuous. Discrete data usually results from measurements or numerical simulations, which increase the interest for visualizing such data in medical and scientific applications. Discrete data contains information only at the vertices of a grid and requires interpolation schemes for the other points. In contrast, continuous data can be evaluated at arbitrary points of Ω. Implicit surfaces given by continuous data are especially relevant for geometric modelling for at least two reasons: they provide powerful primitives for modelling, and express naturally many geometric operations such as deformations, interpolations and derivation mechanisms. Classical methods for the continuous case sample the function at the vertices of a regular grid and apply algorithms for the discrete case. However, the knowledge of the continuous function may help in generating adapted grids.

Extensions of Marching Cubes. Probably the earliest and most influential work on approximating implicit surfaces from discrete data remains the Marching Cubes algorithm by Lorensen and Cline [16]. Marching Cubes works on a cubical grid and identifies cells that intersect the surface by testing the signs of the values at the eight vertices of each cube. If the signs are not all the same, then the surface is approximated within the cell by planar patches: the signs of the eight vertices form an 8-bit word which serves as a *key* for the configuration of the cube, and a lookup table stores the patches for each possible configuration.

The main issues in this approach are ensuring consistency of the approximation across neighbouring cells and resolving topological ambiguities. Many papers have been written on improving Marching Cubes along these lines, and early papers are discussed in a survey by Ning and Bloomenthal [19]. In this paper, we use a recent implementation of Marching Cubes by Lewiner et al. [9] that introduced topological guarantees.

Adaptive strategies. The earliest papers that perform adaptive approximation of implicit surfaces are due to Bloomenthal [2], who used cubical cells, and to Hall and Warren [6], who used tetrahedral cells. Both papers took care to avoid cracks on the approximating mesh. Hall and Warren also tried to eliminate slivers by projecting mesh vertices that are near the surface. Several other authors discuss adaptive subdivision and refinement [22, 24, 1, 5]. Hartmann [7] described a continuation algorithm that produces a regular mesh, extended by Karkanis and Stewart [14] and Araujo and Jorge [4] to adaptive meshes. Quite recently, Ho et al. [8] extended Marching Cubes to generate adaptive approximations that preserve sharp features and provide topological consistency. We used a different approach to ensure crack-free approximations by the use of dual grids, pioneered by Ju et al. [12], Schaefer and Warren [21], and Nielson [18].

Robustness. Very few methods guarantee robustness in the continuous case. Stander and Hart [23] used interval methods to guarantee the correct topology of polygonal approximations during interactive modelling. Their use of Morse theory is further developed for implicit surfaces in the work of Boissonnat et al. [3]. Lopes et al. described a robust method for implicit curves [15], which motivated the present work.

3 Numerical tools

The main numerical tool that we shall use to ensure both robustness and adaptation is the combination of interval arithmetic [17] for localizing isolated pieces of the mesh, and automatic differentiation [25, 17, 20, 13] for guaranteeing the topology of each piece and providing geometric adaptation. In this section we briefly review how these tools work.

Interval arithmetic. Given a function $f\colon \Omega \subseteq \mathbf{R}^3 \to \mathbf{R}$ and a rectangular box B contained in Ω, interval arithmetic computes an interval $F(B) \subseteq \mathbf{R}$ such that

$$F(B) \supseteq f(B) = \{f(x, y, z) : (x, y, z) \in B\}.$$

In other words, interval arithmetic provides a reliable estimate for the complete set of values taken by f in B. It

does so by extending all basic arithmetic operations and elementary functions to work on intervals instead of real numbers. To compute the interval estimate $F(B)$, we write $B = X \times Y \times Z$, where X, Y, and Z are intervals, and simply perform the same operations needed to evaluate f, except that we use the interval version of those operations and that we operate with intervals X, Y, Z instead of numbers x, y, z. By carefully rounding interval extremes outward, we obtain interval estimates that are reliable even when computed in floating-point machine arithmetic, which is subject to rounding errors.

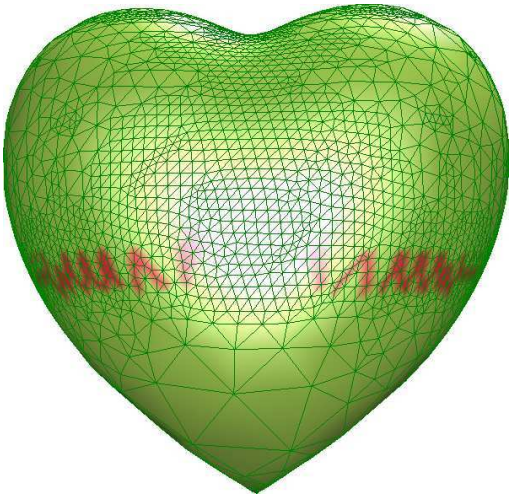

Figure 2. Geometry adapted meshing of the heart surface: $(2x^2+y^2+z^2-1)^6-(0.1x^2+y^2)z^3$

Implicit surface test. The power of using interval estimates in approximating an implicit surface S given by $f(x, y, z) = 0$ is that they provide the following reliable test for B *not* containing any part of S: if $0 \notin F(B)$, then $0 \notin f(B)$. This test, which is a simple consequence of $F(B) \supseteq f(B)$, is used as a stopping criterion for an octree subdivision of Ω (see Section 4). Note that this is *not* an approximate statement: $0 \notin F(B)$ is a *proof* that B does not intersect S; there is no sampling or guessing involved. On the other hand, the converse does not necessarily hold: $0 \in F(B)$ does not imply that B intersects S, because $F(B)$ may be strictly larger than $f(B)$. We only require that interval estimates satisfy $F(B) \supseteq f(B)$; we do not require that $F(B) = f(B)$. Finding the exact $f(B)$ is a global optimization problem; finding the estimate $F(B)$ is simply requires evaluating an interval expression and is thus performed efficiently. Moreover, interval estimates do get better as B shrinks. Thus, testing whether $0 \notin F(B)$ provides a fast and robust rejection test.

An octree exploration of Ω as described above is *spatially adaptive*, in the sense that it is guided by the location of S in Ω, To allow the exploration to depend also on the curvature of S, being *geometrically adaptive* as the example of Figure 2, we need to estimate how the curvature of S varies inside an octree cell. This can be done by using an interval estimate for the components of the gradient of f, because this gradient is normal to S.

Automatic differentiation. There are three main alternatives for computing the gradient of f: symbolic differentiation, which generates expressions for the partial derivatives of f; numerical differentiation, which finds approximations for the values of these partial derivatives; and automatic differentiation, which combines the speed of numerical differentiation with the accuracy of symbolic differentiation. Automatic differentiation works by defining an arithmetic for tuples (u, u_x, u_y, u_z), where u represents the value of a function of x, y, z, and u_x, u_y, u_z represent the values of its partial derivatives. For each basic arithmetic operation and for each elementary function, we can write a corresponding operation or function that operates on these tuples according to rules of calculus. For instance,

$$\sin(u, u_x, u_y, u_z) = (\sin u, u_x \cos u, u_y \cos u, u_z \cos u)$$

$$\exp(u, u_x, u_y, u_z) = (\exp u, u_x \exp u, u_y \exp u, u_z \exp u)$$

To evaluate a complicated expression, we simply write it as a sequence of primitive operations and follow the rules. At the end, we get the value of the expression *and* the values of all its partial derivatives. These derivatives are not approximate: the only possible errors are rounding errors, and these will be significant only when they are already significant for evaluating the function. This makes automatic differentiation at least as accurate as symbolic differentiation (and often more accurate). At the same time, automatic differentiation can be much more efficient than symbolic differentiation because obvious common sub-expressions are automatically identified and evaluated only once (see the rules above). In our algorithm, we use interval estimates for the gradient of f, obtained by applying the automatic differentiation formulas on tuples of intervals instead of tuple of numbers. The width of these interval estimates helps building a geometrically adaptive octree.

4 Robust adaptive polygonisation

This section details first how we combine these numerical tools to generate an adapted octree, and then how we extract an approximation of the implicit surface given by $f(x, y, z) = 0$ in Ω. We assume that f is twice continuously differentiable ($f \in C^2$) and admits zero as a regular value, which means that the gradient of f does not vanish on the implicit surface. This is actually a generic assumption, that is, it is restored by infinitesimal perturbations.

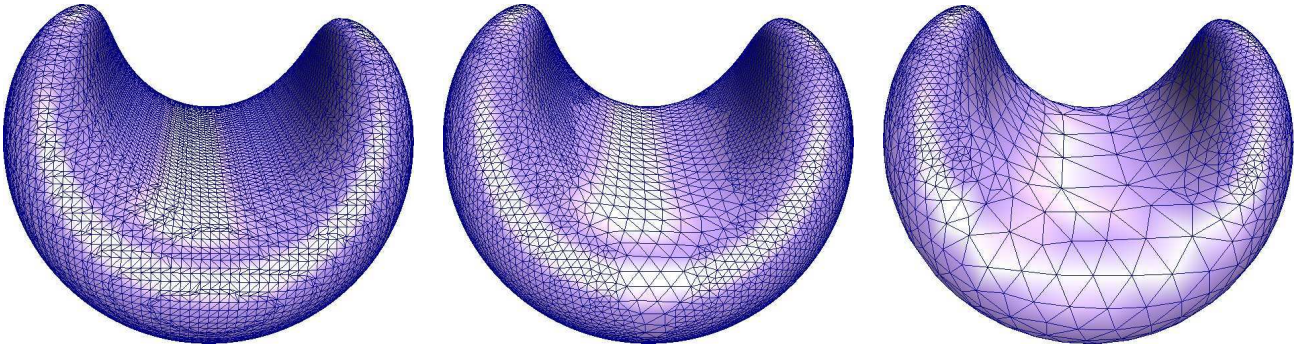

Figure 3. The effect of the geometric criterion on the smile surface $(y-x^2-y^2+1)^4+(x^2+y^2+z^2)^4-1=0$. **The mesh is uniform without this criterion (left), while it tracks regions of higher curvature when** k_{max} **increases, here from** $k_{max}=0.5$ **(middle) to** $k_{max}=0.95$ **(right).**

4.1 Building the octree

Our algorithm starts with the whole domain Ω, and assigns it to the root of the octree. Then, at each step of the subdivision, it checks whether the cell n, having domain the box B_n, must be subdivided. If so, the algorithm recurses on the subdivisions of n. A cell must be subdivided if it satisfies one of the three following criteria: *connected component* criterion, *topology* criterion and the *geometry* criterion. These criteria are derived from the interval evaluation $F(B_n)$ and $\nabla F(B_n)$ of both the function f and its gradient ∇f on the box B_n.

Connected component criterion. This criterion selects the cells of the octree that may contain a patch of the implicit surface, and discards the cells that surely do not intersect the surface. As detailed in Section 3, this can be tested robustly with the interval evaluation $F(B_n)$ of f on the box B_n:

$$\textbf{if} \quad 0 \notin F(B_n) \quad \textbf{then} \quad \mathsf{discard}(n)$$

This test is robust, meaning that it is guaranteed not to discard any connected component of the implicit surface $f^{-1}(0)$ in Ω: one of its points (x,y,z) would be contained in a cell n ($(x,y,z) \in B_n$), and thus $0 = f(x,y,z) \in f(B_n) \subseteq F(B_n)$, which avoids discarding n.

Topology criterion. A connected component of the implicit surface can have arbitrary genus, in particular it may contain tunnels. The previous criterion alone does not guarantee to recover tunnels. However, combined with the topology criterion, it can safely discard the empty parts of these tunnels. If a box B contains a tunnel, then the gradient of f varies inside B from one vector (n_x,n_y,n_z) at point (x,y,z) to its opposite $(-n_x,-n_y,-n_z)$ on the facing point [3]. Therefore, the coordinates I_x, I_y, I_z of the interval estimation of ∇f contain opposite values. Since I_x, I_y, I_z are intervals, $0 \in I_{x,y,z}$. The topology criterion is thus:

$$\textbf{if} \quad (0,0,0) \in \nabla F(B_n) \quad \textbf{then} \quad \mathsf{subdivide}(n)$$

Geometry criterion. To approximate correctly the geometry of the implicit surface with a reduced number of triangles, we need to allow small triangles only in regions of high curvature. The curvature can actually be estimated from $\nabla F(B_n)$: the curvature reflects the variation of the gradient. Therefore, high curvatures implies that the coordinates (I_x, I_y, I_z) of $\nabla F(B_n)$ are wide intervals. Given a user defined threshold k_{max} and choosing $Diam(I_x, I_y, I_z) = \max\{|I_x|, |I_y|, |I_z|\}$ for measuring the gradient variation, the geometry criterion is:

$$\textbf{if} \quad Diam\left(\frac{\nabla F(B_n)}{\|\nabla F(B_n)\|}\right) > k_{max} \quad \textbf{then} \quad \mathsf{subdivide}(n)$$

Note that the topology criterion guarantees that $0 \notin \|\nabla F(B_n)\|$, which validates the above expression. The parameter k_{max} actually weights the geometric adaptation, as illustrated on Figure 3.

Algorithm end. The above criteria may induce a large number of subdivisions, even infinite if the implicit surface has infinite genus (which is a highly non generic case). Moreover, since the interval evaluation $F(B)$ only contains the exact image $f(B)$ with conservative rounding error, subdivisions may be required on empty areas when numerical precision decreases. In practice, the algorithm subdivides the octree until a given maximal level, which may correspond to the size of the pixel for rendering applications, minimal size of the triangles for geometry processing, or numerical precision for simulation. However, the above criteria still point out which parts of the implicit surface are not guaranteed, while guaranteeing the approximation of the others. This robust behaviour allows stopping

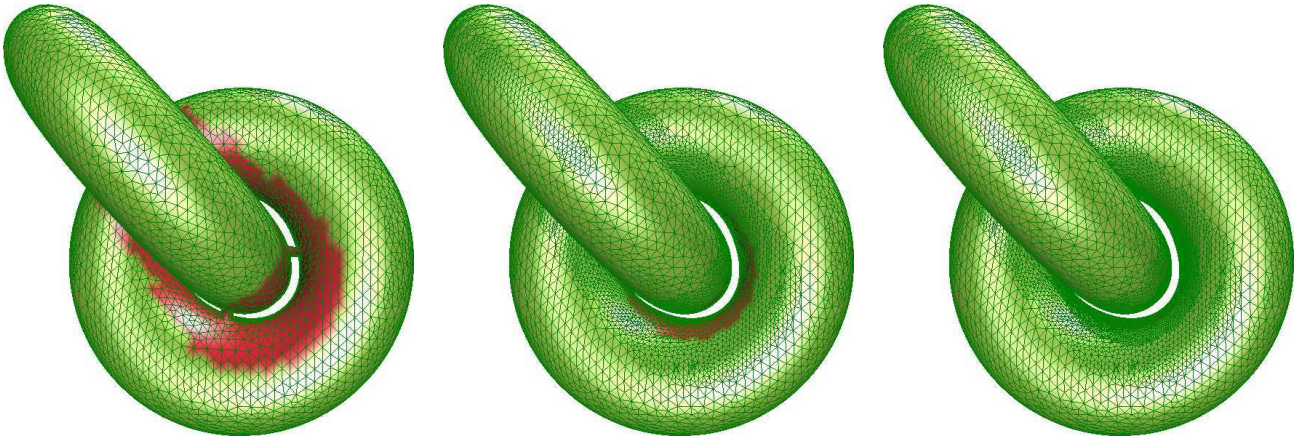

Figure 4. On complex models, such as the two torus surface $g(10x, 8y - 2, 10z, 13) \cdot g(10z, 10y + 2, 10x, 12) + 1000 = 0$ with $g(x, y, z, c) = (x^2 + y^2 + z^2 + c)^2 - 64(x^2 + y^2)$, the topology may not be guaranteed at a given resolution. At level 6 (left), our algorithm marks ambiguities regions (in red), and subdivides them, here stopping at level 7 (centre) and 8 (right) where the whole surface is robustly guaranteed.

the subdivision of the octree before the given maximal level if the approximation is already validated (see Figure 4).

4.2 From octree to dual grid

To generate the mesh we use an enhanced version of the Schaefer–Warren method [21]. Like them, we first create the *dual grid* of the octree, which is the topological dual of the octree: the vertices of the dual grid are the centre points of the octree cells, and the edges correspond to the adjacency between these cells. This way the each volumetric cell of the dual grid is associated with an interior vertex of the octree. Figure 5 illustrate this duality in the simpler case of quadtrees. In our method dual grid creation does not require any explicit neighbour representation in the octree data structure. Since the dual grid is generated once for the whole octree, this task is further optimized by traversing the octree from the root maintaining at each step the adjacencies of the nodes.

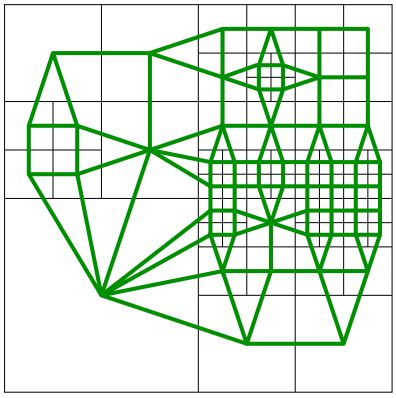

Figure 5. A primal quadtree (thin black) and its dual grid (thick green).

4.3 Mesh generation

The dual grid is then a coherent set of dual cells meeting at dual vertices, dual edges and dual faces. From the dual grid, we extract a mesh approximating the implicit surface by creating triangles in each of these dual cells. We use the public code of Lewiner et al. [9] to do so, which suits well here since the cells are convex and have at most 8 vertices. This algorithm will thus guarantee a manifold output with no crack, while working only locally. The mesh can be then further optimized in terms of aspect ratio by an inexpensive process described at the end of this section.

Cell key generation. Lewiner et al. [9] use a lookup table $L[\ ]$ that extends the one of the original Marching Cubes [16]: for each cube, they compute a key k from the values of its eight vertices and creates the triangles of the approximating mesh from the lookup table match $L[k]$ of the key.

The only difficulty here is to generate a key with eight entries since the dual cells may have less than eight vertices. This occurs when the dual vertices correspond to octree cells $\{c_i\}$ of different levels (see Figure 7). If we subdivided these octree cells until they had all the same level, the dual cell would be a cube. The key for the lookup table

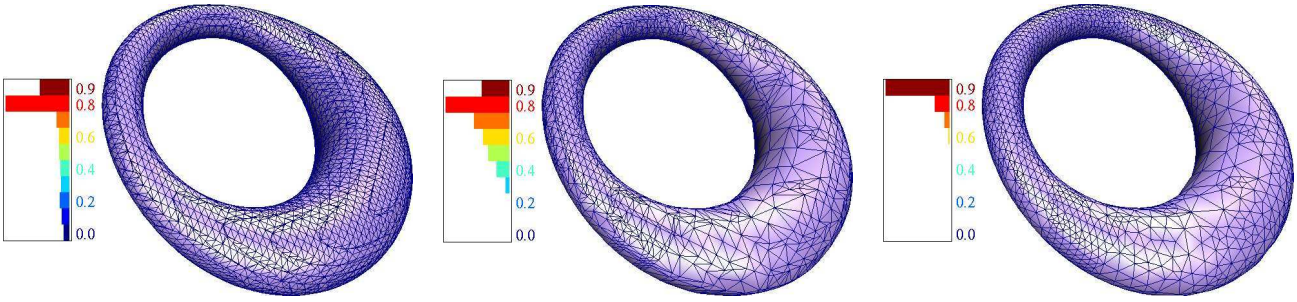

Figure 6. The Marching Cubes algorithm alone (left) generates many triangles which are not nicely shaped, as illustrated by the aspect ratio histograms. On the cyclide surface: $(x^2+y^2+z^2)^2-2(x^2+r^2)(a^2+b^2)-2(y^2-z^2)(a^2-b^2)+(a^2-b^2)^2+6abrx=0$ **with** $a=10$, $b=2$ **and** $r=2$**, it generated 11664 triangles, while the dual marching cubes (middle and right) generated only half of it (5396 triangles). The quality of the triangles can then be improved by simple mesh processing (right).**

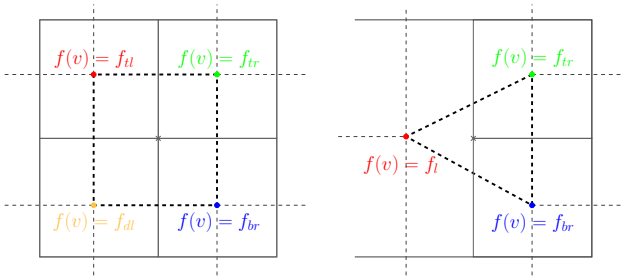

Figure 7. Key generation for the 2D lookup table: for the regular Marching Cubes case (left), the key is generated with the four entries $(f_{tr}, f_{tl}, f_{bl}, f_{br})$**. In the dual case (left), the value of the higher level node (the left one) has to be duplicated to generate a four–entry key** $(f_{tr}, f_l, f_l, f_{br})$**.**

is then generated by duplicating the values of the dual vertices which belong to the same octree cell c_i. In practice, we compute all the keys directly when generating the dual grid, without any extra octree subdivision.

Triangle creation. We use this key in the lookup table, and create the indicated triangles. Note that the tests of the asymptotic decider of Nielson and Hamann [10] are sufficient to guarantee the manifoldness of the results, but in case the subdivision of the octree is limited in space, the lookup table of Lewiner et al. [9] should recover the topology of the intersection better. The vertices of the indicated triangles can be computed exactly on the implicit surface, using a simple bisection method either along the dual edge, or along the normal estimated by automatic differentiation.

Further mesh improvements. The above process already guarantees a correct topology and an adaptive distribution of triangle sizes along the mesh. Although the mesh quality already passes the Marching Cubes one (see Figure 6), it can be further improved by sliding its vertices to have better shaped triangles. We use a technique similar to the one used by Botsch and Kobbelt [11]: for each vertex v, its normal $\vec{n_v}$ and the barycentre b_v of its star are computed. The vertex is then shifted tangentially towards b_v, which writes $v \leftarrow v + (\vec{vb_v} - \langle \vec{vb_v} | \vec{n_v} \rangle \cdot \vec{n_v})$. The tangential movement avoids strong shrinking, and the relaxation towards b_v improves the aspect ratio of the triangles. The vertices are then projected back onto the implicit surface by the bisection mentioned above (see Figure 6). Note that this process is applied on all the vertices at once, and thus does not require adjacency structure for the mesh.

5 Results

We checked the validity of our method on various models. In particular, the robustness paradigm is clearly illustrated on Figure 1: The red regions on this figure indicate that the approximation obtained on that region is not guaranteed, although the algorithm provides a valid global surface. The red region marks that either the topology or the geometry criterions on the corresponding primal node of the octree failed. Observe on the left image that the obtained surface has genus 0 for that level of approximation (4). These ambiguities can be solved by allowing more subdivision. Even if our algorithm is conservative, as illustrated on Figure 4, the ambiguities are solved in practice with only a few subdivisions (until level 6 and 8 for Figures 1 and 4).

The proposed geometric criterion produces well adapted meshes, as the one of Figure 2, with a reduced number of triangles (see Table 1). Its effect is monitored by k_{max}, the only manual parameter of our algorithm. Figure 3 shows how the triangles concentrate on the regions of high curva-

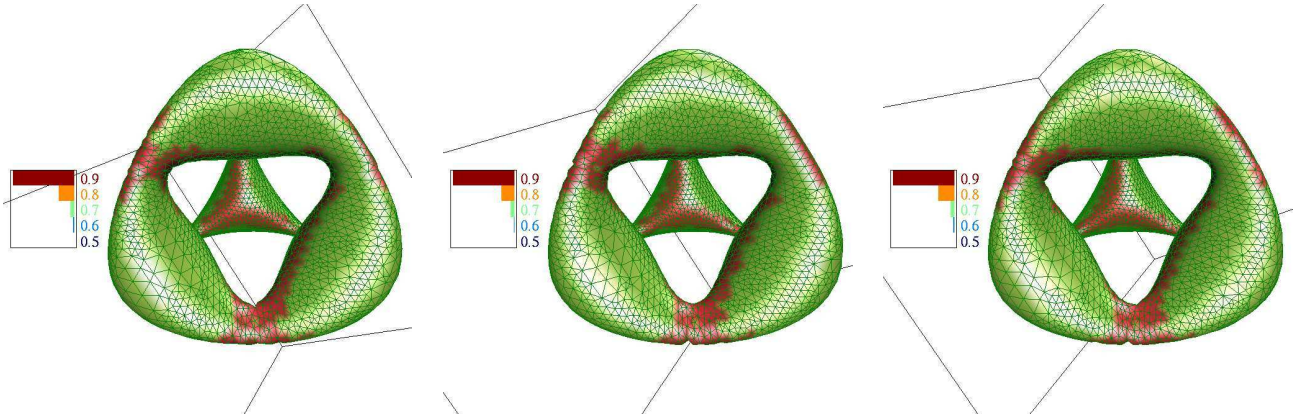

Figure 8. Our algorithm does not suffer from symmetry artefacts. For example, on the chair surface $(x^2+y^2+z^2-ak^2)^2-b((z-k)^2-2x^2)((z+k)^2-2y^2)=0$ with $a=0.95$, $b=0.8$, and $k=5$, it detects the same ambiguities and shows similar triangulations independently on the cube position Ω: $[-10.2,5.8]\times[-10.2,5.8]\times[-8,8]$ (left), $[-8,8]^3$ (middle) or $[-5.8,10.2]\times[-5.8,10.2]\times[-8,8]$ (right).

ture when increasing this parameter. This adaptability cannot be achieved directly by the Marching Cubes method, as illustrated by Figure 6. On this figure, the histogram of the aspect ratio ($\frac{4\sqrt{3}\cdot area}{|ab|^2+|ac|^2+|bc|^2}$) shows that the our method already produces high quality mesh, which can be improved by the method of the end of Section 4. Moreover, the quality of the mesh does not result from a symmetry artefact, as illustrated on Figure 8.

Our algorithm handles nicely degenerated cases, such as the singular points (pinch) of the tear drop model of Figure 9. Even if we do not recover the feature, the algorithm marks that the approximation needs refinements. The case is similar for non–manifold surfaces, as the two intersecting planes of Figure 10. The algorithm reconstructs a manifold approximation, on which the non–manifold parts of the real implicit object are marked as ambiguous.

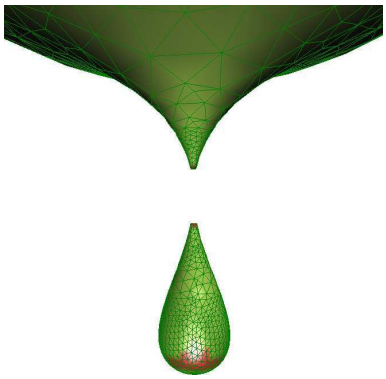

Figure 9. The singularities of the teardrop surface $0.5x^5+0.5x^4-y^2-z^2=0$ are detected by our criterion, but the meshing does not recover them completely.

6 Conclusion

This work addressed the problem of computing good polygonal approximations of implicit surfaces. We proposed a robust algorithm that can either guarantee the exact topology of the implicit surface or point to the user ambiguous parts, which can be solved by further refinements. The robustness is achieved by the combination of interval arithmetic with automatic differentiation into adaptation criteria. We then extract an approximate triangle mesh from this octree through a dual Marching Cubes algorithm. We finally improve the quality of the triangles in the final mesh by simple and efficient geometry processing.

We do guarantee the topological consistency for manifolds: the mesh does not contain cracks and it does have the correct genus. While we do not promise to handle all singular cases, our algorithm clearly show regions that may contain singular points or curves. Since we are dealing only with continuous data, as opposed to discrete data, we do not use any heuristic. In particular, we can use bisection to locate, with great accuracy, the intersections of the surface with the boundary of a cell.

Acknowledgments

A. Paiva, H. Lopes, T. Lewiner are members of the Matmidia lab., which is sponsored by PETROBRAS, CNPq, and FAPERJ. L.H. de Figueiredo, of the Visgraf lab., is sponsored by CNPq, FAPERJ, FINEP, and IBM Brasil.

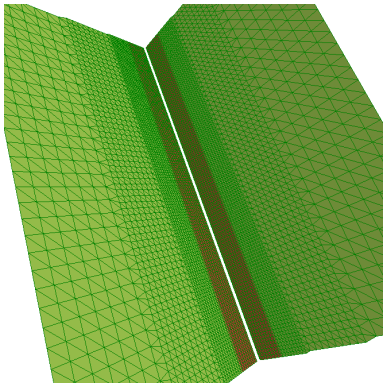

Figure 10. Our algorithm always reconstructs a manifold surface, even in the case of two intersecting planes $xy = 0$. However, the non–manifold parts (in red) are detected by our criterion.

Model	k_{max}	level	# trian.	# vert.	aspect
Torus	4.9	6	7248	3512	92%
Tear drop	0.8	7	2590	1272	97%
Smile	0	6	25172	12588	55%
Smile	0.5	6	22408	11100	90%
Smile	0.95	6	4948	2320	85%
2 torus	1.9	6	17588	8652	92%
2 torus	1.9	7	39700	19212	88%
2 torus	1.9	8	83252	40228	87%
chair left	0.95	6	10564	5096	90%
chair mid	0.95	6	10564	5098	89%
chair right	0.95	6	10564	5096	89%
cyclide	1	5	5396	2528	85%
heart	1.2	6	9048	4068	78%
2 planes	0.1	7	24820	12416	99%

Table 1. Results on the illustrations of the paper. The aspect **column counts the triangles which have an aspect ration over 0.8.**

References

[1] R. J. Balsys and K. G. Suffern. Visualisation of implicit surfaces. *Computers & Graphics*, 25(1):89–107, 2001.

[2] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988.

[3] J.-D. Boissonnat, D. Cohen-Steiner, and G. Vegter. Isotopic implicit surface meshing. In *Symposium on Theory of computing*, pages 301–309. ACM, 2004.

[4] B. R. de Araujo and J. A. P. Jorge. Adaptive polygonization of implicit surfaces. *Computers & Graphics*, 29(5):686–696, 2005.

[5] A. C. Filho, L. G. Nonato, M. Siqueira, R. Minguim, and G. Tavares. The j1a triangulation: an adaptive triangulation in any dimension. *Computers & Graphics*, 30, 2006.

[6] M. Hall and J. Warren. Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, 10(6):33–42, 1990.

[7] E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, 1998.

[8] C.-C. Ho, F.-C. Wu, B.-Y. Chen, Y.-Y. Chuang, and M. Ouhyoung. Cubical marching squares: Adaptive feature preserving surface extraction from volume data. *Computer Graphics Forum*, 24(3):537–545, 2005.

[9] 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.

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

[11] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. In *Symposium on Geometry Processing*, pages 185–192. ACM/Eurographics, 2004.

[12] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *SIGGRAPH*, pages 339–346. ACM, 2002.

[13] H. Kagiwada, R. Kalaba, N. Rasakhoo, and K. Spingarn. *Numerical Derivatives and Nonlinear Analysis*. Plenum, New York, 1986.

[14] T. Karkanis and A. J. Stewart. Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics and Applications*, 22(2):60–69, 2001.

[15] H. Lopes, J. B. Oliveira, and L. H. de Figueiredo. Robust polygonal adaptive approximation of implicit curves. *Computers & Graphics*, 26(6):841–852, 2002.

[16] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, pages 163–169. ACM, 1987.

[17] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[18] G. M. Nielson. Dual marching cubes. In *Visualization*, pages 489–496. IEEE, 2004.

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

[20] L. B. Rall. The arithmetic of differentiation. *Mathematics Magazine*, 59(5):275–282, 1986.

[21] S. Schaefer and J. Warren. Dual marching cubes: Primal contouring of dual grids. In *Pacific Graphics*, pages 70–76. IEEE, 2004.

[22] M. F. W. Schmidt. Cutting cubes: visualizing implicit surfaces by adaptive polygonization. *The Visual Computer*, 10(2):101–115, 1993.

[23] B. T. Stander and J. C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *SIGGRAPH*, pages 279–286. ACM, 1997.

[24] L. Velho, L. H. de Figueiredo, and J. Gomes. A unified approach for hierarchical adaptive tesselation of surfaces. *ACM Transactions on Graphics*, 18(4):329–360, 1999.

[25] R. E. Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.