Visual Comput (2007) 23: 45–58
DOI 10.1007/s00371-006-0083-6

ORIGINAL ARTICLE

Simon Plantinga
Gert Vegter

Isotopic meshing of implicit surfaces

Published online: 31 October 2006

S. Plantinga (✉) · G. Vegter
University of Groningen, Groningen, The Netherlands
{simon, gert}@cs.rug.nl

Abstract Implicit surfaces are given as the zero set of a function $F\colon \mathbb{R}^3 \to \mathbb{R}$. Although several algorithms exist for generating piecewise linear approximations, most of these are based on a user-defined stepsize or bounds to indicate the precision, and therefore cannot guarantee topological correctness. Interval arithmetic provides a mechanism to determine global properties of the implicit function. In this paper we present an algorithm that uses these properties to generate a piecewise linear approximation of implicit curves and surfaces, that is isotopic to the curve or surface itself. The algorithm is simple and fast, and is among the first to guarantee isotopy for implicit surface meshing.

Keywords Implicit surfaces · Approximation · Isotopy · Meshing

1 Introduction

Implicit functions provide a convenient representation of smooth surfaces. However, piecewise linear approximations are often required, for example for visualization. Meshing algorithms can only compute function values at a finite number of points. Since grid based schemes can miss important details of the surface, correct topology usually cannot be guaranteed. To capture the global properties of implicit surfaces, we somehow need to extract information about the surroundings of these points. Lipschitz conditions give a bound on the gradient and can therefore sometimes discard the neighborhood of a point. Another tool that can be used is interval arithmetic.

In this paper we present an algorithm that creates a regularly isotopic approximation of implicit curves in $\mathbb{R}^2$ and of implicit surfaces in $\mathbb{R}^3$. Regularly isotopic means that the approximation is equivalent to the curve or surface under continuous deformation within the embedding space. In particular, the approximation has the same topology as the implicit manifold. Correct topology is important to determine connectedness, and for example in constructing an initial mesh for time-dependent surfaces (e.g., morphing). Our implicit surface meshing algorithm subdivides space using an octree. Interval arithmetic is used to decide which cells of the octree require further subdivision. For each leaf of the resulting octree a local approximation is constructed.

The algorithm given in this paper falls in the category of enumeration methods. A novelty is that we use a fast and simple interval test to decide which cells need subdivision, in such a way that we can guarantee isotopy. Compared to other enumeration methods it adds little overhead, and is therefore quite fast. In particular, we do not need to compute the critical points of the implicit function.

The octree-based subdivision is very flexible. For example we could define a minimal subdivision level to improve the Hausdorff distance between the surface and the approximation. A maximal subdivision level could be

used to improve the speed of the meshing process and to handle surfaces with singularities. In this case color coding of the mesh could be used to identify the (arbitrarily small) regions where the approximation might be topologically incorrect.

In Sect. 2 we give an overview of the existing techniques for implicit curve and surface approximation. Since our algorithm is based on interval arithmetic, in Sect. 3 we give a brief overview of this technique. Section 4 explains our approximation algorithm for implicit curves. After giving the algorithm, we will prove that the resulting piecewise linear approximation is isotopic to the curve itself. Both the algorithm and its proof will be generalized to the three-dimensional case. This section therefore can also be regarded as an introduction to the meshing of implicit surfaces. In Sect. 5 we provide a first step towards our meshing algorithm, by adapting the well-known marching cubes algorithm to construct isotopic approximations from a regular grid. Section 6 extends the previous section by using octrees instead of a regular grid, thereby greatly reducing the complexity of both the algorithm and the resulting mesh. To facilitate the implementation of our algorithm, in Sect. 7 we explain how to construct a tetrahedrization of the octree such that meshing its tetrahedra also results in an isotopic mesh. Locally meshing tetrahedra is much more straightforward than meshing octree leaves, due to the lack of ambiguous cases and the smaller number of mesh triangles for each cell. Results of the algorithms are shown in Sect. 8. Finally, in Sect. 9 we discuss some possible improvements and possible changes to our approximation scheme.

Main contribution. This paper presents a new algorithm to approximate not necessarily algebraic regular implicit curves and surfaces. For surfaces, it is one of the first schemes guaranteeing that the resulting mesh is isotopic to the implicit surface. For curves, it is the first practical algorithm giving this guarantee. Since we do not need to compute the critical points of the implicit function, both algorithms are fast enough to be of practical use. Also, they can easily be adapted to improve the accuracy of the approximation and to deal with singular surfaces. We expect that the algorithm generalizes to isotopic approximation of all codimension one manifolds in Euclidean spaces.

2 Related work

Several algorithms exist for the approximation of implicit curves and surfaces, but very few can guarantee topological correctness. Since we can only compute function values at a finite number of points, there is a risk of missing important details of the curve or surface. Some approximation algorithms depend on a user-specified value to give a trade-off between accuracy and speed. For these methods it is not clear what level of accuracy is required to capture the topology of the implicit manifold. Algorithms that do guarantee a topologically correct result depend either on extra information about the underlying implicit function (e.g., algebraicity of surfaces or bounds on the Lipschitz conditions), or on interval arithmetic.

In this section we will give an overview of the different existing techniques for approximation of implicit curves and surfaces.

Curves. The schemes for approximation of implicit curves can be roughly divided into continuation methods and adaptive enumeration techniques. *Continuation methods* approximate the curve or surface by first finding seed points on all components, and then using these as starting points for tracing the components.

In [8] an implicit curve (in this case a contour in $\mathbb{R}^3$) is approximated using a predictor-corrector method. Initial points are found by shooting random rays to the surface and by moving from the first intersection point along the surface towards the contour. In [15] we adapted this method to guarantee topological correctness of an approximation of the contour generator, but to achieve this the relatively slow interval Newton method was required. Also, a small stepsize for the tracing process is required to guarantee that the resulting approximation is isotopic to the traced component.

Approximation schemes based on *adaptive enumeration* start with a bounding box and subdivide until the cells are "small enough", for example by looking at the local curvature. Then, for each cell of the subdivided box a local approximation is constructed. In [13] interval arithmetic is used to get a good approximation of implicit curves. The size of the subdivided cells depends on the geometry of the curve, such that in areas with high curvature the level of detail is also higher. However, to terminate the subdivision process, the method requires user-specified bounds on cell size, and on the variation of the normalized gradient within a cell. Although the edge length varies with the curvature of the curve, it cannot guarantee correct topology.

As in this paper, [17] uses parametrizability to subdivide a quadtree. This algorithm guarantees topological correctness, but because of a slightly weaker test for parametrizability, it requires a large number of (slow) interval Newton searches for intersection points.

Surfaces. A continuation algorithm is given in [11]. Starting with a seed point on the surface, a triangulation is generated by expanding over the surface, along the boundary of the already triangulated part. As with curves, this algorithm depends on seed points, and on a user-specified stepsize. The triangles of the resulting mesh have more or less the same size, independent of the geometry of the surface. An improvement of this algorithm using a dynamic

triangle size is given in [1]. Enumeration methods use either a regular grid or an octree to subdivide space. Regular grids are used in the well-known marching cubes algorithm [14]. The main purpose of these methods is to mesh sampled voxel data, but they can easily be adapted to approximate implicit surfaces. In [3], cubes are constructed along the surface. Although this saves examining a large number of grid cells, there is a risk of missing components of the implicit surface. Another improvement of marching cubes is given in [10]. The linear variation along an edge used in marching cubes, is extended to trilinear variation over the cube, using the function values at the eight corners of the cube. The algorithm guarantees that the approximation has the same topology as the isosurface of this trilinear function. For implicit surfaces there is no such guarantee.

For dynamic surfaces, critical points and their indices indicate when, where and how the topology changes. This can be used to update the topology of the triangulation, as in [18]. Also, the shrinkwrap algorithm [19] can be adapted to use critical points for updating the topology [7].

An algorithm for isotopic meshing based on critical points is given in [4]. It is based on Morse theory to determine the topology, and therefore requires a priori knowledge of the critical points of the implicit surface, as well as of their indices.

Finally, sampling based algorithms construct a sufficiently dense sample of points on the surface, such that surface reconstruction on this point set results in a homeomorphic approximation. To construct such a sampling, in [5] bounds on the distance to the medial axis are needed, while in [9] the critical points of height functions on the surface and on intersection curves are required. Also, both algorithms assume that the intersection points of a given line segment with the surface can be computed. These Delaunay-based algorithms result in a better mesh quality, but for generic implicit surfaces it is very difficult to meet these assumptions.

An extended abstract of this paper appeared in [16]. In this full paper we provide the proof of isotopy for octree-based meshing. Furthermore, we give a more practical mesh construction algorithm based on tetrahedral subdivision.

A similar algorithm for algebraic surfaces was presented in [2], using so-called u-regularity. By restricting the meshing algorithm to algebraic functions, a rough bound on the complexity can be given.

3 Interval arithmetic

One way to prevent rounding errors due to finite precision numbers is to use interval arithmetic. These intervals can be considered as a value, together with an error bound.

An *inclusion function* $\square f$ for a function $f\colon \mathbb{R}^m \to \mathbb{R}^n$ computes for each m-dimensional interval I (i.e., an m-box) an n-dimensional interval $\square f(I)$ such that

$$x \in I \Rightarrow f(x) \in \square f(I).$$

An inclusion function is said to be *convergent* if the width of the output interval converges to 0 when the (largest) width of the input interval shrinks to 0.

For example, if $f\colon \mathbb{R} \to \mathbb{R}$ is the squaring function $f(x) = x^2$, then a convergent inclusion function $\square f([a, b])$ is given by

$$\begin{cases} [\min(a^2, b^2), \max(a^2, b^2)], & a \cdot b < 0 \\ [0, \max(a^2, b^2)], & a \cdot b \geq 0 \end{cases}$$

When using interval arithmetic to prevent rounding errors, the width of the intervals should be as small as possible in order to give accurate results. We will use interval arithmetic in a different way, and compute function values over large intervals. For example, if the function value computed over a large box results in a strictly positive interval, we can conclude that the function has no zeros within that interval. In other words, the implicit manifold does not intersect this box.

Convergent inclusion functions exist for the basic operators and functions. To compute an inclusion function it is often sufficient to replace the standard number type (e.g., `double`) by an interval type (`Interval`), using an appropriate interval library, such as `Filib++` [12].

4 Curves

In this section we will introduce an algorithm to construct a piecewise linear approximation of an implicit curve S, such that the approximation and S have the same (regular) isotopy. The construction is based on a quadtree. After subdivision this quadtree is *balanced*, i.e., the tree is further subdivided until two neighboring squares differ at most a factor two in size. Balancing does not increase the complexity of the tree.

Possible improvements regarding accuracy and singularities will be discussed in Sect. 9.

Let $S = F^{-1}(0)$ be a bounded implicit curve, where $F\colon \mathbb{R}^2 \to \mathbb{R}$ is a smooth function and 0 is a regular value of F, i.e., the gradient ∇F is nonzero at every point of the curve. The following algorithm constructs an isotopic piecewise linear approximation of S. Details are explained right after the presentation of the algorithm.

Algorithm. APPROXIMATECURVE(F, B)

Input. An implicit function F, and a square bounding box B.
Output. A piecewise linear approximation of the curve $F = 0$.
1. Initialize a quadtree $\mathcal{T}$ to the bounding square B of $F = 0$.
2. Subdivide $\mathcal{T}$ until for all leaves C we have $0 \notin \square F(C) \vee \langle \square \nabla F(C), \square \nabla F(C) \rangle > 0$.

```
3.  BALANCEQUADTREE(T)
4.  for each edge of the quadtree
5.      do if the signs of F at its two endpoints are opposite
6.          then construct a vertex at the midpoint of the edge
7.  for each leaf C of quadtree T
8.      do if the leaf contains two vertices
9.          then connect the vertices by a line segment
10.     if the leaf contains four vertices
11.         then find the two vertices on the same side and
                connect each of them to the other adjacent
                vertex
```

Note that a single side of a cell may consist of two quadtree edges, if the neighboring cell along that side is subdivided. Each cell of the quadtree is therefore surrounded by four to eight quadtree edges.

The construction of the approximating edges handles only a few different cases of vertex placement on the boundary of a cell: either two vertices, or four vertices with two of them along one side of the cell. In the correctness proof we will show that these are the only possible cases satisfying our subdivision condition.

Before proving that this algorithm constructs an isotopic approximation, we will show that the resulting quadtree cells are parametrizable and prove that the algorithm terminates by examining the interval condition:

$$0 \notin \Box F(C) \vee \langle \Box \nabla F(C), \Box \nabla F(C) \rangle > 0.$$

The first clause discards cells where $0 \notin \Box F(C)$. Note that this discards boxes of which we are certain that they do not contain part of the curve S.

The right hand clause implies that $\langle \nabla F(x), \nabla F(y) \rangle > 0$, for all $x, y \in C$. Hence, the direction of the gradient (and, therefore, of the curve) does not change more than $\pi/2$ over C.

We say the curve S is *parametrizable in direction* v, if every line parallel to v intersects S at most once. This definition extends easily to three dimensions.

If $\langle \Box \nabla F(C), \Box \nabla F(C) \rangle > 0$, at least one of the two terms $\Box F_x(C) \cdot \Box F_x(C)$ and $\Box F_y(C) \cdot \Box F_y(C)$ (where we write F_x for $\frac{\partial F}{\partial x}$) does not contain 0. This implies that F is strictly increasing or decreasing in the x or y direction, and therefore locally (i.e., within this cell) parametrizable in the direction of one of the axes.

When we continue the subdivision process, the squares shrink towards a point. Since S is a regular curve, at least one of the two clauses converges to a nonzero value. Therefore, the subdivision process terminates.

After these observations we can move on to the following theorem:

Theorem 1. *The approximation of S constructed by algorithm* APPROXIMATECURVE *is (regularly) isotopic to S.*

Proof. To prove that the approximation is isotopic to the implicit curve C we will proceed as follows:

1. Firstly, we construct an approximation using a regular grid, assuming that S satisfies certain constraints, and show that this approximation is isotopic to S.
2. Secondly, we will remove the constraints on S.
3. Finally, we will show that the approximation constructed by the regular grid is equivalent to the one created by APPROXIMATECURVE. To this end, we will show that further subdivision of the quadtree does not change the isotopy of the corresponding approximation. By repeating the subdivision process until the quadtree is complete, we end up with a regular grid. This grid was already shown to be isotopic to the implicit curve.

Regular grid. For the first part of the proof we will start with a grid-based approximation instead of a quadtree. Let G be a regular grid, such that for each cell C we have $0 \notin \Box F(C) \vee \langle \Box \nabla F(C), \Box \nabla F(C) \rangle > 0$. Furthermore we assume that S intersects each edge of G at most once, and that $F \neq 0$ at the nodes of G. These assumptions will be removed later. This implies that we have to construct at most one vertex at each edge of the grid. Due to the inner product constraint, S is parametrizable in the direction of one of the axes. Therefore we cannot have alternating signs of F at the vertices of C, since F would have to increase along one edge, and decrease along the other parallel edge (Fig. 1). We conclude that S intersects at most two edges of C. For the approximation we connect the two midpoints of these two edges (if any) by a straight line segment s.

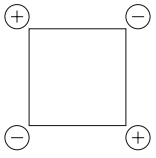

Fig. 1. A sign configuration contradicting the inner product constraint. F decreases along the top edge and increases along the bottom edge. It is therefore not parametrizable in the x-direction. A similar argument holds for the y-direction

The curve S is locally parametrizable, and therefore cannot contain closed loops inside a cell. Also, since S is a regular curve, there are no self-intersections. This implies that within the cell C, if there are two intersection points along the edges, the part of the curve inside C is isotopic to a line segment.

In fact we can easily construct a deformation to move the curve locally to the segment s of the approximation. Suppose for example that S is locally parametrizable in the y-direction, i.e., within the grid cell it is of the form $y = f(x)$. If S intersects the left and right edge, the approximation has the same x-domain as S. By linear interpolation in the y-direction we can continuously move this part of S to segment s. If S does not intersect the left and right edge, we may need to stretch/shrink the x-domain first. An example is given in Fig. 2. By mapping

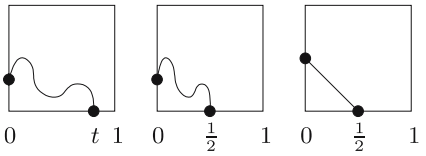

Fig. 2. Deformation of the curve (left) to the approximation (right)

$[0, t]$ linearly to $[0, \frac{1}{2}]$ and $[t, 1]$ to $[\frac{1}{2}, 1]$ the intersection point $(t, 0)$ moves to $(\frac{1}{2}, 0)$. Now, both curves have the same x-domain, so we can do a vertical interpolation again. Note that an edge of the grid maps onto itself. Therefore, we can stitch the local deformations within each square together, resulting in a global deformation that moves the whole implicit curve continuously to the approximation.

Removing constraints. In the second part of the proof we remove the constraints on S. The first assumption was that S intersects each edge of the grid at most once. Now, suppose S intersects an edge of the grid more than once. In this case the curve must be parametrizable in the direction perpendicular to this edge, for both adjacent cells. For two adjacent intersection points α_1 and α_2 we look at the curve between these two points. Using the mean value theorem it is easy to see that the inner product constraint prevents S from bending "too much" (i.e., more than $\pi/2$). Also, recall that the adjacent cells are squares. Therefore, S cannot leave the two cells between α_1 and α_2 (Fig. 3). By a local interpolation in the parametrizable direction between S and the edge, we can continuously deform the part of S between α_1 and α_2 towards the edge, and "push" this part of S through the edge (Fig. 4), thereby removing the two intersection points. Since we can continue removing pairs of intersection points, S is isotopic to a curve that intersects each edge at most once, and hence it is isotopic to the piecewise linear approximation.

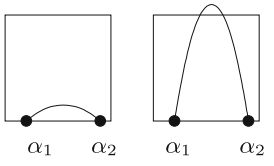

Fig. 3. Multiple intersection along an edge. The curve on the right has too much curvature to satisfy the interval constraint

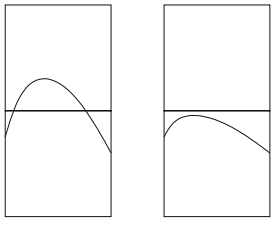

Fig. 4. Removing a pair of intersection points by pushing the curve through the edge. In the union of the two squares the topology of the curve does not change

The second assumption was that $F \neq 0$ on nodes of the grid. If S passes through a node of the grid, we can again deform S, this time by moving it continuously to $F + \epsilon = 0$. For small ϵ this again yields an isotopy, since S is regular. We can take ϵ arbitrarily small using a symbolic perturbation, by considering F to be strictly positive at a vertex whenever $F \geq 0$.

Quadtrees. Now that we have removed the constraints on F we complete the proof by showing that the quadtree approximation is isotopic to the grid approximation. To this end, we consider a leaf C of the quadtree $\mathcal{T}$, together with its approximation edges. When we split C, the inner product constraint still holds for its four children. Furthermore, since S is parametrizable within C, S is parametrizable in its children, and also in the same direction. The only topological changes can therefore occur if subdivision introduces two new vertices at a single edge of C, as shown in Fig. 5.

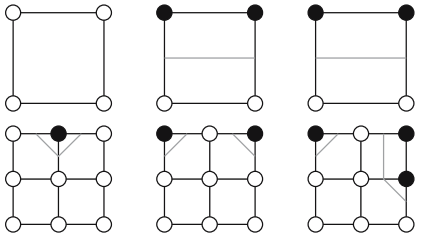

Fig. 5. Topological changes after subdivision

Since two new vertices were created, the neighboring cell along the edge containing these vertices is not subdivided. After subdivision of C, the neighboring cell will detect the two new vertices and connect them either to each other or to two existing vertices. Both types of change correspond to "pushing" part of the curve through the edge. In the union of the cell and its neighbor the approximation is isotopically unaltered (Fig. 6).

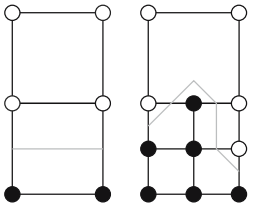

Fig. 6. After subdivision the topology inside a square changes, but inside the union of both squares it stays the same

By repeatedly subdividing the leaves of $\mathcal{T}$ we can turn it into a complete quadtree, with the same approximation as a regular grid. Since this does not change the isotopy of the approximation, the output of algorithm APPROXIMATECURVE is isotopic to S.

5 Marching cubes revisited

For the meshing of implicit surfaces we proceed similarly to the two-dimensional case. As a first step towards our meshing algorithm, we approximate the surface on a regular grid, assuming only single intersections on the edges. Later, we shall remove these constraints. In the next section we will present our meshing algorithm, using a balanced octree instead of a regular grid.

In this section we introduce a slightly modified version of the well-known marching cubes ($\mathcal{MC}$) algorithm. Recall that $\mathcal{MC}$ subdivides space into a uniform cubic grid, constructing a triangulation for each cell of the grid, depending only on the function sign at its eight vertices. Sometimes the function value at the vertices is used to position the vertex by linear interpolation. Since our main goal for now is topological correctness we will just put the vertex at the center of the edge. In the more convenient tetrahedral algorithm (Sect. 7), we shall use linear interpolation for a better looking mesh.

The $\mathcal{MC}$ algorithm as introduced by Lorensen and Cline [14] cannot guarantee topological correctness. Due to ambiguities in the sign configuration of a cell face, the resulting surface could also contain holes. To fix these problems we will introduce a few assumptions, most of which can later be removed.

Let S be a regular, bounded, implicit surface, given as the zero set of a smooth function $F\colon \mathbb{R}^3 \to \mathbb{R}$, so $S = F^{-1}(0)$. Furthermore, we assume that 0 is a regular value of F, i.e., the gradient ∇F is nonzero at every point of the surface.

For now, we will assume that S does not contain vertices of the uniform grid, and that S intersects each edge of the grid at most once. Furthermore, the following interval condition should hold for each cell C:

$$0 \notin \square F(C) \vee 0 \notin \langle \square \nabla F(C), \square \nabla F(C) \rangle.$$

As in the two-dimensional case, the right hand clause implies:

$$\forall x, y \in C\colon \langle \nabla F(x), \nabla F(y) \rangle > 0.$$

Again we have that for each cell where $0 \in \square F(C)$ the surface S is parametrizable in the direction of one of the axes.

For the signs at the vertices, there are 14 possibilities (up to rotation, mirroring and change of sign), shown in Fig. 7.

Of these, configurations 5, 8, 12, 13 and 14 are impossible, due to the interval condition. For example in configuration 5 (see Fig. 8), on edges $\overline{cd}$ and $\overline{ef}$ the function changes sign in opposite directions. Therefore, F is not parametrizable in this direction. For $\overline{ad}$, $\overline{fg}$ and for $\overline{bf}$, $\overline{dh}$ we find that F is not parametrizable in the direction of the other two axes, contradicting the inner product condition.

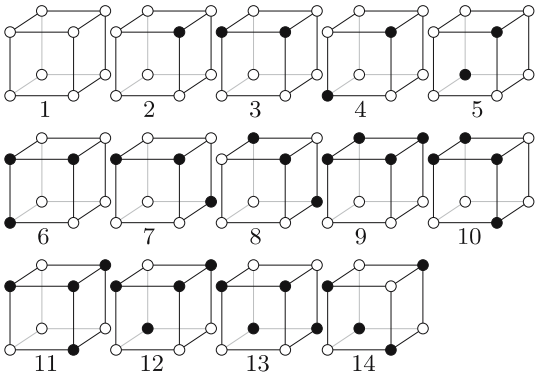

Fig. 7. The 14 possible sign configurations

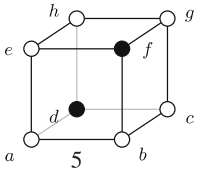

Fig. 8. Configuration 5

For configurations 8, 12 and 14 a similar argument holds. For configuration 13 we have to look at the diagonal pairs ($\overline{ac}$, $\overline{eg}$), ($\overline{bd}$, $\overline{fh}$) and ($\overline{ae}$, $\overline{cg}$), and note that the inner product condition does not depend on any particular orthogonal coordinate system.

The remaining 9 configurations can now be triangulated. Two of these still contain an ambiguity (case 4 and 6 in Fig. 9). To resolve this, note that the ambiguity is due to the sign configuration on a face of the cell. In both cases we have a single face where the function changes sign along all of its four edges, resulting in four vertices of the triangulation. There are two ways to connect these vertices pairwise. We will choose to connect them, such that the resulting segments are parallel to one of the vectors (1, 1, 0), (1, 0, 1) and (0, 1, 1), depending on the orientation of the face. This choice guarantees that the individual triangulations of two adjacent cells fit together. We have to show that this choice does not affect the isotopy.

Since cells have at most one ambiguous face, we regard the block of two cubes adjacent to such a face (see

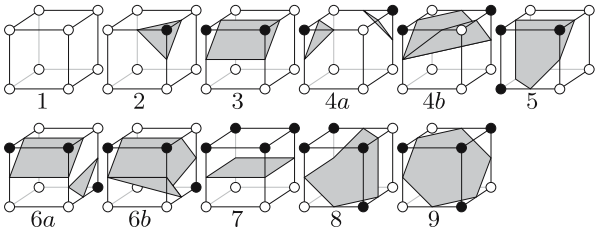

Fig. 9. Triangulation of the subcubes

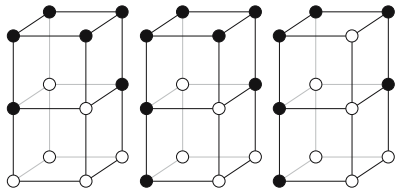

Fig. 10. Two cubes sharing an ambiguous face: two type 4 cells, a type 4 and a type 6 cell, and two type 6 cells

Fig. 10). Note that the border of the union of these two cubes does not contain ambiguities. The sign changes along the edges of the shared face prevent a parametrization in the direction of one of these edges. Both cubes are therefore parametrizable in the vertical direction. Since the vertical component of the gradient does not disappear, the union of these two cells is also parametrizable in the vertical direction. This leads to only three possibilities to join two ambiguous cells, shown in Fig. 10.

For the ambiguous face we have two ways to connect the vertices on the edges pairwise. Both possibilities are shown in Fig. 11. Since both choices lead to the same isotopic approximation, we can choose either of them, as long as the triangulations for both cells fit together.

If the surface intersects the grid-faces in curved segments connecting the intersections of S with the edges of the grid, we are done. However, it is possible that S intersects a face of the grid in a closed curve, lying completely withing that face (Fig. 12). In this case both cubes adjacent

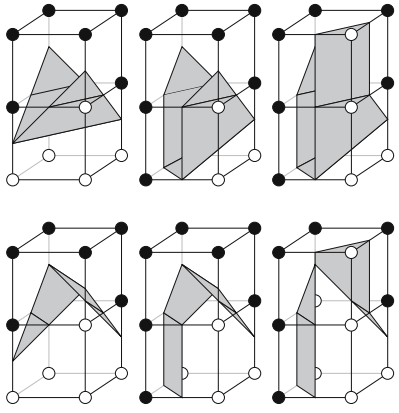

Fig. 11. The two different triangulations of the three ambiguous cell combinations

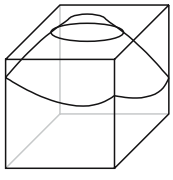

Fig. 12. Surface S intersects a face of the regular grid in a closed loop

to this face must be parametrizable in the direction perpendicular to this face. Also, the "bubble" does not leave the cube on the opposite face, since that would require the gradient to change more than $\pi/2$ (see also Fig. 3). By linear interpolation in the direction of parametrizability, we can continuously flatten the bubble towards the face, and push it through this face, thereby removing the intersection loop without leaving the isotopy class of the surface.

As in the two-dimensional case we still have to remove the constraints on the function: the surface passing through vertices, and multiple edge intersections. The case where the surface passes through vertices of the grid can be handled as with implicit curves, i.e., by considering F to be strictly positive whenever $F \geq 0$.

Now assume that S intersects an edge e of the grid more than once, say at two consecutive points α and β (Fig. 13). Since F changes sign at α and at β, there is a point p between α and β where $\nabla F(p)$ is perpendicular to e. This gradient lies inside the gradient cone of all four cubes adjacent to e (due to the interval constraint all gradients over a cell lie within a cone with top angle $\pi/2$). Therefore, in the union of these four cubes, the surface is parametrizable in the direction of $\nabla F(p)$. Each of the two half-planes bounded by e in the direction of $\nabla F(p)$ intersects one of the cubes. Since $\nabla F(p)$ lies inside the cone of gradients of this cube, the projection of $\nabla F(x)$ does not change by more than $\pi/2$ when x ranges over the intersection of this half-plane with the cube. Therefore in one of these two half-planes, the intersection with S consists of a connected curve between α and β, that does not leave the cube. By linear interpolation in direction $\nabla F(p)$ between this curve and e we can locally deform the surface, and remove the pair α, β.

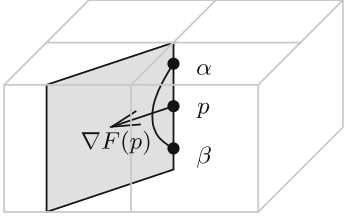

Fig. 13. Surface S intersects an edge of the regular grid more than once

After removing all pairs of edge intersections we have continuously deformed the surface to one with only single edge intersections, for which the grid triangulation was shown to be correct.

We conclude that the triangulation determined by the regular grid is isotopic to the implicit surface.

6 Octrees

In the previous section we have shown that a regular grid satisfying an interval constraint can be used to create an

isotopic approximation of an implicit surface. The regular grid approach usually results in a large number of mesh triangles. An octree-based approximation has the advantage that we only produce a detailed mesh where a higher level of detail is required. In this section we will show how a balanced octree can be used to create an isotopic mesh. Recall that an octree is balanced if neighboring leaves differ at most one level. Since we can subdivide the octree approximation to an approximation on a complete octree without leaving the isotopy class of the surface, we have isotopy between the surface and the octree approximation, similarly to the two-dimensional case.

Overview. The subdivision process is identical to the two-dimensional case. Starting with an octree initialized to a bounding box B, we subdivide the leaves until for each leaf C we have

$$0 \notin \square F(C) \vee \langle \square \nabla F(C), \square \nabla F(C) \rangle > 0.$$

After the subdivision process, we balance the octree. Once we have constructed the octree, we can construct the approximation. First we put vertices at the center of each edge of the octree that has opposite function signs at its endpoints. Then, for each face of the octree we connect these vertices pairwise with segments. For each cell of the octree, these segments form closed chains on the boundary of the cell. If we have two nested loops, we connect these two loops with triangles, resulting in an annulus (this case is shown in Fig. 14). Note that due to parametrizability we can consider the projection of the approximation. This way, it is clear that we cannot have more than two nested loops. In all other cases, we close each loop with triangles, constructing a topological disk for each closed chain of edges (Fig. 15). Section 7 shows a more practical way to create the local triangle patches.

To make sure that further subdivision does not change the isotopy class of the surface, we have to be careful when connecting the vertices. The subdivision can push the approximation up and down in the parametrizable direction. We shall assume a vertical parametrizable direction. For the top and bottom faces, it does not matter how we connect the vertices. Since the parametrizable direction is perpendicular to these faces, changing the connections corresponds to moving the surface slightly up or down through these faces. In the union of the adjacent cubes the isotopy does not change. This is similar to the regular grid case in Fig. 11. The intersection curve in the common face changes, but the isotopy class of the surface inside the union of the two adjacent cells stays the same.

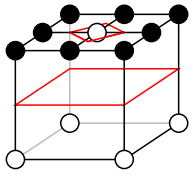

Fig. 14. The top face indicates the only case where two "nested" loops bounding a cylinder occur, resulting in an annulus on the approximating surface. All other configurations correspond to an octree cell contributing triangulated disks to the approximating surface

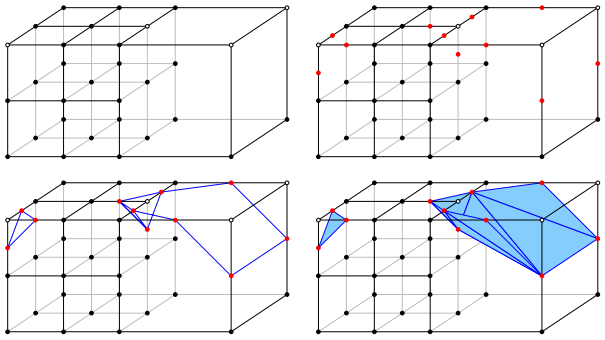

Fig. 15. Meshing an octree by creating vertices, connecting these by line segments, and creating patches for segment loops

For the faces of the octree along the side of a cell, the intersection of the surface with such a face is still parametrizable, but the $\pi/2$ bound on the variation of the gradient restricted to the face does not hold. The intersection curve can therefore pierce through two opposite edges (cf. Fig. 3). This prevents us from uniquely determining the topology of the intersection curve. However, a change in topology of the intersection curve corresponds to moving the surface slightly sideways through such a side face. Although the intersection curve changes, the isotopy class of the surface itself is not affected.

Mesh construction. To construct an isotopic approximation, we first construct mesh vertices on edges exhibiting a sign change. Then we examine these newly created mesh vertices on the boundary of an octree face. If the vertex configuration cannot correspond to a parametrizable intersection curve, we connect adjacent vertices pairwise, starting at an arbitrary vertex. Both ways of pairwise connecting give rise to the same isotopy class of the approximating surface. If the vertex configuration does correspond to a parametrizable intersection curve, we construct a parametrizable piecewise linear approximation, which is unique for a fixed parametrizable direction. Some examples are shown in Fig. 16. For a more straightforward and nonambiguous algorithm to connect the mesh vertices, we refer to Sect. 7. To show that this algorithm also constructs an isotopic mesh, we will subdivide the octree to a complete tree, similar to the two-dimensional case. The subdivision of an octree cell is split into several steps: first we add an octree vertex at the center of an edge (if necessary), then at the center of faces, and finally we add an octree vertex at the cell center.

Vertex on edge. If an octree edge has opposite function signs at its endpoints, adding a center octree vertex to that edge moves a mesh vertex from the center of that edge to one-fourth or three-fourths of that edge. If the function signs are the same, adding the center octree vertex

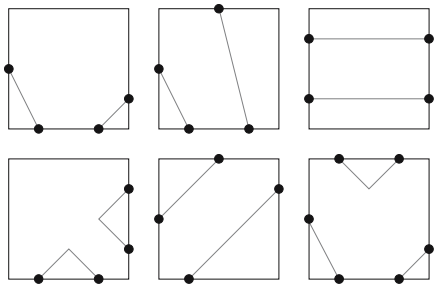

Fig. 16. Examples of how to connect the vertices along the faces of the octree. The top row corresponds to potentially parametrizable intersections. Note that for example the middle figure could be parametrizable, since the vertices do not represent the exact intersection point; the top vertex may be moved to the right, resulting in a parametrizable approximation. The bottom row is not parametrizable and can be connected arbitrarily, by joining subsequent vertices starting at an arbitrary mesh vertex. In the figures we show one of the two possible edge configurations

can give rise to two new mesh vertices. However, the octree cells adjacent to this edge are at least as large as the edge itself, otherwise the octree vertex would have existed already. Therefore we can use the same argument with which we removed double edge intersections in Sect. 5 (cf. Fig. 12). That is, the isotopy of the mesh does not depend on the function sign at the center of the edge.

Vertex on face. Once all edges of a leaf are subdivided, we can add octree vertices at the centers of the faces. In Fig. 17 we have subdivided the face of the octree and added diagonals in the four quadrants by connecting the center vertex to the four corners of the face. Due to the choice of the diagonals, meshing the resulting triangles connects adjacent mesh vertices, similar to the algorithm explained earlier. With the chosen diagonals adjacent mesh vertices are connected, where the sign of the center vertex determines which of the two connection schemes is used. The resulting intersection curves depend on our particular triangulation or choice of diagonals. However, choosing other diagonals corresponds to other ways to resolve ambiguous quadrants. This is identical to the ambiguous faces in the regular grid approximation from Sect. 5 (Fig. 11). Although we can construct various configurations of intersection curves, they all correspond to surfaces with the same isotopy class.

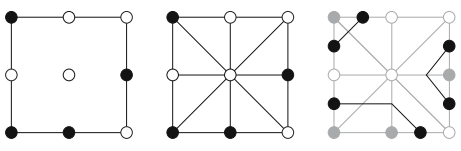

Fig. 17. Meshing a face after adding a center vertex

Both the regular grid method and the octree algorithm therefore connect adjacent mesh vertices of the face. The remaining problem is that the sign of the center vertex determines which adjacent mesh vertices are connected.

For (possibly) parametrizable faces we constructed a parametrizable approximation. Along the line through the center vertex we may shift the intersection curve in the parametrizable direction. By moving it over the center vertex we can change the function sign at this center vertex without changing the isotopy.

For nonparametrizable faces we can assume there are at least four mesh vertices along the boundary, otherwise there is only a single way to connect them. We have to show that both ways of connecting adjacent mesh vertices result in isotopic approximations. Due to the gradient variation constraint, the part of the surface above or below the convex hull of the vertices stays close to the shared octree face, similar to the bubble in Sect. 5 (Fig. 12). Stated otherwise, if the surface passes through four points of the face, the bound on the curvature locally keeps the surface close to that face. It does not leave the adjacent cells, and therefore we may move the surface up and down slightly, resulting in different ways to connect the vertices inside the octree face. Again, different topology of intersection curves does not influence the isotopy class of the surface inside the union of the adjacent octree leaves.

Vertex in cell. Finally we have to add an octree vertex at the cell center. Since the function is increasing in the parametrizable direction, this can shift a mesh vertex in that direction, but it cannot introduce new mesh vertices. Summarizing, we now have shown that the octree together with the function signs at its vertices fully determines the isotopy of the surface. The signs at the newly created octree vertices have no influence on the isotopy.

Now we can use the regular grid approximation for the eight subcubes of the octree cell, which is isotopic to the surface. Since the approximation is parametrizable (except for vertical triangles), it is easy to see that connected patches usually form topological disks. The only way to create a hole is when the projection of that hole is a loop around the center vertex, as in Fig. 14. This is exactly the case for which we create an annulus. Therefore the regular grid approximation for the eight subcubes is isotopic to the grid created by the octree algorithm.

7 Tetrahedra

In this section we introduce a meshing algorithm based on tetrahedral subdivision. For the proof of isotopic correctness we required an octree-based mesh construction, since interval arithmetic works on axis-aligned boxes. Now that we have shown that the resulting mesh is isotopically correct, we can facilitate the mesh generation, by subdividing

the octree leaves into tetrahedra. We will show that meshing these tetrahedra results in a mesh isotopic to the octree mesh as described in the previous section. The number of mesh triangles will be slightly larger, but tetrahedrization results in a more straightforward algorithm and better interpolation of the mesh vertices. In particular, we do not need to consider the special case of annulus construction, and we do not have any ambiguities to resolve.

The sides of each face of the octree can consist of either one octree edge or of two octree edges in case of a subdivided side. We walk around an octree face in a fixed direction, for example counterclockwise as seen from the positive axis perpendicular to that face, and starting at the corner vertex with largest coordinates. By checking whether the four sides are subdivided, this results in a unique sign pattern. We then triangulate the face according to the table in Fig. 18.

Once all faces of an octree cell are triangulated, we can tetrahedrize the cell by constructing a tetrahedron for each triangle, by connecting it to the center of the cube (Fig. 19). Since the triangulation of octree faces is determined uniquely, this results in a tetrahedrization of the entire octree.

To generate the mesh, we construct at most two triangles for each tetrahedron. Note that unlike cubes, tetrahedra do not have ambiguous cases for meshing. If one of the four vertices has an opposite sign we construct one triangle, if two vertices have the opposite sign of the other two, we construct two triangles forming a 4-gon (Fig. 20).

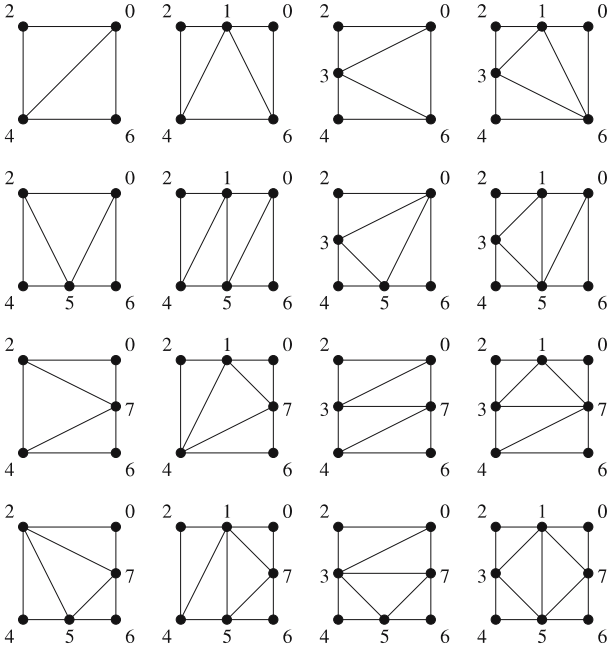

Fig. 18. Triangulation of octree faces. The eight possible vertex positions are labeled 0 through 7 in counterclockwise direction around the positive axis

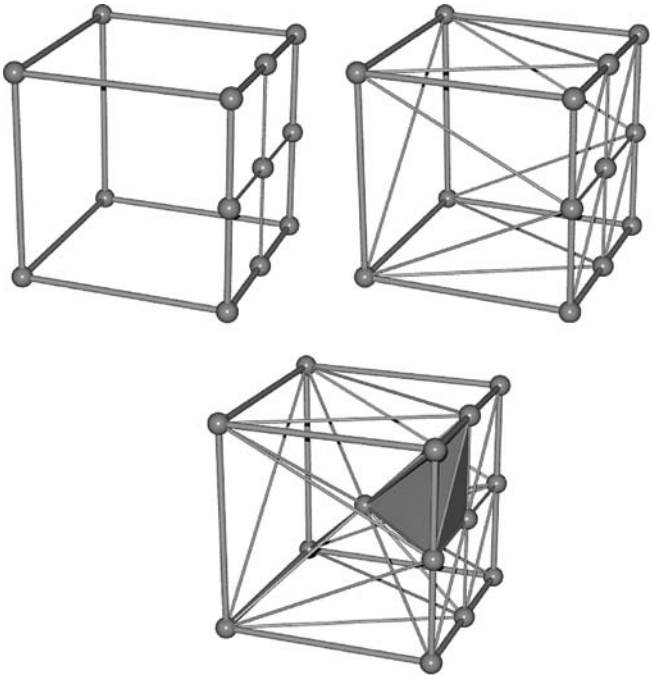

Fig. 19. Tetrahedrization of a cell by triangulating its boundary. One of the 22 tetrahedra is shown in the bottom figure

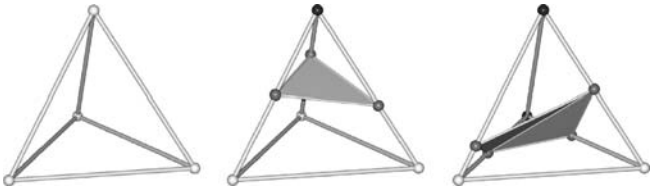

Fig. 20. Meshing the tetrahedra

By considering the projection in the parametrizable direction it is easy to see that the projection of the mesh for all tetrahedra in a given octree cell, is isotopic to the projection of the mesh produced by the previously given algorithm. The mesh from the tetrahedral algorithm is therefore also isotopic to the implicit surface.

8 Results

Figures 21 and 22 show two examples of the curve approximation algorithm. The quadtree is also shown in the images. Both examples took 0.04 seconds to compute.

In Figs. 23, 25, 26 and 27, four examples of implicit surface approximation are shown. Note that due to the placement of vertices at centers of edges, flat triangles are generated. However, the structure of the resulting mesh

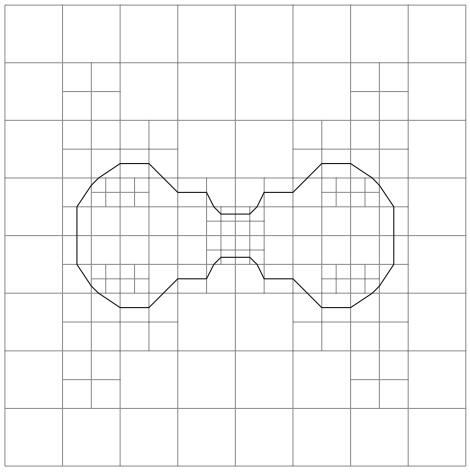

Fig. 21. Implicit curve approximation of $f(x, y) = x^2(1-x)(1+x) - y^2 + 0.01$

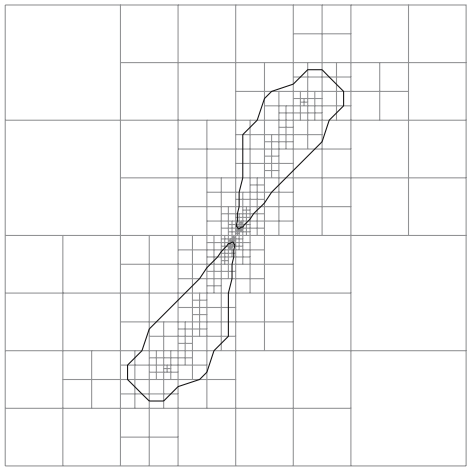

Fig. 22. Implicit curve approximation of $f(x, y) = x^2 - xy + y^4 + 0.0001$

forms a simplicial complex. Figure 24 shows the tangle cube meshed with tetrahedral subdivision and linear interpolation of the mesh vertices. The algorithm was implemented in C++. The table below shows the number of leaves in the octree before and after subdivision, the number of triangles in the mesh, and the running time in seconds on a Pentium 667 MHz, running Linux.

Surface	Octree	Balanced	Triangles	Time
Tangle	24 648	24 816	8704	0.8
Chair	232 072	233 402	43 014	6.2
Bear	497 596	688 794	366 746	113.0
Nonalg.	29 275	40 293	24 612	2.0

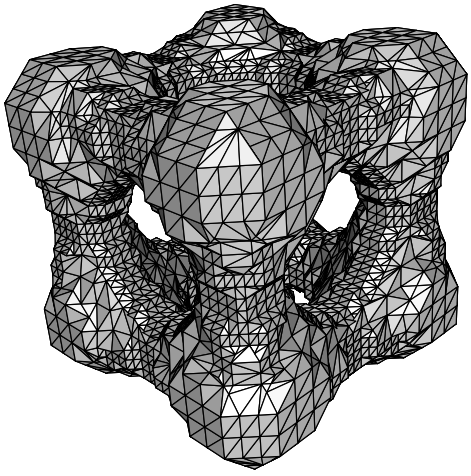

Fig. 23. Tangle cube $f(x, y, z) = x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 10$

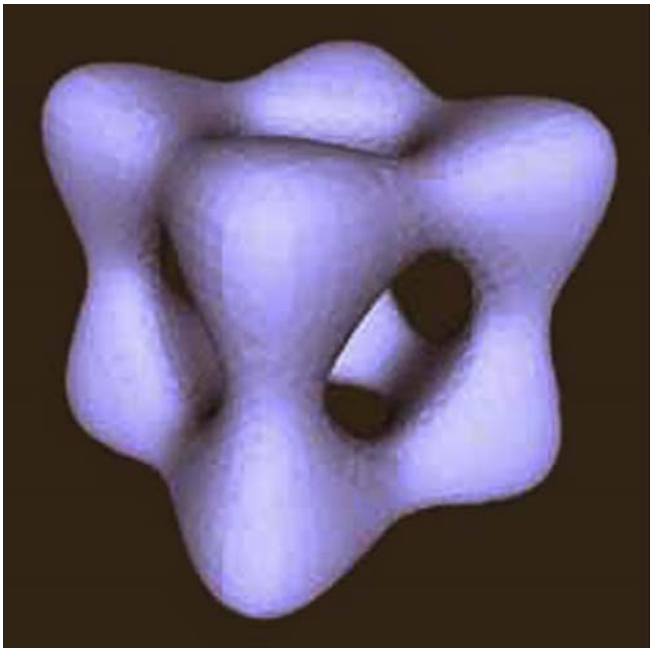

Fig. 24. Tangle cube using tetrahedral subdivision

9 Conclusion

We have presented a simple and fast algorithm for meshing implicit surfaces, that is among the first to guarantee isotopy between the mesh and the surface. The current implementation does not try to create a good approximation in terms of Hausdorff distance. An open problem remains how to improve the mesh quality (for example lower

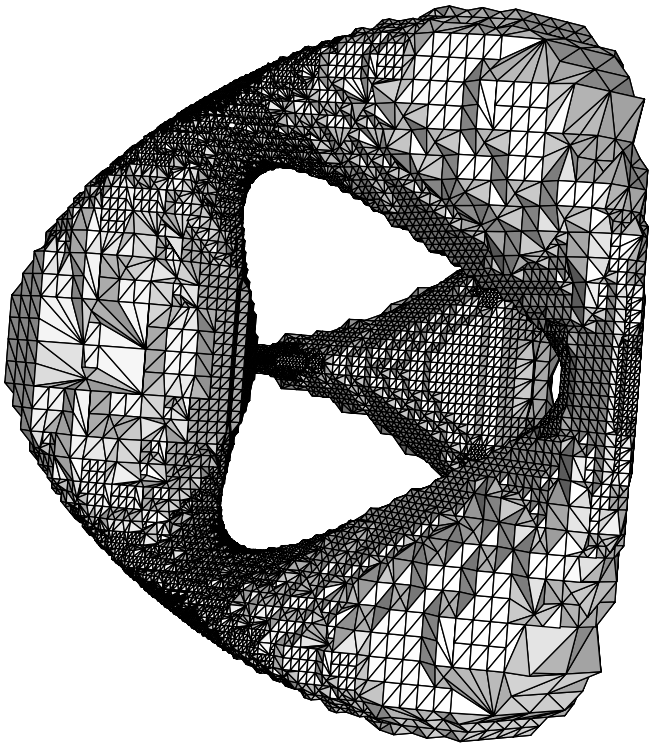

Fig. 25. Chair $f(x, y, z) = (x^2 + y^2 + z^2 - ak^2)^2 - b((z-k)^2 - 2x^2)((z+k)^2 - 2y^2)$, for $k = 5$, $a = 0.95$ and $b = 0.8$

bounds on the angles mesh triangles) without loosing isotopy. Other improvements to the algorithm are given below.

Improvements. Both algorithms use balanced trees. This balancing facilitates both the algorithm and the proof. We believe it is not necessary for an isotopic result. Without balancing, the approximation for the leaves becomes more complicated and with the tetrahedral subdivision the individual tetrahedra could become very elongated, resulting in numerical problems. However, removal of the balancing would produce a smaller approximation in the number of mesh triangles.

The octree-based approach allows for local updates if S changes only locally, for example when using blobs or metaballs. These models are constructed of spheres, blended by adding Gaussian-like functions centered around points in space. Adding a single metaball does not change the implicit function outside the influence region of that metaball. Therefore, we only need to remesh the part of the octree intersecting the new metaball.

Although the resulting approximation is isotopic to S, it does not have to be close in terms of Hausdorff distance. To get a closer approximation the algorithm could be extended by subdividing cells where $0 \in \square F(C)$ to a given minimal level, before starting the subdivision process (recall that the surface does not leave the neighboring cell).

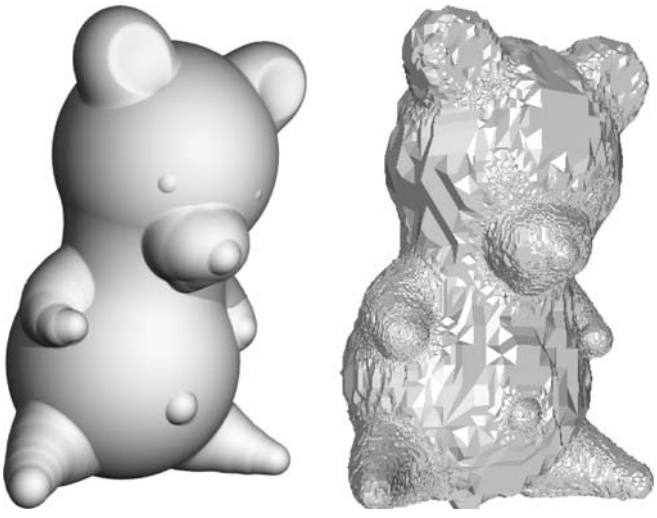

Fig. 26. An implicit teddy bear together with its approximation. The bear consists of 48 metaballs

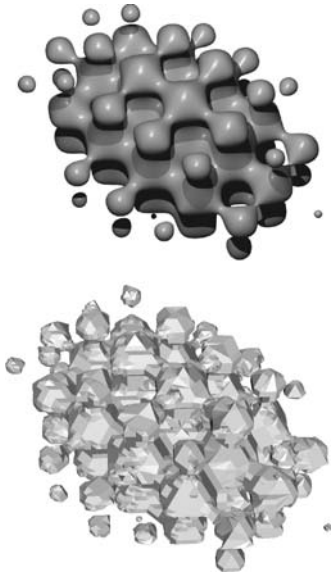

Fig. 27. The nonalgebraic surface $-0.4(\sin(5x) + \sin(5y) + \cos(5z)) + 0.1x^2 + 0.3y^2 + 0.2z^2 - 0.5$

Note that if the gradient variation bound holds for the cell, it also holds for its subdivisions. We can therefore increase the resolution of the mesh, without performing extra interval tests. Due to the bound on gradient variation, a smaller cell size also results in a better approximation of normals (see Fig. 24).

For curves or surfaces containing singularities, the subdivision does not terminate. By using a maximum depth for the subdivision level we can mesh these curves and surfaces. Although the isotopy close to a singularity is not guaranteed, this results in an isotopic approximation outside arbitrarily small bounding boxes around the singular

points. This can also be used to speed up the approximation, since the algorithm does not continue to refine the tree in difficult areas. Although isotopy is not guaranteed, we can identify the leaves where the interval condition holds, that is where the approximation is isotopically correct. The remaining (arbitrarily small) leaves where we are not yet sure of the topology could be identified, for example by triangulating these areas in a different color. This way the user cannot only easily identify where the problematic areas are, but can also decide to refine locally until the isotopy holds or the approximation is close enough. Also, other methods can be used to examine the behavior of the surface inside these cells, e.g., algebraic methods.

The algorithm can also be used for unbounded curves and surfaces. Recall that the undetected parts of the curves and surfaces cannot pierce through opposite edges or faces of a cell. By making the initial bounding box larger than the area of interest, and discarding the outer cells afterwards, we can construct an isotopic approximation for an unbounded manifold within a given box.

For general implicit surfaces it is difficult if not impossible to arrive at a complexity bound. Such a bound could be based on, for example, the local feature size. However, the complexity does not only depend on the surface, but also on the particular implicit function.

The method introduced in [6] meshes implicit k-Lipschitz surfaces, which are not necessarily C^1. Our method does not apply to this larger class of surfaces. However, it seems feasible that our condition on bounded variation of surface normals can be relaxed in the sense that we require bounded variation of *normal cones* of the surface. Here a normal cone is the collection of all unit vectors that are perpendicular to some locally supporting plane of the surface. In this way we obtain a class of surfaces that is larger than the set of C^1-surfaces, but smaller than the class of k-Lipschitz surfaces. However, this extension goes beyond the scope of the current paper.

References

1. Akkouche, S., Galin, E.: Adaptive implicit surface polygonization using marching triangles. In: D. Duke, R. Scopigno (eds.) Computer Graphics Forum, vol. 20(2), pp. 67–80. Blackwell, Berlin (2001)
2. Alberti, L., Comte, G., Mourrain, B.: Meshing implicit algebraic surfaces: the smooth case. In: L.L. Schumaker, M. Maehlen, K. Morken (eds.) Mathematical Methods for Curves and Surfaces: Tromso'04, pp. 11–26. Nashboro, Gatlinburg (2005)
3. Bloomenthal, J.: Polygonization of implicit surfaces. Comput. Aided Geom. Des. **5**, 341–355 (1988)
4. Boissonnat, J., Cohen-Steiner, D., Vegter, G.: Isotopic implicit surface meshing. In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, pp. 301–309, Chicago (2004)
5. Boissonnat, J., Oudot, S.: Provably good sampling and meshing of surfaces. Graph. Models **67**, 405–451 (2005)
6. Boissonnat, J.D., Oudot, S.: Provably good sampling and meshing of Lipschitz surfaces. In: SCG '06: Proceedings of the 22nd Annual Symposium on Computational Geometry, pp. 337–346. ACM, New York (2006)
7. Bottino, A., Nuij, W., van Overveld, K.: How to shrinkwrap through a critical point: an algorithm for the adaptive triangulation of isosurfaces with arbitrary topology. In: Proceedings of Implicit Surfaces, pp. 55–72 (1996)
8. Bremer, D., Hughes, J.F.: Rapid approximate silhouette rendering of implicit surfaces. In: Proceedings of Implicit Surfaces, pp. 155–164 (1998)
9. Cheng, S., Dey, T., Ramos, E., Ray, T.: Sampling and meshing a surface with guaranteed topology and geometry. In: Proceedings of the Symposium on Computational Geometry, pp. 280–289 (2004)
10. Chernyaev, E.V.: Marching cubes 33: construction of topologically correct isosurfaces. Tech. Rep. CERN CN 95–17 (1995)
11. Hartmann, E.: A marching method for the triangulation of surfaces. Vis. Comput. **14**(3), 95–108 (1998)
12. Lerch, M., Tischler, G., von Gudenberg, J.W., Hofschuster, W., Krämer, W.: Filib++ interval library. http://www.math.uni-wuppertal.de/wrswt/software/filib.html. Cited (2005)
13. Lopes, H., Oliveira, J.B., de Figueiredo, L.H.: Robust adaptive polygonal approximation of implicit curves. Comput. Graph. **26**(6), 841–852 (2002)
14. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 163–169. ACM, New York (1987)
15. Plantinga, S., Vegter, G.: Contour generators of evolving implicit surfaces. In: Proceedings of the 8th ACM Symposium on Solid Modeling and Applications, pp. 23–32. ACM, New York (2003)
16. Plantinga, S., Vegter, G.: Isotopic approximation of implicit curves and surfaces. In: Proceedings of the Symposium on Geometry Processing, pp. 245–254 (2004)
17. Snyder, J.M.: Interval analysis for computer graphics. Comput. Graph. **26**(2), 121–130 (1992)
18. Stander, B.T., Hart, J.C.: Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, pp. 279–286. ACM Press/Addison-Wesley, New York (1997)
19. van Overveld, C., Wywill, B.: Shrinkwrap: an adaptive algorithm for polygonizing an implicit surface. Tech. Rep. 93/514/19, University of Calgary (1993)

SIMON PLANTINGA is a researcher at the Institute of Mathematics and Computer Science of the University of Groningen, The Netherlands. His research interests include computational geometry and implicit surfaces.

GERT VEGTER is professor of Computer Science at the Institute of Mathematics and Computer Science of the University of Groningen, The Netherlands. His research interests are computational geometry, differential geometry and dynamical systems.