Non-local Isotopic Approximation of Nonsingular Surfaces

Long Lin, Chee Yap and Jihun Yu
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012 USA
{llin,yap,jihun}@cs.nyu.edu

December 3, 2011

Abstract

ABSTRACT: We consider the problem of computing isotopic approximations of nonsingular surfaces which are implicitly represented by equations of the form $f(x, y, z) = 0$. This mesh generation problem has seen much recent progress. We focus on methods based on domain subdivision using numerical primitives because of their practical adaptive complexity. Previously, Snyder (1992) and Plantinga-Vegter (2004) have introduced techniques based on parametrizability and non-local isotopy, respectively. In our previous work (SoCG 2009), we synthesized these two techniques into a highly efficient and practical algorithm for curves. In this paper, we extend our approach to surfaces. The extension is by no means routine, as the correctness arguments and analysis are considerably more complex. Unlike the 2-D case, a new phenomenon arises in which local rules for constructing surfaces are no longer sufficient.

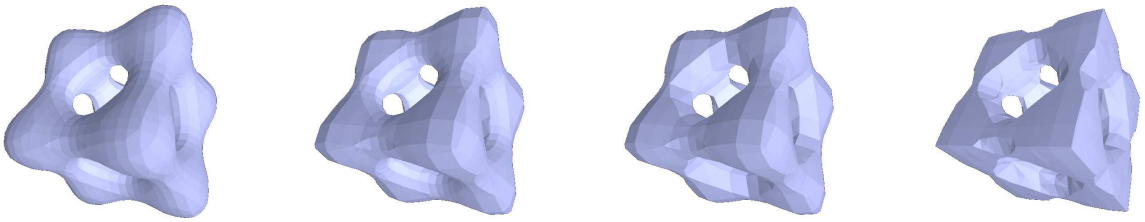
We treat two important extensions, to exploit anisotropic subdivision and to allow arbitrary geometry for the region-of-interest (ROI). Anisotropy means that we allow boxes to be split into 2, 4 or 8 children which are rectangular boxes with bounded aspect ratio. Using ROI allows our algorithms to be extremely "local", and anisotropy increases their adaptivity.

Our algorithms are relatively easy to implement, as the underlying primitives are based on interval arithmetic and exact BigFloat numbers. We report on very encouraging preliminary experimental results.

Key Words: Mesh Generation, Surface Approximation, Isotopy, Parametrizability, Subdivision Algorithms, Interval Methods, Topological Correctness, Exact Numerical Algorithms.

1 Introduction

A basic problem in areas such as physics simulation, computer graphics and geometric modeling is that of computing approximations of curves and surfaces from implicit definitions. Typically, the surface is represented by an equation, $f(x, y, z) = 0$ as illustrated in Figure 1. We assume the approximation is a triangulated surface, also known as a **mesh**. The recent book of Boissonnat and Teillaud [5] provides an algorithmic perspective for this general area; chapter 6 in particular is a survey of meshing algorithms.



(a) Cxyze (b) PV (c) Cxyz (d) Rect-2

Figure 1: Approximation of tangled cube $f(x, y, z) = x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 = -10$.

The approximate surface or mesh must satisfy two basic requirements: topological correctness and geometric accuracy. For instance, in Figure 1(c) is produced by our algorithm with only topological correctness as stopping criterion. For some applications, this is sufficient. But if one desires geometric accuracy as well, this can be further refined as in Figure 1(a), where the error bound is $\varepsilon = 0.25$. Generally, the problem of ensuring topologically correctness is more challenging than refinement.

Formally, the **mesh generation problem** ("meshing problem" for short) is this: *given a region-of-interest (ROI) $R_0 \subseteq \mathbb{R}^3$, an error bound $\varepsilon > 0$, a surface S implicitly represented by an equation $f(x, y, z) = 0$, to find a piecewise linear ε-approximation G of S restricted to R_0.*

Geometric accuracy in G means that the Hausdorff distance between G and $S \cap R_0$ is at most ε. Topological correctness means the surface G should be isotopic to S in the interior of R_0, and also on the boundary ∂R_0; we denote this by "$G \simeq S$ (**mod** R_0)". We focus on guaranteeing topological correctness by means of numerical techniques. Numerical methods have many advantages: they tend to have adaptive complexity, are efficient in practice and easy to implement. Numerical methods are more general than algebraic ones since hey are applicable to non-algebraic functions such as frequently arise in mathematical analysis. But numerical methods traditionally do not offer topological guarantees, and so this is our main challenge.

Throughout this paper, we fix the function $f : \mathbb{R}^3 \to \mathbb{R}$, the surface $S := f^{-1}(0)$ and region-of-interest $R_0 \subseteq \mathbb{R}^3$. The region R_0 is a nice region (see below) represented by an octree, and f is nonsingular in R_0. Unless otherwise noted, we assume $\epsilon = \infty$ (i.e., we focus on isotopy, with no concern for geometric accuracy). For the algorithms of this paper, it is easy refine to any desired ϵ once we have the correct isotopy.

¶1. Subdivision Algorithms. Our main algorithmic paradigm is **(domain) subdivision** where an initial axes-parallel box $B_0 \subseteq \mathbb{R}^3$ is repeated subdivided into smaller boxes, forming an octree T rooted at B_0. Each non-leaf of T can have 2, 4 or 8 children, corresponding to half-, quarter- or full-splits of boxes into congruent subboxes. The leaves of T provide a partition of B_0 into boxes, and to subdivide T means to split its leaves. All algorithms in this paper are viewed as instances of the following:

Generic Subdivision Algorithm:
INPUT: AN OCTREE T REPRESENTING A REGION B_0
I. Subdivision Phase:
    Keep subdividing T until some stopping criterion holds.
II. Refinement Phase:
    Further subdivide T until some refinement criterion holds.
III. Construction Phase:
    Construct the approximation G from the refined tree T.

The conceptual question is: *what kind of stopping and refinement criteria do we need in order to ensure that the Construction Phase has sufficient information to construct an isotopic approximation G?* This question is ill-formed unless we constrain the Construction Phase. The well-known Marching Cubes [15] gives us a clue: for each leaf box B, the Marching Cubes algorithm computes a small surface patch $G_B \subseteq B$ based *only* on the signs of f at the corners of B. This is $O(1)$ work per leaf, and G is defined to be union of all these patches G_B. Such a Construction Phase is said to be **MC-like** ("Marching Cubes like"). But it is well-known that the Marching Cubes could not ensure correct isotopy. The achievement of Plantinga & Vegter (PV) [18] is that, by using the "small normal variation predicate", they could ensure correct isotopy with a MC-like construction. Theirs is the first topologically correct algorithm for meshing of nonsingular surfaces based on numerical primitives. In contrast, the construction phase in Snyder's algorithm [24] is not MC-like, but requires highly nontrivial processing (e.g., root isolation). In [14, 13], we characterize the PV approach as exploiting "non-local isotopy". We show that the stopping criterion of PV can be weakened to the parametrizability predicate of Snyder, leading to greatly improved efficiency. Our previous result was only for curves; in this paper, we will extend it to surfaces. As we shall see, the extension to surfaces is far from routine, requiring new ideas in the algorithm as well as in its correctness proofs. For instance, a new phenomenon arises in the Construction Phase in which local rules are no longer sufficient.

The algorithms in this line of research are very practical for two reasons: first, it is based on the easily implementable subdivision paradigm. Second, all our primitives are explicitly numerical (no hidden implementation gaps). We stress this point because many exact algorithms in the literature have primitives that are impractical for exact implementation. The numerical primitives are based on two simple foundations: (a) interval methods [17, 19], and (b) BigFloats, some software implementation of dyadic numbers. Moreover, machine arithmetic can be exploited in two ways: first, it can replace BigFloats when machine precision suffices (taking care to detect overflows which indicate the need for higher precision). In fact all the examples in this paper are run at machine precision. Second, they can be used as filters to speed up BigFloats. See [14] for further discussion. We have implemented our algorithm and preliminary evidence suggests that our algorithms can be much more efficient than previous algorithms.

¶2. Our Contribution and Overview of Paper. Our general contribution is the further development of non-local isotopy analysis. In Section 2, we review this concept. Our main technical contribution is a new exact, efficient and practical algorithm for isotopic surface approximation. We describe a sequence of three increasingly sophisticated algorithms: **Regularized Cxyz** (Section 4), **Balanced Cxyz** (Section 5), and **Rectangular Cxyz** (Section 6). Each has independent interest, but is also useful in our development: we reduce the correctness of each algorithm to that of the previous one. Because of space limitation, we only briefly touch on another important topic, allowing input region-of-interest (ROI) with arbitrary geometry as represented by a suitable octree. Section 7 contains our experimental results, and we conclude in Section 8. All the proofs and further experimental data are available in the thesis of Lin [13] and may be downloaded from [9]

2 Related Work

We broadly classify approaches to mesh generation into three categories: algebraic, geometric, and numerical. Algebraic approaches [1, 22, 8, 23], exploit tools such as cylindrical algebraic decomposition (CAD), resultants, and manipulation of algebraic numbers ([5, Chapter 3] reviews these technique). These tools are exact, but the algorithms may be slow with non-adaptive complexity. A promising direction to remedy this is to combine symbolic with numeric methods [10]. The geometric approaches [25, 3, 7, 4] postulate some abstract computational model where geometric primitives such as ray shooting are available, and algorithms based on these primitives are constructed. Implementing these abstract models can be an issue (e.g., ray shooting returns points with algebraic coordinates, which may be unsuitable for implementation). The numerical approaches [15, 18, 16, 20, 26, 27] are based on numerical approximations, evaluation and derivatives of function, and interval methods. It is the most pragmatic of the three approaches. Its advantages include having adaptive and local complexity, and relative ease of implementation. Guaranteeing topological correctness is the traditional weakness of this approach. The non-local isotopy idea can be exploited in other applica-

tions: recently we constructed a new subdivision method for complex root isolation [21] that has proved very efficient [11, 12]. To motivate the general approach of our paper, we review four particular subdivision algorithms: Marching Cubes [15], Snyder's Algorithm [24], Plantinga & Vegter's (PV) Algorithm [18], and our Cxy Algorithm (in 2-D) [14]. We use the framework of the Generic Subdivision Algorithm in ¶1.

¶3. Marching Cubes. Marching Cubes is one of the most popular subdivision algorithms for surface reconstruction. The stopping criterion for its Subdivision Phase is "box has width $< \epsilon$" for some arbitrary $\epsilon > 0$. In the Construction Phase, we determine the sign of the function f at the corners of each leaf box B of T. Up to symmetry and interchange of signs, the possible **sign types** are given in Figure 2.

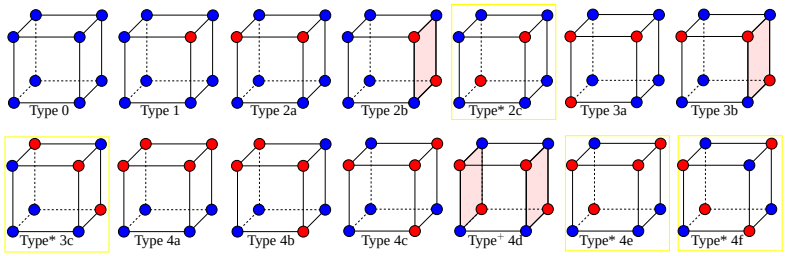


Figure 2: 14 Sign Types of f at box corners: only 10 arise under C_{xyz}

If an edge of B has different signs at its two corners, we introduce a **vertex** in the middle of the edge. We then connect pairs of vertices on faces of B by **arcs**. Some possibilities for these **arc types** are illustrated in Figure 3 (our figure shows only those types that can arise in our algorithm). Note that Sign Types 2b, 3b and 4d each gives rise to two arc types, and they are topologically distinct. This "ambiguity" will be one of our main correctness concerns. Once the arcs are fixed, we can introduce a triangulated surface patch G_B in B such that G_B intersects boundary of B with the given arc type. The union $G = \bigcup_B G_B$ of these patches constitutes an approximation of S. Fortunately, the isotopy type of G_B is uniquely determined by the chosen arcs:

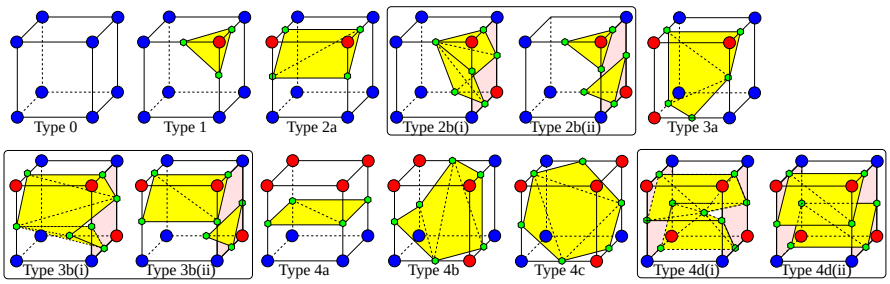


Figure 3: The 13 Arc Types under C_{xyz} Predicate.

LEMMA 1. *Each of the 13 arc types in Figure 3 uniquely determines the isotopy of the surface patch in a box.*

This lemma follows by case analysis. Although the Marching Cubes does not guarantee topology, we stress that the Marching Cubes is very useful and widely-used.

¶4. Parametrizability of Snyder. A key paper towards ensuring correct topology in subdivision algorithms is Snyder [24]. He introduced interval methods to determine the correct topology within each subdivision box B. Snyder's stopping criterion is "$S \cap B$ is parametrizable". This means that surface patch $S \cap B$ is the graph of some function $g(i, j)$ in two coordinate directions $i, j \in \{x, y, z\}$. This condition can be detected using interval arithmetic: we call this the $C_{xyz}(B)$ predicate below. Snyder is then able to construct a triangulated surface patch $G_B \subseteq B$ with the property $G_B \simeq S$ (**mod** B). His algorithm is recursive in dimension: to construct G_B, recursively solve the 2-D problem of computing the topology of $S \cap F$ on each face F of B. In turn, this requires solving the 1-D problem of root isolation along the edges of F. There are two issues. First, the algorithm may not terminate if S intersects the boundary of B tangentially at isolated points [2, p. 195]. Second, G_B can have arbitrary combinatorial complexity, and thus is not MC-like.

¶5. Non-local Isotopy of Plantinga & Vegter The second key paper is from Plantinga & Vegter [18]: instead of parametrizability, they introduce two simple criteria for termination of subdivision: the **exclusion predicate** $C_0(B)$ and the **small normal variation predicate** $C_1(B)$ (see below for the definitions of C_0 and

C_1). The predicate $C_1(B)$ implies that the angle between two gradient vectors of f in B is less than 90 degrees, and in particular it implies that $S \cap B$ is parametrizable. Snyder constructs the **local isotopy** of the surface in each box B. In a radical departure from Snyder, they no longer require that G_B be isotopic to $S \cap B$. Remarkably, this approach also solves the two issues of Snyder. We view non-local isotopy very favorably because enforcing local isotopy is considered wasteful (after all, subdivision boxes are artifacts of the algorithm, not inherent in topology of S).

¶6. Our Synthesis. Our paper [14] is a synthesis of the parametrizability approach of Snyder with the non-local isotopy of PV. We only treated curves. Basically, we want to run the PV algorithm but replacing the C_1 predicate with parametrizability. It turns out that this is justifiable provided we take care to disambiguate certain configurations by subdivisions. Our motivation is that using C_1 is an overkill for isotopy (though C_1 has other uses, including controlling normal deviation and refinement). Experiments confirm our expectation: our synthesis is more efficient than either approach separately.

3 Preliminaries

For any set $S \subseteq \mathbb{R}$, let $\square S$ denote the set of all closed intervals with endpoints in S. We mainly use $S = \mathbb{R}$ and $S = \mathbb{F}$ where $\mathbb{F} := \{m2^n : m, n \in \mathbb{Z}\}$ denote the set of dyadic numbers (BigFloats). A **box** (or d-box) is any element of $\square\mathbb{R}^d$ $(= (\square\mathbb{R})^d)$. Usually, $d = 1, 2, 3$. If $f : \mathbb{R}^d \to \mathbb{R}$ is any function, then a function of the form $\square f : \square\mathbb{F}^d \to \square\mathbb{F}$ is called a **box function** for f if for all $B, B_i \in \square\mathbb{F}$, we have (1) (inclusion) $f(B) \subseteq \square f(B)$, and (2) (convergence) if $\lim_{i\to\infty} B_i = p \in \mathbb{R}^d$, then $\lim_{i\to\infty} \square f(B_i) = f(p)$.

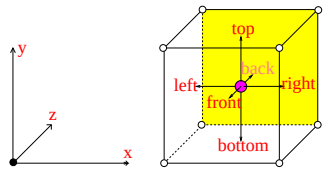


Figure 4: Box face conventions.

Note that using interval arithmetic, it is very easy to construct box functions when f is a polynomial. For a box $B = \prod_{i=1}^d I_i$, let $w(B) = \min_{i=1}^d w(I_i)$ denote the **width** of B, where $w(I)$ denotes the width of an interval. The 0-, 1- and 2-dimensional features of a box are called its **corners**, **edges**, and **faces**. For $i \in \{x, y, z\}$, an i**-face** is a face that is normal to the i-direction We also name each face of a box as 'front', 'back', 'top', 'bottom', etc, using the convention in Figure 4.

By an infinitesimal perturbation, we may assume that f has positive or negative signs at box corners (never the zero sign). Viewing signs ($+$ or $-$) as colors, we can talk about edges and boxes being **monochromatic** or **bichromatic**. As in Section 2, we introduce **vertices** in the middle of bichromatic edges. In our implemented code, we use linear interpolation to improve the quality of the meshes. On a face, we will introduce **arcs** connecting pairs of vertices (this need not be uniquely determined, as we saw). Finally, for each box B, we introduce a collection of **triangles** to form a triangulated patch G_B such that $G_B \cap \partial B$ is precisely these vertices and arcs. Thus, we use the corner/edge/face terminology for boxes, but reserve the vertex/arc/triangle terminology for the triangulated mesh.

¶7. Octrees. *We assume that each leaf of our octrees is labeled as "in" or "out".* A leaf box B is called an **in-box** if it is labeled "in"; similarly for an **out-box**. The set of all the in-boxes of T is called the **box-complex** defined by T. The union of all in-boxes is denoted $R(T)$, the **region represented by** T. Following [6], a set of the form $R(T)$ is called a **nice region**. Such regions are closed subsets of $\mathbb{R}^3$, but could be disconnected with holes and cavities. Two boxes of an octree are **neighbors** of each other if they have disjoint interiors but they share an open face (i.e., the relative interior of the face of one of the two boxes). We say they are **edge-neighbors** if they share an open line segment. Note that neighbors are automatically edge-neighbors, but the converse may not hold.

¶8. Box Predicates for Subdivision. The stopping criterion of the Subdivision Phase (¶1) is based on two box predicates: an **exclusion predicate** $C_{out}(B)$ and an **inclusion predicate** $C_{in}(B)$. Subdivision Phase ends when each in-box B satisfies $C_{out}(B)$ or $C_{in}(B)$. The in-boxes of T fall into three mutually exclusive types:
1. Discarded Boxes: these satisfy C_{out}
2. Candidate Boxes: these do not satisfy C_{out}, but an ancestor satisfies C_{in}.
3. Inconclusive boxes: do not satisfy C_{out} or C_{in}.

If B satisfies C_{in} but not C_{out}, then the above definition implies B is a candidate box (since B is an ancestor of itself). Discarded boxes will no longer be considered. Whenever we split a candidate box, we always check if each subboxes satisfy C_{out}: if so, it is discarded; otherwise it remains a candidate box. After the Subdivision Phase, no inconclusive boxes remain. For the Refinement Phase, we only split candidate boxes. The following list contains various instantiations for C_{out} and C_{in} used in this paper:

$$\left.\begin{array}{lcl} C_0(B) & : & 0 \notin \square f(B) \quad \text{(Exclusion)} \\ C_x(B) & : & 0 \notin \square f_x(B) \quad (x\text{-Monotonicity}) \\ C_{xy}(B) & : & C_x(B) \vee C_y(B) \quad \text{(Parametrizability)} \\ C_{xyz}(B) & : & C_x(B) \vee C_y(B) \vee C_z(B) \quad \text{(Parametrizability)} \\ C_1(B) & : & 0 \notin (\square f_x(B))^2 + (\square f_y(B))^2 + (\square f_z(B))^2 \quad \text{(Small Normal Variation)} \end{array}\right\} \tag{1}$$

Note that f_x, f_y, f_z refers to partial derivatives of f. Clearly, if $C_0(B)$ holds, then $S \cap B$ is empty. So we use C_0 as the exclusion predicate C_{out} in all our algorithms. For Snyder's and Cxyz Algorithms, $C_{in} = C_{xyz}$, and for PV Algorithm, $C_{in} = C_1$.

4 Regularized Cxyz Algorithm

An octree is "regular" if every leaf is at the same level, as in Marching Cubes. So the "Regularized Algorithm" amounts to enforcing this regularity during the Refinement Phase. In our Regularized Cxyz Algorithm, we can relax this requirement: we only require that two candidate boxes who are edge-neighbors must have the same width. The correctness of the Regularized Cxyz Algorithm is far from trivial. Its analysis will be critical for extension to subsequent algorithms. This tact of going through the regularized case follows [18, 14].

The algorithm only perform full-splits, and recall that its inclusion predicate C_{in} is C_{xyz}. This completely defines its Subdivision Phase. The Refinement Phase is defined by the rule that we split a candidate box B if it has an edge-neighbor that is a candidate box of smaller width. At the end of this process, any two edge-neighbors that are both candidates would have the same width. The rest of this section will focus on the Construction Phase, and correctness proof.

At this juncture, we insert a concept that will be useful in subsequent analysis. At the end of the Subdivision Phase, each candidate box B in the octree is known to satisfy $C_i(B)$ for some $i \in \{x, y, z\}$. We arbitrarily pick one of these i's and call it the **known monotone direction** ("monotone direction" for short) for B. In subsequent computation, when we split B, the candidate descendants of B will inherit this monotone direction. This direction is stored with B by our algorithm since some decisions will depend on it.

¶9. Sign Types, Arc Types and Surface Types under the C_{xyz} Predicate Of the 14 possible sign types of f at box corners shown in Figure 2, only 10 can arise under the Cxyz predicate. The 4 excluded cases are indicated by asterisks: Types $^*2c$, $^*3c$, $^*4e$, $^*4f$. As usual, we introduce vertices in the middle of bichromatic edges, and connect pairs of vertices on each face by arcs. The 10 sign types give rise to 13 **arc types** in Figure 3. Lemma 1 asserts that these arc types give rise to unique **surface type** within each box, shown in yellow in Figure 3.

¶10. Counter Example to the Neighborly Connection Rule. In 2-D, we can apply the above method to construct a surface in each box, without consideration of other boxes [14]. But now, there are two choices of arc connections when a face has 4 vertices: we call these **alternating faces**. In Figure 2, these faces are colored pink, as in Types (2b), (3b) and (4d). This implies that constructing surface patches in each box must (at least) be **neighborly**, meaning that two boxes sharing an alternating face must agree on which choice of arcs to make. Alternating faces arise even under the C_1 predicate of PV Algorithm. They showed *any* neighborly choice will lead to a correct surface, which is rather non-intuitive. For our C_{xyz} predicate, neighborly choices alone is insufficient: Figure 5 gives a counter example.

In Figure 5(a), the arc connections are neighborly. The two boxes satisfy C_x, but the triangulated surface determined by the indicated arc connections violate the C_x condition. Using a different arc connection, we obtain the triangulated surface in Figure 5(b) (this one is consistent with the C_x condition). Extending this example (using the phenomenon of "blocks" below) we see that a choice in one box can force the choice of boxes arbitrarily far away. E.g., Figure 5(c).

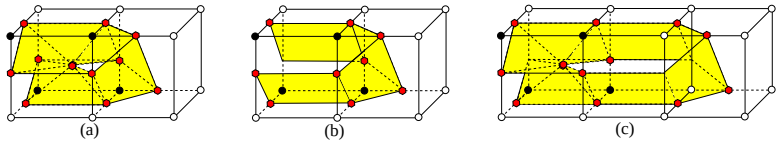


Figure 5: Neighborly choice of arc patterns is insufficient for correctness.

¶11. Alternating Faces (AF) Rule. For alternating faces, we provide the following globally consistent rule for connecting arcs: *RULE: the arcs will be line segments that are parallel to one of the three vectors:* $(1,1,0), (1,0,1), (0,1,1)$, *depending whether the alternating face is an* z*-,* y*- or* x*-face (respectively).* E.g., for an alternating x-face we will connect its four vertices with line segments that are parallel to the vector $(0,1,1)$, as in Type 2b(ii), and not as in Type 2b(i) of Figure 3. Call this the **Alternating Faces Rule** (AF Rule for short). With this rule, we have now completely specified the Regularized Cxyz Algorithm. See Appendix **A.4** for the correctness proof of Regularized Cxyz algorithm.

5 Balanced Cxyz Algorithm

We now extend the Regularized Cxyz Algorithm to the Balance Cxyz Algorithm. This extension aims at reducing the number of unnecessary splits. The idea is to allow the widths of edge neighbors to differ by a factor of ≤ 2; this is called "balancing". The tradeoff is that we are faced with more involved connection rules and correctness analysis. The Subdivision Phase is the same as in the regularized case. For the Refinement Phase, we need some notation. Let $i \in \{x, y, z\}$. An edge of a box is an i-**edge** if it is parallel to the i-axis. The i-**width** of a box is the length of its i-edges. An octree is i-**balanced** if for all pairs of candidate boxes B, B' which are edge-neighbors, then the i-widths of B and B' is within a factor of 2 of each other. The octree is **balanced** if it is i-balanced for all $i = x, y, z$. This general definition will be used later for the Rectangular Cxyz Algorithm. For now, we only do full splits and we can use $w(B)$ as the definition of width.

In the rest of the Balanced Cxyz Algorithm, all our queues will be minimum priority queues. The comparison criterion for these queues is $w(B)$ for each box B. The Refinement Phase has three sub-phases:

Refinement Phase:
1. $T_1' \leftarrow Balance(T_1)$
2. For each candidate box in T_1', introduce vertices in the middle of bichromatic edges.
3. $T_2 \leftarrow Disambiguate(T_1')$

The first sub-phase $Balance(T_1)$ amounts to splitting any candidate box B that has an edge-neighbor of width $> 2w(B)$. At the end of this sub-phase, we say the octree is "balanced". The third sub-phase is based on the concept of ambiguity which we next introduce.

¶12. Disambiguation Sub-phase We want to call certain boxes "ambiguous" if there is not enough information to do a MC-like construction, and this is resolved by splitting the ambiguous box. This may in turn cause new boxes to become ambiguous. In the following we will identify three kinds of ambiguity.

Let us indicate the issues that arise if we simply replace C_1 by C_{xyz} in the Balanced (Cxyz) Algorithm. Consider an horizontally-stretched hyperboloid as in Figure 6 (a_1). We run the Balanced Algorithm on this hyperboloid, and the Subdivision Phase terminates with the 10 boxes shown in Figure 6 (a_2). Clearly, both of the two larger boxes (B_1 and B_3) satisfy C_x. The output graph obtained by our connection rules (in the Regularized Algorithm) is the yellow polytope G seen in Figure 6(a_2). Since G forms a closed surface, it is clearly wrong. An error occurred in box B_1 (and also B_3) where $S \cap B_1$ is a tube while $G \cap B_1$ is a planar surface. If we had split B_1, we would have discovered this error. We say B_1 (resp., B_3) has "3D ambiguity". A similar problem is seen in Figure 6(b_1), corresponding to "2D ambiguity" in each of the boxes B_1, B_3, B_4, B_6. Suppose B satisfies C_y. Then we say B has **3D ambiguity** if the interior of its top or bottom faces has four vertices. We say B has **2D ambiguity** if one or more of its vertical faces has exactly two vertices on the same edge. Note that this edge is not a vertical edge because $C_y(B)$ is satisfied.

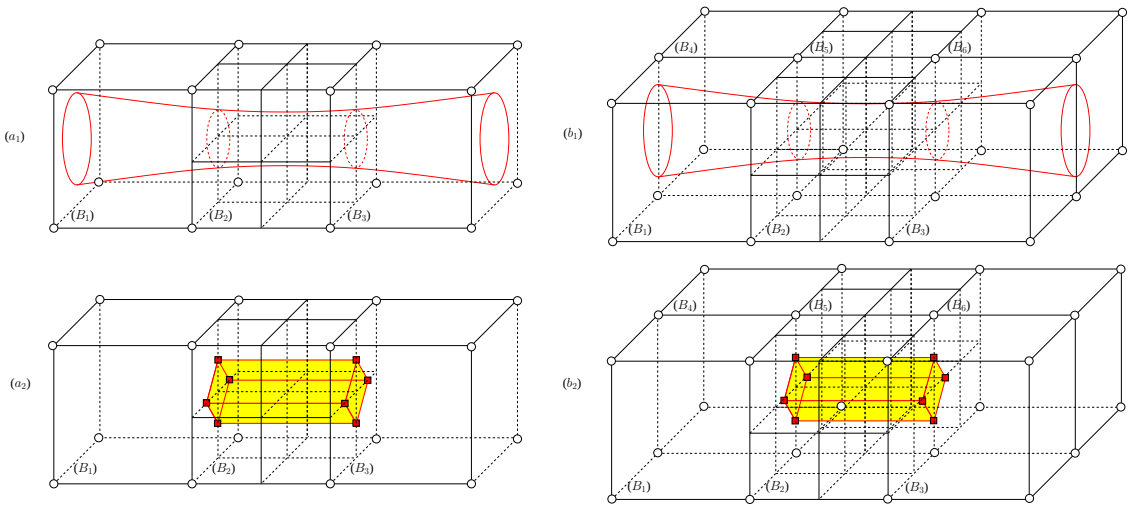


Figure 6: Examples of 2D and 3D ambiguity.

This definition is modified accordingly if B satisfies C_x or C_z. In Figure 6(a1), the ambiguous boxes satisfies C_x. In Figure 6(b1), the ambiguous boxes might satisfy C_x or C_z.

We now describe the third kind of ambiguity. Its motivation will be become clearer in the Construction Phase below. Let $i \in \{x, y, z\}$ be the monotone direction of a box B. We say B has an **alternating ambiguity** if it properly contains the i-face F of its neighbor, and this F is alternating.

Finally, a box B is said to be **ambiguous** if it is 2D, 3D or alternating ambiguous.

LEMMA 2. *If we split an ambiguous box B into 8 subboxes, none of these subboxes will be ambiguous.*

Nevertheless, splitting of ambiguous boxes might induce its edge-neighbors to become ambiguous. and also cause the octree to be unbalanced. The re-balance procedure is very local, we only need to propagate the "modified" boxes. We will next describe the Construction Phase for the Balanced Cxyz Algorithm.

¶13. Construction Phase Let F be a face of some box B. Our first goal is to connect the vertices on F by arcs. Let B' be a neighbor of B that shares part of F as a common face. There are two possibilities: If $B' \cap B = F$, then B' has width at least that of B. This is the case we are interested in: call F **active** in this case. Otherwise, F is **inactive**; this means B' must have width that is half that of B. We are not interested in inactive F because we would have processed the faces of B' before B, and in particular, any vertex in F would have been processed. Henceforth, we will only focus on arc connections for active faces.

Recall that at the end of the Refinement Phase, we have an octree T_2 in which all the bichromatic edges have a vertex in its middle. Our goal is to connected pairs of these vertices into arcs. Define an **arc loop** to be a closed curve comprising of such arcs on the boundary of a box B. The Construction Phase has three steps:

Construction Phase:
Let Q be a priority queue of the candidate boxes in T_2.
While (Q is non-empty)
Remove a box B from Q
1. Arc connect the vertices on the active faces of B
2. Group the arcs on B's boundary into arc loops
3. Triangulate the arc loops on the boundary of B

Steps 2 and 3 are straightforward. In the following, we will describe how to implement Step 1.

¶14. Sign Types of Active Faces Note that each edge of an active face can have at most two vertices. There might be a neighbor B' of B that shares an edge with an active F. If B' has smaller width than B, then a corner of B' would be the midpoint of an edge of F. Therefore, in considering sign types of F, we need to consider signs of such midpoints. There can be up to 8 signs on the boundary of F. The possible **Sign Types**

of such faces are enumerated in Figure 7 – there are 13 in number. The sign type of F will uniquely determine the vertices that are introduced into F (as illustrated in Figure 7).

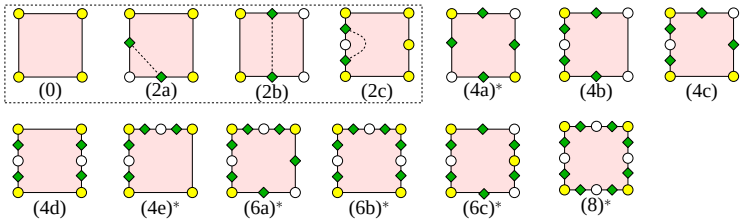


Figure 7: Sign Types of active faces.

¶15. Arc Types of Active Faces Let F be an active face, and suppose F bounds two boxes B and B', i.e., $F = B \cap B'$. The rule for arc connection in F depends on whether F is (known to be) "parametrizable" or not. Let us define this concept. We say F is **known parametrizable** if F is parallel to the monotone direction of B or B'. Otherwise, F is said to be **not known parametrizable**.

Assume B is a C_y box. Then the four faces of B which are parallel to the y-direction are clearly known parametrizable faces. It follows from our analysis for curves [14] that each of these faces can have at most 4 vertices. So B can have at most 16 vertices on its edges. Indeed, it is easy to see that 16 vertices can arise. Our connection rule for the known parametrizable faces can follow the rules given in [14]. For reference, call this the **parametrizable face rule** which is reproduced in Figure 8.

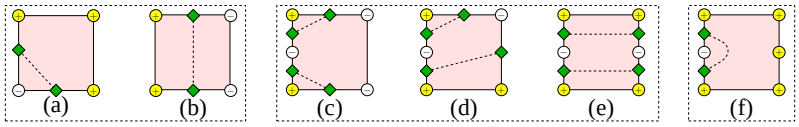


Figure 8: Parametrizable Face Rules.

It remains to give the connection rule for the case where F is not known parametrizable. In the Regularized Algorithm, the arc connections on F may be arbitrary, as long as we ensure block-wise consistency. But the Balanced Cxyz Algorithm needs a new approach.

We define the term i-block ($i \in \{x, y, z\}$) for the balanced octree T_2. For definiteness, let $i = y$. A **y-block** $\mathcal{B}$ is a sequence $B_1, \ldots, B_t$ of candidate boxes of T_2 such that (1) the bottom face of B_j is the top face of B_{j+1} for $j = 1, \ldots, t-1$; (2) the monotone direction for each B_i is y; and (3) the block is maximal. Note that this implies that all the boxes in a block have the same width. The **width** of the block is defined as the width of any B_i. Also the **end faces** of $\mathcal{B}$ refers to the top face of B_1 and bottom face of B_t.

Recall that every candidate box in our octree T_2 has been assigned or inherited a monotone direction from the Subdivision Phase. This partitions the set of candidate boxes of T_2 into blocks as defined above. All the boundary faces of a block can be connected using the above Parametrizable Face Rule, except for the end faces which is addressed in the next lemma.

LEMMA 3. *Let F be an active end face of a block.*
(a) If F is not known parametrizable, then it has at most 2 vertices.
(b) If F is known parametrizable, then F has at most 4 vertices. When there are 4 vertices, the sign types are one of Figure 7(4b), (4c) and (4d). These can be connected using the Parametrizable Face Rule.

The correctness of above lemma depends on the fact that we have resolved alternating ambiguities in the Refinement Phase. The only faces whose connection rule remains undecided after the above discussion are those in the interior of blocks. We know from previous counter examples that there is a need for global consistency, but it cannot be solved using a simple fixed rule like the AF Rule. Our solution is as follows:
(1) if all but one face remains unconnected, we can connect this face in a safe way (i.e., one which will not lead to contradiction). This connection rule will be known as the "Matching Rule".
(2) in any candidate box, at most two opposite faces cannot be connected by the Parametrizable Face Rule.

To "process" a box B in the present context means to connect all the vertices on the faces of B. We can now process B as follows: if (1) holds, we can process B by using the Matching Rule to connect its remaining unconnected face. Otherwise (2) holds, and we search in any one of the two directions of the block containing B, looking at neighboring boxes $B_1, B_2, \ldots$ until we find a box B_k that satisfies (1). Then we apply the

Matching Rule to B_i for $i = k, k-1, \ldots, 1$. Thus each B_i is processed, and B can now be processed using the Matching Rule.

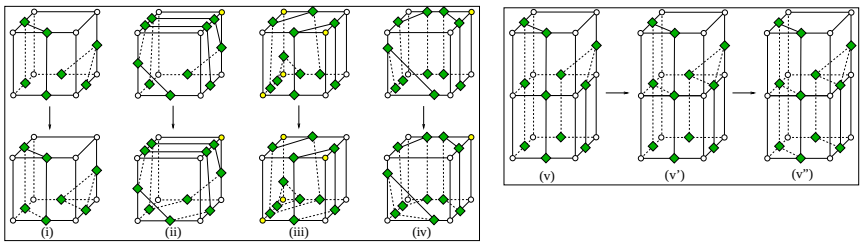


Figure 9: Examples of matching rules ((i), (ii), (iii) and (iv)) and propagation rules ((v)→(v')→(v")) to connect vertices.

Let us now define the **Matching Rule** for a candidate box B with parametrization direction y. Assume that B's top face, as well as the other four faces parallel to y-direction, have been connected. Then the Matching Rule tells us how to connect the bottom face F. Let $v_1, v_2, \ldots, v_{2m}$ be the vertices on the boundary of F. Note that $m \leq 4$. The Matching Rule tells us to introduce the arc (v_i, v_j) if there exists a path of arcs on the boundary of B from v_i to v_j. Note that this rule yields a unique way to connect all the vertices on F. Figure 9 illustrates this Matching Rule. The correctness proof of the Balanced Cxyz Algorithm follows the same structure as that of the Regularized Cxyz Algorithm: See Appendix **A.5** for the correctness proof of Balanced Cxyz algorithm.

6 Rectangular Cxyz Algorithm – Exploiting Anisotropy

The ability to have partial splits (i.e., half-splits or quarter-splits) can be highly advantageous. We design an algorithm called **Rectangular Cxyz Algorithm** to exploit this. A technique from the 2-D version [14] can be applied here, though the details are considerably more complicated. To ensure termination, we must fix some arbitrary upper bound $\rho > 1$ on the aspect ratio of any inconclusive box. The **aspect ratio** of a box is the ratio of the lengths of the longest edge to shortest edge. Please refer to [13] and Appendix **A.2** for details.

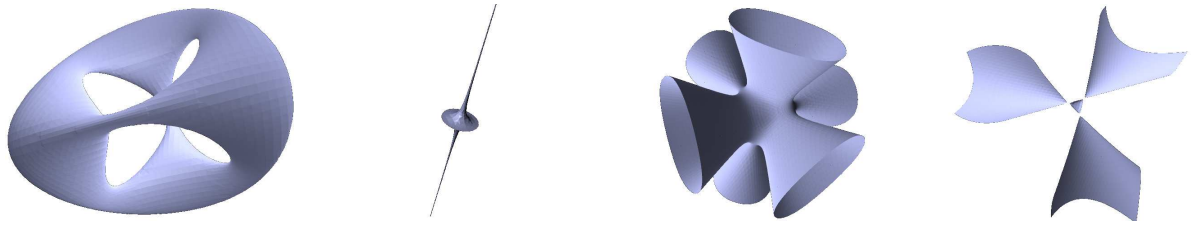
7 Experimental Results

Our algorithms are implemented in `Java` on the Eclipse Platform. All examples are run on an Intel Core2 Duo Mobile Processor T2500 (2.0Ghz, 667FSB, 2MB shared L2 Cache) and 2.0Gb of RAM. We use the default `Java` heap memory 256MB (some runs result in OutOfMemoryError (OME)). We plan to convert the `Java` codes to `C++` for distribution with our open source `Core Library`. We implemented four algorithms: PV, Balanced Cxyz, Balanced Cxyz with epsilon precision, and Rectangular Cxyz. These are abbreviated as PV, Cxyz, Cxyze, and Rect-n (where n is the maximum aspect ratio). Table 1 lists 11 examples of our tests. Figure 10 visualizes the surfaces of Eg2, Eg3, Eg6 and Eg7. Table 2 compares the number of boxes and timings (in ms) among Cxyz, PV, and Rect-n $(n = 2, 4, 8, 16, 32)$. The percentages represent the relative number of boxes and the relative timing, with Cxyz as 100%. See Appendix **A.1** for additional images.

#	Curve name	Equation $f(x, y, z) = 0$	Original Box
Eg1	tangle cube	$x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 10$	$[(-8,-8,-8),(8,8,8)]$
Eg2	chair	$(x^2 + y^2 + z^2 - 23.75)^2 - 0.8((z-5)^2 - 2x^2)((z+5)^2 - 2y^2)$	$[(-8,-8,-8),(8,8,8)]$
Eg3	quartic cylinder	$y^2x^2 + y^2z^2 + 0.01x^2 + 0.01z^2 - 0.01$	$[(-8,-8,-8),(8,8,8)]$
Eg4	quartic cylinder	$y^2(x-1)^2 + y^2(z-1)^2 + 0.01(x-1)^2 + 0.01(z-1)^2 - 0.2002$	$[(-5,-5,-5),(7,7,7)]$
Eg5	quartic cylinder	$y^2(x-1)^2 + y^2(z-1)^2 + 0.01(x-1)^2 + 0.01(z-1)^2 - 1.0002$	$[(-12,-12,-12),(14,14,14)]$
Eg6	shrek	$-x^4 - y^4 - z^4 + 4(x^2 + y^2z^2 + y^2 + z^2x^2 + z^2 + x^2y^2) - 20.7846xyz - 10$	$[(-8,-8,-8),(8,8,8)]$
Eg7	tritrumpet	$8z^2 + 6xy^2 - 2x^3 + 3x^2 + 3y^2 - 0.9$	$[(-8,-8,-8),(8,8,8)]$
Eg8a	eclipse	$x^2 + 10^2y^2 + 10^2z^2 - 1$	$[(-8,-8,-8),(8,8,8)]$
Eg8b(n) ($n = 2, 4, 6$)	eclipse	$x^2 + 10^ny^2 + 10^nz^2 - 1$	$[(-7,-7,-7),(8,8,8)]$

Table 1: Equations and input boxes of examples

(1) Cxyz is at least as good as PV, and is significantly faster than PV in most examples. In Eg8b(4), Cxyz is 7.5 times faster than PV. In Eg8b(6), Cxyz spends 1.3 seconds to construct the mesh, compared to PV which spends more than 70 seconds and runs out of memory. Rect is the fastest in both Eg8b(4) and Eg8b(6): Rect-2 spends 141 ms for Eg8b(4), and 172 ms for Eg8b(6). The only exception is Eg8a where Cxyz and PV produce the same number of boxes, and spend the same amount of time. In Eg8b(2), we use the same function as Eg8a, but with an asymmetric original box. Cxyz is twice as fast as PV. Also note that in the Eg3, Cxyz and PV



(a) Eg2:chair (b) Eg3:quartic cylinder (c) Eg6:shrek (d) Eg7:tritrumpet

Figure 10: Approximation of various examples in Table 1.

also produce the same number of boxes, but Cxyz is faster than PV because the computational cost for the C_1 predicate is bigger than the C_{xyz} predicate.

(2) Rect can be significantly faster than Cxyz, but the performance of Rect is inconsistent. In Eg3, Rect-32 takes 11.8% of Cxyz's time; and in Eg8b(6), Rect-2 takes 12.8% of Cxyz's time. The input surface for these examples are very long and thin, allowing Rect to take advantage of larger aspect ratios. The results show that although Rect produces fewer boxes than Cxyz in all examples but Eg8b(2), the running time of Rect is not always faster than the Cxyz (as in Eg2 with a "squarish" input surface). This is because Rect must spend more time checking splitting criteria, and processing boxes in 3 directions.

(3) Increasing the maximum aspect ratio n in Rect does not necessarily improve the performance of the algorithm. In Eg3, increasing the maximum aspect ratio directly improves the performance of Rect; but in Eg8b(6), it has an opposite effect. This is because increasing the maximum aspect ratio might cause the boxes to "over split" in one direction, which is also the reason for the inconsistency of Rect. Another example for over-splitting in Rect is Eg2, where Rect-n spends more time than Cxyz. Figure 17 in Appendix shows the resulting boxes, meshes, and details by running Cxyz, Rect-8, and Rect-32 on Eg2.

Equation	Cxyz	PV	Rect-2	Rect-4	Rect-8	Rect-16	Rect-32
Eg1	2584 / 391	198% / 184%	42% / 148%	50% / 168%	66% / 200%	81% / 236%	103% / 288%
Eg2	26104 / 4516	406% / 349%	51% / 163%	76% / 236%	98% / 302%	118% / 372%	141% / 451%
Eg3	35792 / 3437	100% / 112%	33% / 82%	18% / 47%	9% / 28%	6% / 17%	3% / 12%
Eg4	80662 / 10282	$OME_{>90sec.}$	54% / 174%	41% / 129%	34% / 105%	36% / 115%	33% / 103%
Eg5	134163 / 17187	$OME_{>90sec.}$	48% / 205%	28% / 86%	23% / 71%	21% / 65%	20% / 61%
Eg6	31144 / 4046	319% / 296%	44% / 134%	52% / 171%	62% / 208%	70% / 255%	77% / 283%
Eg7	1688 / 328	172% / 128%	47% / 109%	50% / 119%	61% / 129%	74% / 138%	98% / 176%
Eg8a	400 / 94	100% / 100%	44% / 133%	50% / 149%	58% / 166%	68% / 166%	80% / 183%
Eg8b(2)	274 / 125	789% / 200%	54% / 87%	56% / 87%	72% / 100%	82% / 112%	102% / 112%
Eg8b(4)	1247 / 203	1774% / 754%	28% / 69%	34% / 69%	39% / 77%	44% / 85%	53% / 100%
Eg8b(6)	15226 / 1343	$OME_{>70sec.}$	5% / 13%	5% / 14%	6% / 15%	6% / 15%	7% / 16%

Table 2: Cxyz vs. PV vs. Rect-n

(4) We also ran our algorithm on the high order polynomial $f(x, y, z) = x^{300} + y^{300} + z^{300} - 1 = 0$. To construct a correct mesh, Cxyz uses 188 ms; PV uses 219 ms; Rect-2 uses 296 ms and Rect-4 uses 375 ms. This shows that subdivision algorithms can perform well when the input function is a high degree polynomial. On the other hand, starting from Rect-8, there are overflow/underflow errors. This problem can be resolved if we use a library like our `Core Library`.

8 Conclusion

This paper introduces new algorithms for the isotopic approximation of implicit surfaces. Our algorithms are relative simple, efficient and easy to implement. A main idea is to exploit parametrizability (as in Snyder) and nonlocal isotopy (as in Plantinga & Vegter), and we further extend this idea to anisotropic subdivision. Our comparison with three algorithms (PV, Balanced Cxyz, and Rectangular Cxyz) show that our Cxyz Algorithm is consistently more efficient than PV and the Rectangular Cxyz Algorithm can exhibit significant speedup. But the precise way to exploit anisotropy remains a research problem. The major open problem is to extend this work to higher dimensions. It is a challenge to find faster methods for surface refinement. Finally two general open problems are the effective treatment of singularity using numerical methods, and the complexity analysis of subdivision algorithms.

References

[1] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics. Springer, 2003.

[2] J.-D. Boissonnat, D. Cohen-Steiner, B. Mourrain, G. Rote, and G. Vegter. Meshing of surfaces. In Boissonnat and Teillaud [5]. Chapter 5.

[3] J.-D. Boissonnat, D. Cohen-Steiner, and G. Vegter. Isotopic implicit surfaces meshing. In *ACM Symp. Theory of Comput.*, pages 301–309, 2004.

[4] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.

[5] J.-D. Boissonnat and M. Teillaud, editors. *Effective Computational Geometry for Curves and Surfaces*. Springer, 2006.

[6] M. Burr, S. Choi, B. Galehouse, and C. Yap. Complete subdivision algorithms, II: Isotopic meshing of singular algebraic curves. In *33th Int'l Symp. Symbolic and Alge. Comp. (ISSAC'08)*, pages 87–94, 2008. Hagenberg, Austria. Jul 20-23, 2008. Accepted for Special Issue of ISSAC 2008 in JSC. Also, in arXiv:1102.5463.

[7] S.-W. Cheng, T. Dey, E. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. In *Proc. 20th ACM Symp. on Comp. Geom.*, pages 280–289, 2004.

[8] A. Eigenwillig, L. Kettner, E. Schmer, and N. Wolpert. Complete, exact, and efficient computations with cubic curves. In *20th ACM Symp. on Comp. Geom.*, pages 409 – 418, 2004. Brooklyn, New York, USA, Jun 08 – 11.

[9] Exact Geometric Computation homepage, Since 1996. FAQs, downloads, documentation and links from URL `http://cs.nyu.edu/exact/`.

[10] H. Hong. An efficient method for analyzing the topology of plane real algebraic curves. *Mathematics and Computers in Simulation*, 42:571–582, 1996.

[11] N. Kamath. Subdivision algorithms for complex root isolation: Empirical comparisons. Master's thesis, Oxford University, Oxford Computing Laboratory, Aug. 2010.

[12] N. Kamath, I. Voiculescu, and C. Yap. Empirical study of an evaluation-based subdivision algorithm for complex root isolation. In *4th Intl. Workshop on Symbolic-Numeric Computation (SNC)*, pages 155–164, 2011.

[13] L. Lin. *Adaptive Isotopic Approximation of Nonsingular Curves and Surfaces*. Ph.D. thesis, New York University, Sept. 2011.

[14] L. Lin and C. Yap. Adaptive isotopic approximation of nonsingular curves: the parameterizability and nonlocal isotopy approach. *Discrete and Comp. Geom.*, 45(4):760–795, 2011. Special Conference Issue based on 25th ACM Symp. on Comp.Geom, 2009.

[15] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, July 1987.

[16] R. Martin, H. Shou, I. Voiculescu, A. Bowyer, and G. Wang. Comparison of interval methods for plotting algebraic curves. *Computer Aided Geometric Design*, 19(7):553–587, 2002.

[17] R. E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, NJ, 1966.

[18] S. Plantinga and G. Vegter. Isotopic approximation of implicit curves and surfaces. In *Proc. Eurographics Symposium on Geometry Processing*, pages 245–254, New York, 2004. ACM Press.

[19] H. Ratschek and J. Rokne. *Computer Methods for the Range of Functions*. Horwood Publishing Limited, Chichester, West Sussex, UK, 1984.

[20] H. Ratschek and J. G. Rokne. SCCI-hybrid methods for 2d curve tracing. *Int'l J. Image Graphics*, 5(3):447–480, 2005.

[21] M. Sagraloff and C. K. Yap. A simple but exact and efficient algorithm for complex root isolation. In *36th Int'l Symp. Symbolic and Alge. Comp. (ISSAC'11)*, pages 353–360, 2011. June 8-11, San Jose, California.

[22] E. Schoemer and N. Wolpert. An exact and efficient approach for computing a cell in an arrangement of quadrics. *Comput. Geometry: Theory and Appl.*, 33:65–97, 2006.

[23] R. Seidel and N. Wolpert. On the exact computation of the topology of real algebraic curves. In *Proc. 21st ACM Symp. on Comp. Geom.*, pages 107–116, 2005. Pisa, Italy.

[24] J. M. Snyder. Interval analysis for computer graphics. *SIGGRAPH Comput.Graphics*, 26(2):121–130, 1992.

[25] B. T. Stander and J. C. Hart. Guaranteeing the topology of an implicit surface polygonalization for interactive meshing. In *Proc. 24th Computer Graphics and Interactive Techniques*, pages 279–286, 1997.

[26] G. Taubin. Distance approximations for rasterizing implicit curves. *ACM Transactions on Graphics*, 13(1):3–42, 1994.

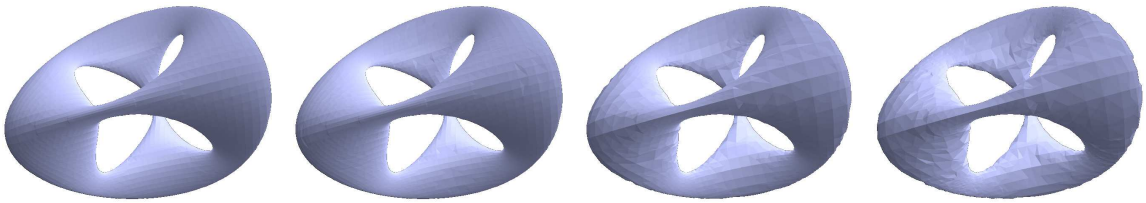
[27] G. Taubin. Rasterizing algebraic curves and surfaces. *IEEE Computer Graphics and Applications*, 14(2):14–23, 1994.

A Appendix

In this appendix, we provide details of the correctness proofs. Correctness is nontrivial because our exploitation of non-local isotopy forces us to do global arguments. Most of the proofs are included here, but the omitted ones may be found in Lin's Thesis [13, 9]. First, we show more figures from our experiments.

A.1 More Examples

This section illustrates the surfaces for Eg.2 to Eg.7 in Table 1 using Cxyze, PV, Cxyz and Rect-n. n is selected in a way that Rect-n is the fastest among all Rect algorithms.

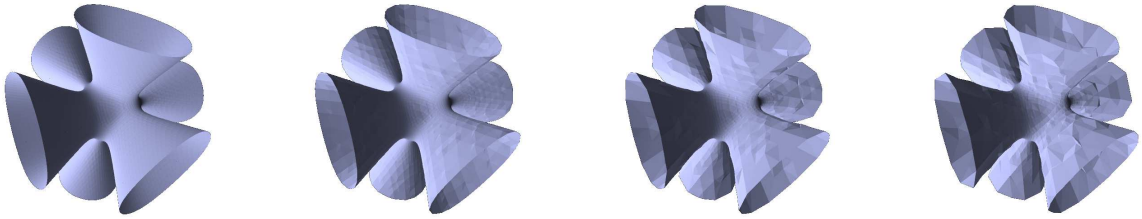


(a) Cxyze (b) PV (c) Cxyz (d) Rect-2

Figure 11: Approximation of Eg2: chair $f(x,y,z) = (x^2+y^2+z^2-23.75)^2 - 0.8((z-5)^2-2x^2)((z+5)^2-2y^2) = 0$.

(a) Cxyze (b) PV (c) Cxyz (d) Rect-32

Figure 12: Approximation of Eg3: quartic cylinder $f(x,y,z) = y^2x^2 + y^2z^2 + 0.01x^2 + 0.01z^2 - 0.01 = 0$.

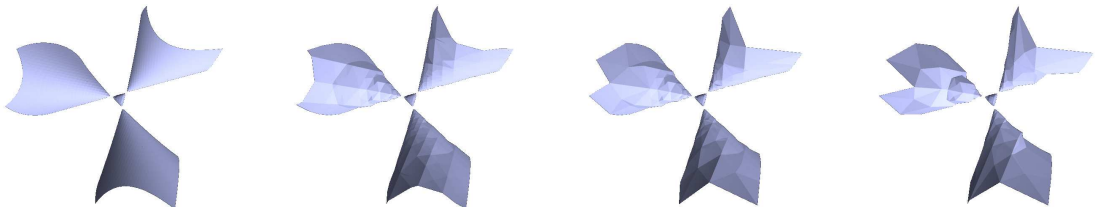


(a) Cxyze (b) PV (c) Cxyz (d) Rect-2

Figure 13: Approximation of Eg6: shrek $f(x,y,z) = -x^4 - y^4 - z^4 + 4(x^2 + y^2z^2 + y^2 + z^2x^2 + z^2 + x^2y^2) - 20.7846xyz - 10 = 0$.

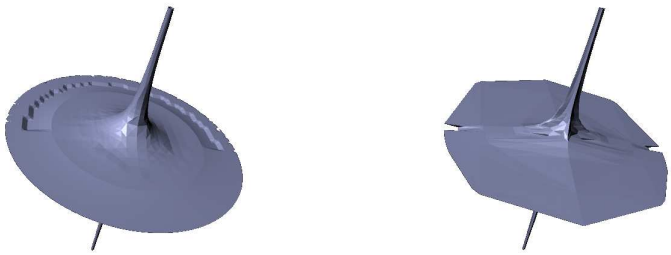
A.2 Rectangular Cxyz Algorithm

The ability to have partial splits (i.e., half-splits or quarter-splits) can be highly advantageous. We design such an algorithm, known as the Rectangular Cxyz Algorithm. A technique from the Rectangular Cxy Algorithm [14] can be applied (the implementation details are considerably more complicated). To ensure termination, we must fix some arbitrary upper bound $\rho > 1$ on the aspect ratio of any inconclusive box. The **aspect ratio** of a box is the ratio of the lengths of the longest edge to shortest edge. For the Subdivision Phase, we test



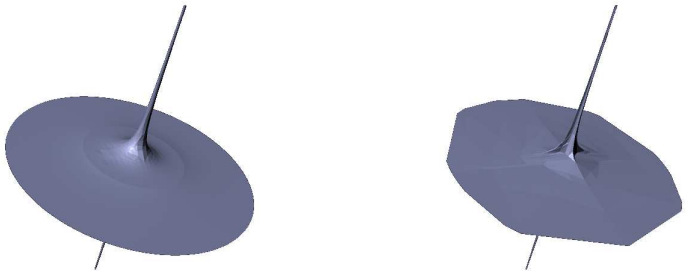
(a) Cxyze (b) PV (c) Cxyz (d) Rect-2

Figure 14: Approximation of Eg7: tritrumpet $f(x,y,z) = 8z^2 + 6xy^2 - 2x^3 + 3x^2 + 3y^2 - 0.9 = 0$.



(a) Cxyz (b) Rect-32

Figure 15: Approximation of Eg4: quartic cylinder1 $f(x,y,z) = y^2(x-1)^2 + y^2(z-1)^2 + 0.01(x-1)^2 + 0.01(z-1)^2 - 0.2002 = 0$.

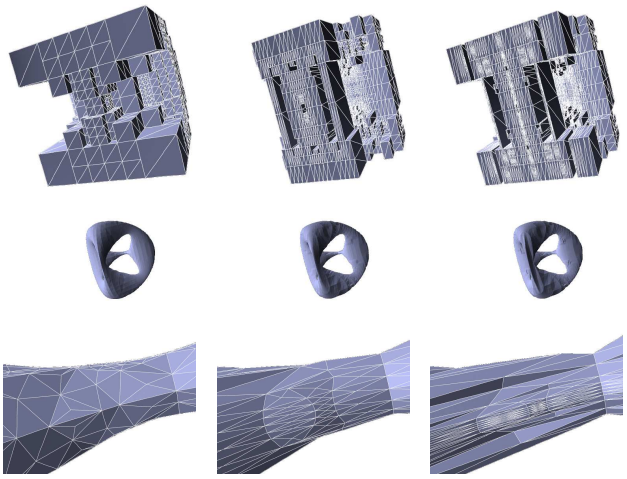


(a) Cxyz (b) Rect-32

Figure 16: Approximation of Eg5: quartic cylinder2 $f(x,y,z) = y^2(x-1)^2 + y^2(z-1)^2 + 0.01(x-1)^2 + 0.01(z-1)^2 - 0.1002 = 0$.

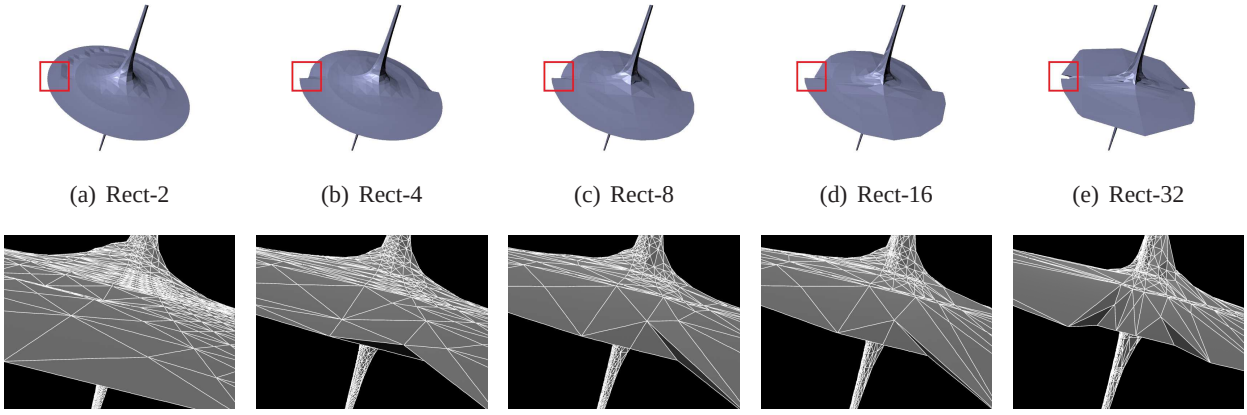
each box B as follows. We go through the following list of predicates which amounts to checking C_0 or C_{xyz} on the whole, half-, quarter- parts of B. This list of of predicates is given as (2) in the Appendix.

$$
\left.
\begin{array}{l}
L_0: \\
\quad C_{out}: C_0(B) \\
\quad C_{in}: C_{xyz}(B) \\
L_1: \\
\quad C_{out}: C_0(B_{1234}), C_0(B_{5678}), C_0(B_{1278}), C_0(B_{3456}), C_0(B_{1458}), C_0(B_{2367}) \\
\quad C_{in}: C_{xyz}(B_{1234}), C_{xyz}(B_{5678}), C_{xyz}(B_{1278}), \\
\qquad C_{xyz}(B_{3456}), C_{xyz}(B_{1458}), C_{xyz}(B_{2367}) \\
L_2: \\
\quad C_{out}: C_0(B_{12}), C_0(B_{34}), C_0(B_{56}), C_0(B_{78}), C_0(B_{14}), C_0(B_{23}), \\
\qquad C_0(B_{67}), C_0(B_{58}), C_0(B_{18}), C_0(B_{27}), C_0(B_{36}), C_0(B_{45}) \\
\quad C_{in}: C_{xyz}(B_{12}), C_{xyz}(B_{34}), C_{xyz}(B_{56}), C_{xyz}(B_{78}), C_{xyz}(B_{14}), C_{xyz}(B_{23}), \\
\qquad C_{xyz}(B_{67}), C_{xyz}(B_{58}), C_{xyz}(B_{18}), C_{xyz}(B_{27}), C_{xyz}(B_{36}), C_{xyz}(B_{45})
\end{array}
\right\} \quad (2)
$$



(a) Cxyz (b) Rect-8 (c) Rect-32

Figure 17: Boxes, meshes, and details of Eg2 using Cxyz, Rect-8 and Rect-32. Note that the triangles are elongated as the maximum aspect ratio increases.



(a) Rect-2 (b) Rect-4 (c) Rect-8 (d) Rect-16 (e) Rect-32

(a) Rect-2 (b) Rect-4 (c) Rect-8 (d) Rect-16 (e) Rect-32

Figure 18: (a)-(e): Approximations of quartic cylinder1 $f(x,y,z) = y^2(x-1)^2 + y^2(z-1)^2 + 0.01(x-1)^2 + 0.01(z-1)^2 - 0.2002 = 0$ using Rect-n ($n = 2, 4, 8, 16, 32$). (f)-(j): Local topology preservation in the squared area of the approximations.

In this list, the subboxes of B are labeled using some fixed convention[1] for labeling the 8 orthants of the coordinate system. This list has three sublists (L_0, L_1, L_2). If a condition in L_0 is verified we tag B as an in- or out-box, accordingly. If a condition in L_1 (L_2) is verified, we half- (quarter-) split to produce a child that satisfies that condition, and tag that child accordingly. If no condition is verified, we do a full-split. Finally, for balancing, we balance in the x-, y- and z-directions independently. This could create pairs (B, B') of neighboring boxes where $B \cap B' = F$ but F is a proper subface of B and of B'. We half-split either B or B' to make F a face of a subbox. Now, F would be active, and this allows our former analysis to work. The Disambiguation Sub-phase and Construction Phase are unchanged.

[1] Unlike the 2-D case, there seems to be no universally accepted convention for this. See, e.g., http://godplaysdice.blogspot.com/2007/09/convention-for-quadrantoctantorthant.html. We will use the gray code to label successive orthants, starting from $1 = 000, 2 = 001, 3 = 011, 4 = 010, 5 = 110, 6 = 111, 7 = 101, 8 = 100$.

A.3 Overview of Correctness Proof

In this section, we will give an overview of the correctness proof, both for the regularized algorithm and the balanced algorithm. They have a common structure, but we will point out differences.

Correctness means that the output graph G is isotopic to S in the input region $R(T_0)$, denoted $G \simeq S$ (**mod** $R(T_0)$). Let T be the final octree produced by the algorithm.

The proof consists of two major steps. First we show the existence of a surface $\widetilde{S}$ that is isotopic to S via an isotopy that respects the vertices of T. This means that the intermediate surfaces of the isotopy does not intersect the vertices of T. We denote this relation by "$S \simeq \widetilde{S}$ (**mod** T)". Moreover, this surface $\widetilde{S}$ has some nice properties relative to T, namely, $\widetilde{S}$ should intersect all the segments and faces of T in a "clean" way. Here, "segment" means any edge of a box that does not have a corner in its interior. To intersect a face "cleanly" means $\widetilde{S}$ does not intersect the face in any loop. To intersect a segment "cleanly" means $\widetilde{S}$ intersects it at most once. To show the existence of such an $\widetilde{S}$, we will give a conceptual process to remove loops and remove pairs of intersections on segments. But we need to define a partial order on loops and pairs and to show that we can remove minimal elements of this partial order repeatedly. When this partial order is empty, the surface is clean.

It turns out that to define this partial order, we need to maintain some monotonicity property of the surface (not the underlying function that defines the surface). Here we see a major difference between the regularized and the balanced case: in the former, we could remove all the loops before the pairs, and so we can define a separate partial order on loops, and on pairs. In the latter, we need to define a single partial order on their union.

A maximal set of boxes that are connected by alternating faces is called an **alternating block**. The second major step is to show that $G \simeq \widetilde{S}$ within each alternating block of T. Finally, we can conclude that $G \simeq \widetilde{S} \simeq S$ (**mod** $R(T)$).

A.4 Correctness of Regularized Cxyz Algorithm

We address the correctness of the Regularized Cxyz Algorithm. The proof is subtle, and harder than the $2D$ Regularized Cxy Algorithm or the $3D$ Regularized PV Algorithm. Our previous $2D$ proof for Cxy does not seem easy to generalize to $3D$, so we use a different approach. This proof will form the basis for proving the correctness of the Balanced Cxyz Algorithm in the next section.

Let T be an octree. We say S **intersects the boundary of** $R(T)$ **generically** if:

- For each boundary face F, the surface S intersects F transversally, and does not pass through any corner of F.
- The set $S \cap F$ is a finite collection of a finite set of closed loops and/or open curves. By an open curve, we mean one that has two distinct endpoints. The loops lie in the interior of F, and the open curves terminate transversally on the edges of F.

First, we will prove the termination of the subdivision phase. Let T_0 denote the octree representation of the original nice region R_0.:

LEMMA 4. *If $S = f^{-1}(0)$ intersects the boundary of $R(T_0)$ generically, and if f has no singularities in $R(T_0)$, then the subdivision phase will terminate.*

Proof. If the subdivision phase does not terminate, then there is an infinite decreasing sequence of boxes $B_0 \supset B_1 \supset \cdots$ such that each $C_0(B_i)$ and $C_{xyz}(B_i)$ fail. Thus:

$$0 \in (\square f(B_i) \cap \square f_x(B_i) \cap \square f_y(B_i) \cap \square f_z(B_i)). \tag{3}$$

The boxes B_i must converge[2] to some point $p \in R(T_0)$ as $i \to \infty$. Since $\square f$ is a box function for f, we conclude that $\square f(B_i) \to f(p)$. Then (3) implies $0 = f(p) = f_x(p) = f_y(p) = f_z(p)$. Thus, f has a singular point in $R(T_0)$. **Q.E.D.**

[2] The existence of p depends only on the existence of a bound r on the maximum aspect ratio – so this proof applies in the more general setting of Rectangular Cxyz Algorithm later.

From now on, let T be the octree at the termination of the Regularized Cxyz Algorithm, and G be the graph constructed by our rules from T.

¶16. Monotone Surfaces Let $S \subseteq \mathbb{R}^3$ be a continuous surface, $B \subseteq \mathbb{R}^3$ be a rectangular box and $i \in \{x, y, z\}$. An i-**line** is a straight line that is parallel to the i-axis.

We say S is i-**graph-like** in B if $|S \cap B \cap L| \leq 1$ for every i-line L. We say S is i-**monotone** in B if it is i-graph-like and we can assign a plus or negative sign to each connected component of $B \setminus S$ so that adjacent components have different signs and for each i-line L that is directed in the increasing i-direction, the line L never pass from a negative region to a positive region. In $2D$ case, we can similarly define i-**monotone** on the faces F of B. $2D$ examples of graph-like and monotone cases are shown in Figure 19. Note that L may keep the same sign as it passes through F/S, or it may change from a positive to a negative region.

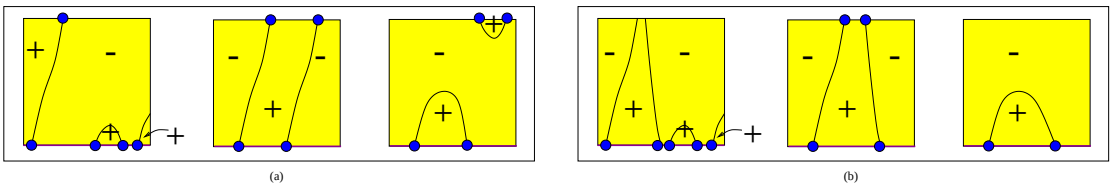


Figure 19: (a) $S \cap B$ is graph-like in B but not monotone, (b) $S \cap B$ is monotone.

Here is an alternative characterization of i-monotone:

LEMMA 5. *Let $B = I_x \times I_y \times I_z$. Then f is z-monotone in B iff there is a continuous function $\phi : I_x \times I_y \to I_z$ such that the graph $gr(\phi) = \{(x, y, \phi(x, y)) : (x, y) \in I_x \times I_y\}$ of ϕ is equal to S in the interior of B, i.e.,*

$$gr(\phi) \cap int(B) = S \cap int(B).$$

The easy proof is omitted. Note that if $(x, y) \in I_x \times I_y$ and $(x, y, \phi(x, y)) \notin S$ then $\phi(x, y)$ must be either $\max I_z$ or $\min I_z$. The continuity of the function ϕ is necessary to ensure monotonicity.

We simply say "graph-like" or "monotone" if i is understood from the context. For specificity, we usually let $i = y$ in illustrations. These definitions also make sense in $2D$ where S is a curve and B is a planar rectangle.

LEMMA 6. *Suppose $S = f^{-1}(0)$ where $f : \mathbb{R}^3 \to \mathbb{R}$. For any box B, if $\frac{\partial f}{\partial i}(p) \neq 0$ for all $p \in B$ then S is i-monotone in B.*

This lemma shows the origin of our monotonicity concept, and the proof of it is immediate. Next, suppose T is the octree produced by our regularized Cxyz algorithm on the input function f. Then for each box B in T which is intersected by $S = f^{-1}(0)$, there is a direction $i = i_B \in \{x, y, z\}$ such that S is i-monotone in B. Let $i : T \to \{x, y, z\}$ denote this (canonical) direction. Hence for each candidate box $B \in T$, we have a fixed direction i, where S is i-monotone in B.

S is **monotone** in T if S is i-monotone in each box B in T for some $i \in \{x, y, z\}$. Let S and $\widetilde{S}$ be two surfaces. We say $\widetilde{S}$ preserves the **monotonicity** of S in T if for any candidate box B in T, if S is i-monotone in B, then $\widetilde{S}$ is also i-monotone on B.

In our proof, we will begin with a surface that is monotone in all the candidate boxes in T, and we will repeatedly modify S to some $\widetilde{S}$ which preserves the monotonicity of S in T. What is important is that we can basically "forget" about the original function f as we do this modification, and we do not have to produce a suitable $\widetilde{f}$ with the property that $\widetilde{f}^{-1}(0) = \widetilde{S}$.

Relative to a surface S, an edge E is **dirty** if $|S \cap E| \geq 2$ or S intersects E tangentially, and a face F is **dirty** if $S \cap F$ contains a loop (i.e., closed curve) or S intersects F tangentially. The opposite of dirty is **clean**. A surface $\widetilde{S}$ is **clean** if every edge and face of T is clean relative to $\widetilde{S}$.

For the correctness[3] of our algorithm, we must modify our algorithm to do special "boundary processing" so that T is clean relative to S on the boundary faces. This processing amounts doing root isolation on the

[3] All our correctness is up to an infinitesimal perturbation of f. It means that our algorithms miss tangential intersections of $S \cap R(T)$, when these components only occur on the boundary of $R(T)$. On the other hand, tangential intersections of $S \cap R(T)$ in the interior of $R(T)$ are excluded by explicit assumption.

edges on $\partial R(T)$, followed by the $2D$ Cxy algorithm on the boundary of $R(T)$. These $1D$ and $2D$ processing are performed by splitting boxes in the octree. Boundary processing in the Cxyz Algorithm is similar to the Cxy Algorithm. For the following part, we will assume that the surface S intersects $\partial R(T)$ cleanly.

Note that for a box B, $S \cap B$ might be comprised of several connected components, but one can prove that (in the Regularized Cxyz algorithm) all these components must belong to the same (global) component of $S \cap R(T)$. Note that each component of S can give rise to zero, one, or more components of $S \cap R(T)$.

¶17. Partial Order on Pairs We fix the usual octree T and f that defines the surface $S = f^{-1}(0)$. Let $\mathcal{P}(S)$ denote the set of all **pairs** of points $\{p, q\}$ such that there is an edge E of T, $\{p, q\} \subseteq E \cap S$ and the segment $[p, q]$ intersects S in an even number of points. *Note that the definition of pair in Cxyz Algorithm is more general than the definition of convergent pair in Cxy Algorithm.* We assume that $\mathcal{P}(S)$ is a finite set. We also regard the empty set $\mathcal{O}$ as a special element of $\mathcal{P}(S)$; all other pairs are called **non-empty pairs**. We say $\mathcal{P}(S)$ is **trivial** if its only member is $\mathcal{O}$.

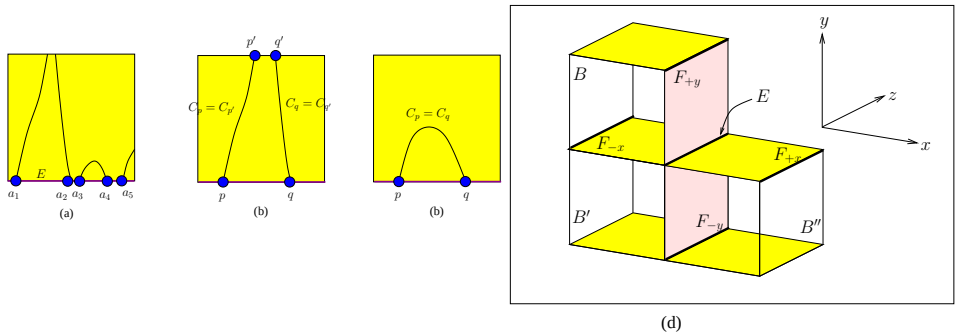


Figure 20: (a) Pairs on edge E, (b) $\{p, q\} \succ \{p', q'\}$, (c) $\{p, q\} \succ \mathcal{O}$

Example: Figure 20(a) shows an edge E with 5 intersection points with S. There are 6 pairs on E given by

$$\{a_1, a_2\}, \{a_2, a_3\}, \{a_3, a_4\}, \{a_4, a_5\}, \{a_1, a_4\}, \{a_2, a_5\}.$$

In general, an edge with n intersection points with S determines $p(n)$ pairs where $p(0) = 0$ and for $n \geq 1$, $p(n) = p(n-1) + \lceil (n-1)/2 \rceil$. So $p(1) = 0, p(2) = 1, p(3) = 2, p(4) = 4, p(5) = 6$.

We define a relationship between pairs of $\mathcal{P}(S)$. For any face F of T, we consider the connected curve components of $F \cap S$. If o is a point in $S \cap \partial F$, let C_o denote the connected component of $F \cap S$ that has o as one endpoint. Given two pairs $\{p, q\}, \{p', q'\}$, we define the relation

$$\{p, q\} \succ \{p', q'\} \,(\textbf{mod}\, F) \tag{4}$$

if $d(p, q) > d(p', q')$ and F has two opposite edges, E and E' such that $\{p, q\} \subseteq E$ and $\{p', q'\} \subseteq E'$, and the connected components of $S \cap F$ has this property: $C_p = C_{p'}$ and $C_q = C_{q'}$. Further define

$$\{p, q\} \succ \mathcal{O} (\textbf{mod}\, F) \tag{5}$$

if $\{p, q\} \subseteq \partial F$ and $C_p = C_q$. Both the relations (4) and (5) are illustrated in Figure 20(b,c).

For pairs $A, B \in \mathcal{P}(S)$, define the relation $A \succ B$ if there exists a face F such that $A \succ B (\textbf{mod}\, F)$. Let $\succeq$ denote the reflexive transitive closure of $\succ$: $P \succeq Q$ iff $P = Q$ or there is a finite sequence of pairs where $P = P_0 \succ P_1 \succ \cdots \succ P_k = Q$.

LEMMA 7. *The relation $(\mathcal{P}(S), \succeq)$ is a partial ordering on $\mathcal{P}(S)$*

Proof. We check three properties. Let $A, B, C \in \mathcal{P}(S)$. Reflexivity: $A \succeq A$ (by definition). Symmetry: $A \succeq B$ and $B \succeq A$ implies $A = B$. This is true if A or B is equal to $\mathcal{O}$. Otherwise, if $A \neq B$, we see that $A \succeq B$ implies $d(A) > d(B)$. Similarly, $B \succeq A$ implies $d(B) > d(A)$, contradiction. Transitivity: $A \succeq B \succeq C$ implies $A \succeq C$. This follows from the definition of $\succeq$. **Q.E.D.**

If $A \succeq B$, we say B is "smaller" than A and we are interested in minimal elements in this partial order.

Intuitively, $\mathcal{O}$ is the unique minima in $\mathcal{P}(S)$. Towards proving this result, we need a useful property of our octree T:

LEMMA 8. *Let S be a surface which is monotone in T, and E be any non-boundary edge of T such that $|S \cap E| \geq 2$. Assume (wlog) that E is parallel to the z-axis, and the four faces bounded by E are F_x, F_{-x}, F_y and F_{-y}, as in Figure 20(d). Then either S is x-monotone on $F_x \cup F_{-x}$, or S is y-monotone on $F_y \cup F_{-y}$.*

Proof. Suppose S is not x-monotone on F_{-x}. Consider the box B lying above F_{-x}. Since S cannot be z-monotone in B (because E intersects S in more than one point) and it cannot be x-monotone (since S is not x-monotone on F_{-x}), we conclude that S must be y-monotone in B. The same reasoning implies that S must be y-monotone in the box B' below F_{-x}. This concludes that S must be y-monotone on $F_y \cup F_{-y}$. **Q.E.D.**

LEMMA 9. *The empty set $\mathcal{O} \in \mathcal{P}(S)$ is the unique minimal element of $\mathcal{P}(S)$.*

See [13] Lemma 23 for the proof of this lemma.

¶18. Cleansing Strategy We are going to transform S to another surface $\widetilde{S}$ that is clean relative to T. We do this by transforming S isotopically to $\widetilde{S}$. A difficult problem in this transformation is that it is very hard to keep track of the nice properties of the original f with respect to T. For instance, we know that each candidate box B of T must satisfy $C^f_{xyz}(B)$. We first overview the cleansing processes:

1. First, we clean all faces. Here we can exploit the original property of f. Because f is monotone in some coordinate direction in each box B, there cannot be loops in two adjacent faces of B. Moreover, the set of all such loops has a natural nesting partial order in each coordinate direction.

2. Next, assuming all the faces are clean, we can clean edges. Actually, we cannot clean an entire edge at once, but we remove pairs from $\mathcal{P}(S)$, one pair at a time. Let $S = S_0$ and we construct a new surface S_{i+1} from S_i by removing one pair. The fact that $\mathcal{P}(S_{i+1})$ is a proper subset of $\mathcal{P}(S_i)$ allows us to preserve the partial order that is induced from the original $\mathcal{P}(S) = \mathcal{P}(S_0)$. We show that each pair removal does not introduce any loop. So, at the end of this process, we have a surface S_k that is clean, and isotopic to S.

We next give details of these cleansing routines.

¶19. Cleaning Faces Consider the set of loops of S in faces of our octree T. Denote this set by $\mathcal{L}(S)$, and as before, introduce an artificial element $\mathcal{O}$ in $\mathcal{L}(S)$. We say $\mathcal{L}(S)$ is **trivial** if its only member is $\mathcal{O}$. We also assume that $\mathcal{L}(S)$ is a finite set.

Let L, L' be two distinct loops of $\mathcal{L}(S)$, and they lie on the boundary of a common box B. Let C_L denote the connected component of $S \cap B$ that is bounded by L. Wlog, let f be y-monotone in B. This implies that L and L' can only lie on y-faces of B. These two y-faces can be distinct or the same. We write $L \succ L'$(**mod** B) if $C_L = C_{L'}$ and the y-projection of L' is contained in the interior of the y-projection of L (by y-projection, we mean the projection onto the $y = 0$ plane). Note that either $L \succ L'$ or $L' \succ L$ must occur because f is y-monotone in B. This ensures that we have a global partial ordering on $\mathcal{L}(S)$. This global property is derived from our original function f, and is critical for our proof. We must carry some of this information along in the induction, even after we have transformed f. Also, observe that the partial ordering can be naturally partitioned into three subrelations $\mathcal{L}(S) = \mathcal{L}_x(S) \cup \mathcal{L}_y(S) \cup \mathcal{L}_z(S)$, corresponding to the three coordinate directions.

Note that there can be several loops $L^{(i)}$ ($i = 1, 2, \ldots$) such that $L \succ L^{(i)}$. These $L^{(i)}$ can lie in the same face as L or in the opposite face. A fundamental property of this relation is this:

LEMMA 10. *For each loop L', there is at most one L such that $L \succ L'$.*

Proof. Say these loops lie on y-faces. If $L \succ L'(\textbf{mod}\, B)$, then the y-projection of L' is in the interior of the y-projection of L. Moreover, the component $C_L \subseteq B \cap S$ projects into the interior of L. If $L_0 \succ L'$ for some loop L_0, then we see that $C_{L_0} = C_L$ and $L_0 = L$. **Q.E.D.**

In the special case where the boundary of C_L is connected, then we have $\partial C_L = L$. In this case, we write $L \succ \mathcal{O}(\textbf{mod}\, B)$. This produces a partial order on the set of all loops (treating $\mathcal{O}$ as a special loop). Moreover, $\mathcal{O}$ is the unique minimum in this partial order. If $L \succ \mathcal{O}(\textbf{mod}\, B)$, we call $C_L \subseteq B$ a **cap**. Our transformation for loops amounts to repeated removing caps. Initially, let $S_0 = S$. We will define a sequence of surfaces, $S_1, S_2, \ldots$ such that the loops $\mathcal{L}_y(S_{i+1})$ is a proper subset of $\mathcal{L}_y(S_i)$ for each i.

Let $L \succ \mathcal{O}$ in $\mathcal{L}_y(S_i)$ lies in the face F and suppose B' is another box that is bounded by F. We can easily define a $(B \cup B')$-isotopy to transform S_i to S_{i+1} in which L does not occur in $\mathcal{L}_y(S_{i+1})$, but all the other loops of $\mathcal{L}_y(S_i)$ remains. Of course, if $L' \succ L$ in $\mathcal{L}_y(S_i)$, the removal of L may induce the new relation $L' \succ \mathcal{O}$ in $\mathcal{L}_y(S_{i+1})$.

Eventually, $\mathcal{L}_y(S_i)$ becomes trivial and contains only $\mathcal{O}$. We can independently repeat this argument on $\mathcal{L}_x(S_i)$ and $\mathcal{L}_z(S_i)$. All faces are clean when $\mathcal{L}(S)$ is empty.

¶20. Semi-loops and Bases We now have clean faces. To discuss the cleansing of edges, we need some additional concepts. Suppose F is a face and the surface intersects F in a number of curves, including loops (i.e., curve components with no endpoints). A non-loop curve component C whose two endpoints lie on the same edge E of F is called a **semi-loop** (E.g., C on F_{y+} or C' on F_{x+} in Figure 21). If p, q are the two endpoints of C, we call the line segment $[p, q] \subseteq E$ the **base** of the semi-loop C. Suppose F' is another face that is bounded by E, and F' has another semi-loop C' sharing the same base as C. Then we say C and C' are **linked** by this base. Suppose C, C' are linked semi-loops, there are two possibilities: they could be coplanar (Figure 21, C' and C'') or they may lie on a pair of perpendicular planes (Figure 21, C and C'). In general, a base can be shared by up to 4 semi-loops. The next lemma shows that this will not happen.

LEMMA 11 (NO FOURSOMES). *Let S be a surface which is monotone in T. Then at most 3 semi-loops can be linked together.*

See [13] Lemma 25 for the proof of this lemma. REMARK: in subsequent transformation of S, "NO FOURSOMES" property will be preserved (as we will see).

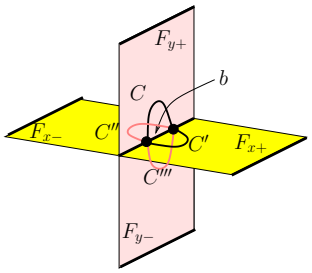


Figure 21: Impossibility of 4-linked semi-loops.

LEMMA 12 (NO HOLES). *Let S be the surface after the face cleaning process (note that S is monotone in T). Let $C, C' \subset S$ be linked semi-loops on the boundary of B. Let $P \subseteq S \cap B$ be a surface patch in B (i.e., P is a connected component of $S \cap B$). If $C \cup C' \subseteq \partial P$, then $\partial P = C \cup C'$. In other words, P is topologically a disc.*

Proof. Let B be the box containing C and C' in Figure 21. S must be monotone in x or y-direction in B. Wlog, let us assume that S is monotone in y-direction in B. Since P is converging in $y+$ direction, the projection of $P \cap int(B)$ onto F_{x+} must lie within C'. Also, $S \cap B$ contains no loop on the faces of B. So we can conclude that P is a topological disc and $\partial P = C \cup C'$. **Q.E.D.**

In other words, this lemma says that P cannot contain any holes as illustrated in Figure 22.

From the proof of Lemma 12, and the fact that a connected subset of an i-block can be viewed as a rectangular box in which S is monotone in i-direction, it is easy to see that the following lemma is also correct:

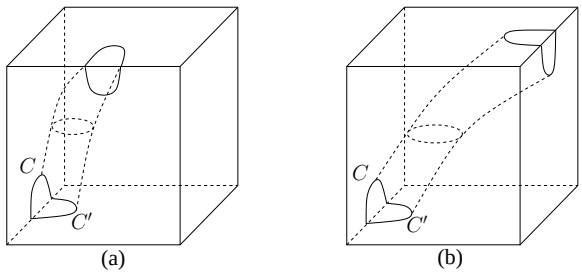


Figure 22: Examples of holes.

LEMMA 13 (NO HOLES 1). *Let $\mathcal{B}$ be a connected subset of an i-block, and S be a surface that is monotone in T which intersects the faces of $B \in \mathcal{B}$ cleanly. Let $C \subseteq S \cap \partial(\cup_{B\in\mathcal{B}} B)$ be a closed curve, and $P \subseteq S \cap \mathcal{B}$ be a connected component. If $C \subseteq \partial P$, then $\partial P = C$. In other words, P is topologically a disc.*

REMARK: in subsequent transformation of S, this property will also be preserved (as we will see).

¶21. Cleaning Edges via Base Removal Operations Let us retain the notations of Figure 20 relative to an edge E containing a pair $\{p, q\}$. We call a pair $\{p, q\}$ **penultimate minimum** (or $\{p, q\} \succ_* \mathcal{O}$) if for any pair P, $\{p, q\} \succ P$ implies $P = \mathcal{O}$. If $\{p, q\} \succ_* \mathcal{O}$ and for exactly i of the faces $F \in \{F_x, F_{-x}, F_y, F_{-y}\}$, $\{p, q\} \succ \mathcal{O}(\textbf{mod}\, F)$, then we say $\{p, q\} \succ_i \mathcal{O}$. Note that if $\{p, q\} \succ_i \mathcal{O}$, then $i \geq 1$. In other words, $\{p, q\} \succ_0 \mathcal{O}$ is not possible. We call a base $b = [p, q]$ a **penultimate minimum base** if $\{p, q\}$ is a penultimate minimum pair. Clearly, penultimate minimum base is a base of some semi-loops.

We will remove one penultimate minimum pair in $\mathcal{P}(S)$ each time. Let $S = S_0 = f^{-1}(0)$ and suppose we construct a new surface S_{i+1} from S_i by removing one pair from $\mathcal{P}(S_i)$. The fact that $\mathcal{P}(S_{i+1})$ is a proper subset of $\mathcal{P}(S_i)$ allows us to preserve the partial order that is induced from the original $\mathcal{P}(S) = \mathcal{P}(S_0)$. Our removing of penultimate minimum pairs will not change the partial order in $\mathcal{P}(S)$. In each step $\mathcal{P}(S_i) = \mathcal{P}(S_{i+1}) \cap \{\{p_i, q_i\}\}$ where $\{p_i, q_i\}$ is the penultimate minimum pair which we remove at step i. The removing only creates new relations of the form $\{p, q\} \succ \mathcal{O}$ where $\{p, q\} \succ \{p', q'\}$ in $\mathcal{P}(S_i)$.

The next lemma shows that if a base $b = [p, q]$ is a penultimate minimum base and $\{p, q\} \succ_2 \mathcal{O}$, then the two linked semi-loops must lie on a pair of perpendicular planes:

LEMMA 14. *Let S be a surface that is monotone in T, and $\{p, q\}$ be a pair of $S \cap T$. Consider two distinct faces F_s and F_v in Figure 20 where $\{s, v\} \subset \{x, -x, y, -y\}$. If $\{p, q\} \succ_2 \mathcal{O}$ where $\{p, q\} \succ \mathcal{O}(\textbf{mod}\, F_s)$ and $\{p, q\} \succ \mathcal{O}(\textbf{mod}\, F_v)$, then $\{s, v\} \neq \{x, -x\}$ and $\{s, v\} \neq \{y, -y\}$.*

Proof. If $\{p, q\} \succ \mathcal{O}(\textbf{mod}\, F_x)$ and $\succ \mathcal{O}(\textbf{mod}\, F_{-x})$, and curves $C_p, C_q \subseteq S \cap (F_{-y} \cup F_y)$ are the connected components that passes through p and q, then C_p and C_q must be different components in $F_{-y} \cup F_y$. Since $\{p, q\}$ is a penultimate minimum pair, S can not be y-monotone in $F_y \cup F_{-y}$. From Lemma 8, we know that S is x-monotone in $F_x \cup F_{-x}$, which contradicts the fact that $[p, q]$ is the base of two coplanar linked semi-loops on $F_x \cup F_{-x}$. **Q.E.D.**

Suppose $P \succ_i \mathcal{O}$ where P is a pair. We already noted that $i = 0$ is not possible. From Lemma 11, if we can preserve the monotonicity of S during the surface transformation (which will be proven later), then $i = 4$ is also impossible. So the only possibilities for i is $1, 2$ and 3. Because of Lemma 14, a penultimate minimum base b could have three possibilities, as shown in Figure 23(I), (II) and (III). Note that if b is not a penultimate minimum base, Figure 23(III'') might arise.

Let b be a penultimate minimum base for some semi-loop. To "remove" b means to simultaneously remove all the semi-loops that share the base b. Since there are only three possibilities, so there are three distinct base removal operations. This is shown in Figure 23. In Figure 23 $(I) \to (I')$, we push down the part of semi-loop component to form a "tunnel" below the edge E. In Figure 23 $(II) \to (II')$, we push the topological disc component bounded by the two semi-loops in both $x-$ and $y-$ directions to eliminate it. In Figure 23 $(III) \to (III')$, we push down the topological disc component bounded by the three semi-loops to remove the it. Note that these operations are well-defined: this depends on the fact that in each box B that contains

a pair of linked semi-loops C and C', the surface patch bounded by $C \cup C'$ is a topological disc (i.e., the "NO HOLES" property in Lemma 12 holds as long as we preserve the monotonicity of the surface during our operations, which will be proven in the following part).

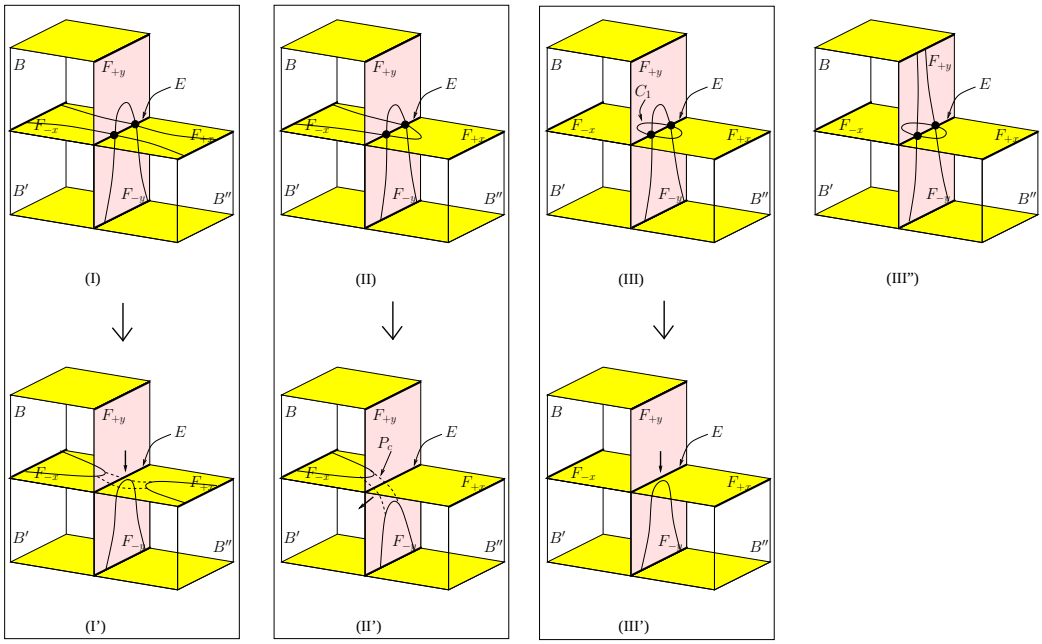


Figure 23: Three Base Removal Operations.

We next describe some properties that our transformation preserves. Let T be an octree and V_T be the set of all corners of the boxes in T. Let S, S' be two surfaces. We say S is **compatible** with S' (respect to T) iff there exist an isotopy $I : \mathbb{R}^3 \times [0,1] \to \mathbb{R}^3$, s.t. $I(\cdot, 0)$ is the identity; $I(S,1) = S'$ and $\forall t \in [0,1]$, $I(S,t) \cap V_T = \emptyset$.

LEMMA 15. *The face cleaning operations and the base removal operations preserve the compatibility of S in T.*

Proof. The correctness of this lemma is based on the nature of our operations: we never transform the surface "across" any corners in T. **Q.E.D.**

LEMMA 16 (Surface Monotonicity Preservation). *Base removal operations preserve the monotonicity of S in T.*

See [13] Lemma 30 for the proof of this lemma.

The next example shows that if we remove the bases in arbitrary order, we might create holes within the boxes. Let $b1$ be the smallest base in the box B in Figure 24(I). Assume S is y-monotone in B, since our operation preserves the monotonicity, we have the length of $b3$ is less than the length of $b4$.. If we remove the bases in arbitrary order, we might remove $b1$ and $b4$ before $b2$ and $b3$, which results in a hole as shown in Figure 24(I').

LEMMA 17. *The face cleaning operations do not induce new dirty faces, and the base removal operations do not induce new dirty edges and dirty faces.*

Proof. It is clear that the face cleaning operations do not induce new dirty faces, and the base removal operations do not induce new dirty edges. We will show that the base removal operations do not induce new dirty faces. Let R be a base removal operation which removes a penultimate minimum pair b and induces a new loop l on a face F. Then before the operation, l was a semi-loop with the base b. This contradicts the fact that R removed all the semi-loops that share the same base b. **Q.E.D.**

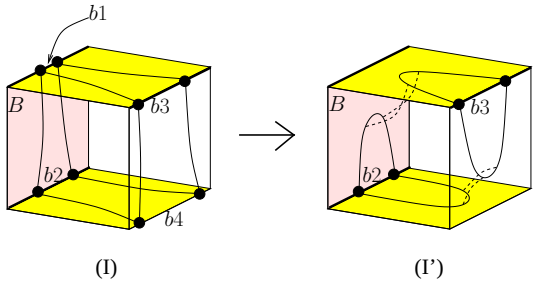


Figure 24: Removing bases in arbitrary order might create holes.

The above base removal process halts only when $\mathcal{P}(S)$ is empty. At this point, all faces and edges are clean relative to T. From the analysis above, we have the following theorem:

THEOREM 18. *Let T be the octree produced by our Regularized Cxyz Algorithm. There $\exists \widetilde{S}$, s.t.*
(1) $\widetilde{S} \simeq S(\textbf{mod}\, R(T))$.
(2) $\widetilde{S}$ is compatible with S respect to T.
(3) $\widetilde{S}$ intersects T cleanly.
(4) $\widetilde{S}$ preserves the monotonicity of S within each candidate box of T.

Proof. We first clean the faces, then we clean the edges. From Lemma 15, Lemma 16 and Lemma 17, and the fact that each operation is an isotopic transformation, the resulting $\widetilde{S}$ satisfies all the properties in this theorem. **Q.E.D.**

THEOREM 19. *Let G be the mesh we construct by the Regularized Cxyz Algorithm, then $G \simeq S(\textbf{mod}\, R(T))$.*

Proof. Based on the construction phase of our algorithm, for each alternating block $\mathcal{B}$, $\widetilde{S} \cap \partial(\cup\mathcal{B})$ "agrees" with $G \cap \partial(\cup\mathcal{B})$. From Lemma 13, we know that $\widetilde{S}$ is isotopic to G within each block. So $G \simeq \widetilde{S}(\textbf{mod}\, R(T))$. From Theorem 18, we have $G \simeq \widetilde{S} \simeq S(\textbf{mod}\, R(T))$. **Q.E.D.**

A.5 Correctness of Balanced Cxyz Algorithm

Let T be the octree produced by our Balanced Cxyz Algorithm. Similar to the correctness proof of the Regularized Cxyz Algorithm, we will first transform the input surface $S = f^{-1}(0)$ to another surface $\widetilde{S}$ which has some nice properties.

In the correctness proof of the Regularized Cxyz Algorithm, we separately defined the partial orders for loops and pairs of S in T. In the Balanced Cxyz Algorithm, we need to define the partial order for the combination of all loops and pairs. The reason is that a loop might be "blocked" by pairs (an example is shown in Figure 25(I)), and we need to remove the pairs first in order to remove the loop. Also, a pair might be "blocked" by loops ,as shown in Figure 25(II) (we do not have such problem in the Regularized Cxyz Algorithm since the loops are removed before pairs).

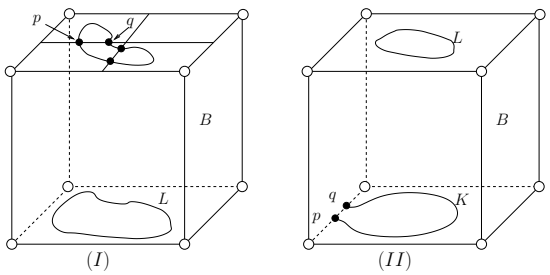


Figure 25: Partial order between a loop and a pair.

We define the new partial order for the set of $\mathcal{P}(S) \cup \mathcal{L}(S)$, where $\mathcal{P}(S)$ is the set of all pairs of $S \cap T$, and $\mathcal{L}(S)$ is the set of all loops of $S \cap T$ (see ¶17 and ¶19). The partial order between loops and between pairs are the same as the partial order defined in the Regularized Cxyz Algorithm: let $\prec_P \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$ be the partial order defined for pairs, and $\prec_L \subseteq \mathcal{L}(S) \times \mathcal{L}(S)$ be the partial order defined for loops. We need to define a partial order on the set $\mathcal{P}(S) \cup \mathcal{L}(S)$.

Let B be a box with monotone direction y. Let L be a loop on the bottom face of B and $\{p, q\}$ be a pair on the top face of B. If the y-projection of $\{p, q\}$ is contained within the y-projection of L, we say $\{p, q\} \prec L$ (as shown in Figure 25(I)). In order to remove L, we need to remove $\{p, q\}$ first. We can similarly define such relations in x and z directions. Let $\prec_{PL} \subseteq \mathcal{P}(S) \times \mathcal{L}(S)$ be all the relations so defined. Similarly, we can define $\prec_{LP} \subseteq \mathcal{L}(S) \times \mathcal{P}(S)$: let $\{p, q\}$ be a pair, and K be a semi-loop whose base is $[p, q]$. If there exist a loop L which lies in the same box B as K, and the i-projection of L (for some $i \in \{x, y, z\}$) lies in the interior of the i-projection of K, we say $L \prec \{p, q\}$ (as shown in Figure 25(II)).

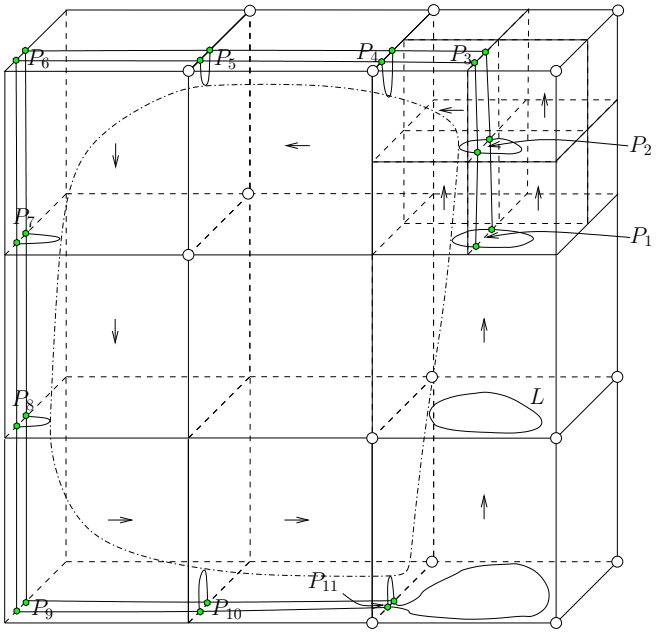


Figure 26: Example of a loop in $\prec_P \cup \prec_L \cup \prec_{PL} \cup \prec_{LP}$.

In the Regularized Cxyz Algorithm, we removed all loops before we remove pairs. But in the Balanced Cxyz Algorithm, we are forced to intermix pair removal with loop removal because of the relations in $\prec_{PL}$ and $\prec_{LP}$. However, if we look at the relation $\prec_P \cup \prec_L \cup \prec_{PL} \cup \prec_{LP}$, we do not obtain a partial order on $\mathcal{P}(S) \cup \mathcal{L}(S)$ (see Figure 26: the green points form pairs, and the arrows show the monotone direction of the boxes. It is possible that $L \prec P_{11} \prec \ldots \prec P_1 \prec L$, which forms a loop).

Our solution is to define a partial order based only on $\prec_{Bal} := \prec_P \cup \prec_L \cup \prec_{PL}$. This is clearly a partial order on $\mathcal{P}(S) \cup \mathcal{L}(S)$.

LEMMA 20 (DAG). *The partial order relationship $\prec_{Bal}$ forms a DAG G_p where the pairs and loops are the nodes of G_p and the partial order relations are the (directed) edges of G_p.*

Why is this a solution? As usual, we plan to inductively remove elements from $\mathcal{P}(S) \cup \mathcal{L}(S)$, which are minimal relative to $\prec_{Bal}$. The possible complication arises when we want to remove a pair $\{p, q\}$ where $L \prec_{LP} \{p, q\}$ for some loop L. It turns out, we can remove $\{p, q\}$ without first removing L provided that we generalize our previous base removal operation as follows: to remove a pair $\{p, q\}$, we will remove all semi-loops K whose base is $[p, q]$. There are two possible situations: (A) If there is a loop L s.t. $L \prec_{LP} \{p, q\}$, then we know that $[p, q]$ is the base of a semi-loop K where the i-projection of L (for some $i \in \{x, y, z\}$) lies in the interior of K. In this case, we transform the surface S so that $\{p, q\}$ is removed from $\mathcal{P}(S)$, and a new loop K' appears in $\mathcal{L}(S)$. And moreover, $L \prec K' \in \prec_L$. See Figure 27 $(II*) \rightarrow (II*')$ and $(III*) \rightarrow (III*')$ for

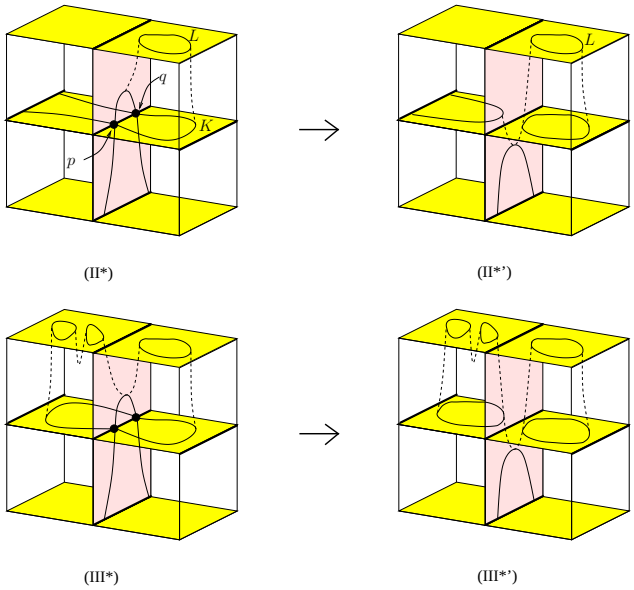


Figure 27: Universal Base Removal Operations.

the illustration of this operation. Note that there might be more than one such loops L. (B) If no such loop L exists, then the operation is defined as in the Regularized Cxyz Algorithm. Similar to the proof of Lemma 16, we can prove that those two generalized operations also preserve the surface monotonicity of S in T. Based on the correctness analysis in the Regularized Cxyz Algorithm, we have the following (similar) theorem for the Balanced Cxyz Algorithm:

THEOREM 21. *Let T be the octree produced by our Balanced Cxyz Algorithm. There $\exists \widetilde{S}$, s.t.*
(1) $\widetilde{S} \simeq S(\textbf{mod}\, R(T))$.
(2) $\widetilde{S}$ is compatible with S respect to T.
(3) $\widetilde{S}$ intersects T cleanly.
(4) $\widetilde{S}$ preserves the monotonicity of S within each candidate box of T.

Proof. The correctness of this theorem follows from the analysis of the face cleaning and edge cleaning processes. **Q.E.D.**

In the Regularized Cxyz Algorithm, we proved Lemma 13. We have a similar result in the balanced algorithm:

LEMMA 22 (NO HOLES 2). *Let $\widetilde{S}$ be the surface described in Theorem 21 and $\mathcal{B}$ be a connected subset of an i-block. Let C be a closed curve which is the intersection of $\widetilde{S}$ with $\partial(\cup\mathcal{B}_{B\in\mathcal{B}})$. Let $P \subseteq \widetilde{S} \cap \mathcal{B}$ be a surface patch in $\mathcal{B}$ (i.e., P is a connected component of $\widetilde{S} \cap \mathcal{B}$). If $C \subseteq \partial P$, then $\partial P = C$. In other words, P is topologically a disc.*

Proof. The correctness of this lemma follows from the facts that $\widetilde{S} \cap \mathcal{B}$ is monotone in $\mathcal{B}$, and $\widetilde{S}$ intersects $\mathcal{B}$ cleanly. The proof is similar to the proof of Lemma 12. **Q.E.D.**

From Lemma 22, it is easy to see that $\widetilde{S} \cap B$ is a set of topological discs for each candidate box B.

THEOREM 23. *The mesh G constructed by our Balanced Cxyz Algorithm is isotopic to $\widetilde{S}$ within each i-block $\mathcal{B}$ of T. In other words, $G \simeq \widetilde{S} \simeq S(\textbf{mod}\, R(T))$.*

Proof. From Theorem 21, it is easy to see that $\widetilde{S}$ intersects the boundary of $\mathcal{B}$ cleanly. Our construction rule guarantees that $G \cap \partial(\cup\mathcal{B})$ "agrees" with $\widetilde{S} \cap \partial(\cup\mathcal{B})$. And each connected component of $G \cap \mathcal{B}$ is a topological disc. So based on Lemma 22, we have $G \cap \mathcal{B} \simeq \widetilde{S} \cap \mathcal{B}$. **Q.E.D.**