# Isotopic Approximation within a Tolerance Volume

Manish Mandad David Cohen-Steiner Pierre Alliez
Inria Sophia Antipolis - Méditerranée

## Abstract

We introduce in this paper an algorithm that generates from an input tolerance volume a surface triangle mesh guaranteed to be within the tolerance, intersection free and topologically correct. A pliant meshing algorithm is used to capture the topology and discover the anisotropy in the input tolerance volume in order to generate a concise output. We first refine a 3D Delaunay triangulation over the tolerance volume while maintaining a piecewise-linear function on this triangulation, until an isosurface of this function matches the topology sought after. We then embed the isosurface into the 3D triangulation via mutual tessellation, and simplify it while preserving the topology. Our approach extends to surfaces with boundaries and to non-manifold surfaces. We demonstrate the versatility and efficacy of our approach on a variety of data sets and tolerance volumes.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary representations;

**Keywords:** Isotopic approximation, tolerance volume, mesh refinement, mesh simplification, intersection-free, mutual tessellation

## 1 Introduction

Faithful approximation of complex shapes with simplicial meshes is a multifaceted problem, involving geometry, topology and their discretization. This problem has received considerable interest due to its wide range of applications and the ever-increasing accessibility of geometric sensors. Increased availability of scanned geometric models, however, does not mean improved quality: while many practitioners have access to high-end acquisition systems, a recent trend is to replace these expensive tools with a combination of consumer-level acquisition devices. Measurement data generated by, and merged from, these heterogeneous devices are reputedly unfit for direct processing. Similarly, the growing variety of geometry processing tools often increase the net amount of defects in data: Conversion to and from various geometry representations often degrades the input, and rare are the algorithms that have stronger guarantees on their output than they have requirements on their input. As we deal with ever finer discretizations to capture intricate geometric features, this issue of offering strict geometric guarantees to be robust to the occurrence of artifacts is becoming more prevalent.

Geometric guarantees usually refer to upper bounds on the approximation error and to the absence of self-intersections. Topological guarantees refer to homotopy, homeomorphism or isotopy. In our context *isotopy* means that there exists a smooth deformation that maps one shape to another while maintaining a homeomorphism between the two. Surface meshes with such guarantees are required for artifact-free rendering, computational engineering, reverse engineering, manufacturing and 3D printing. While *geometric* simplification can reduce the number of primitives, *topological* simplification can repair holes and degeneracies in existing discretizations. Combined, the two may also be used for reconstructing clean shapes from raw geometric data such as point sets or polygon soups.

### 1.1 Related Work

A vast array of methodologies has been proposed for shape approximation over the years, ranging from decimation to optimization through clustering and refinement. Fewer, however, provide error bounds. In addition, they only apply to specific types of input geometry, and often fail to satisfy geometric *and* topological guarantees as we now review.

[Agarwal and Suri 1998] proposed a polynomial-time approximation algorithm with guaranteed maximum error and minimum number of vertices, but this algorithm is too complex to be practically relevant. Approximation with bounded error has also been targeted through clustering [Kalvin and Taylor 1996], mesh decimation [Cohen et al. 1996; Klein et al. 1996; Guéziec 1996; Ciampalini et al. 1997; Cohen et al. 2003; Botsch et al. 2004; Ovreiu et al. 2012] or a combination of both [Zelinka and Garland 2002]. In general the error metric considered is the one-sided Hausdorff distance to the input mesh, but the normal deviation has also been considered [Borouchaki and Frey 2005]. In general these approaches are not generic enough to handle heterogeneous input data, and they are not designed to guarantee a valid, intersection-free output. Guarantees for intersection-free output can be obtained by preventing intersections during mesh decimation [Gumhold et al. 2003]) but this approach is not sufficient when the input itself self-intersects. Also, searching for the locus of points in space that avoids intersections when applying a decimation operator, and tunneling out of situations where every operator is forbidden is often too labor-intensive to be considered a practical solution. Another class of approaches based on Delaunay filtering and refinement, instead, provide intersection-free approximations by construction [Boissonnat and Oudot 2005]. Unfortunately, these approaches generate only isotropic meshes and thus do not target very coarse approximations and, as such, cannot be used for shape simplification.

When dealing with imperfect or heterogeneous data, methods involving repairing [Ju 2004; Bischoff et al. 2005; Attene 2010], conversion [Shen et al. 2004], or reconstruction [Hornung and Kobbelt 2006; Kazhdan et al. 2006] are designed to generate clean meshes, but do not yield low-polygon-count approximations with bounded error. Other approaches target only hole filling and do not remove the self-intersections [Dey and Goswami 2003]. A great variety of methods have been proposed for surface reconstruction. Among them, those which come with theoretical guarantees of isotopy [Amenta and Bern 1998; Amenta et al. 2000; Boissonnat and Cazals 2000; Dey 2006] assume that the sampled surface is smooth. Surface reconstruction from noisy point sets [Dey and Sun 2005; Dey 2006] has also been investigated well. A recent approach deals with boundaries but does not handle noisy data [Dey et al. 2009]. In addition, all these methods generate isotropic meshes, overly com-

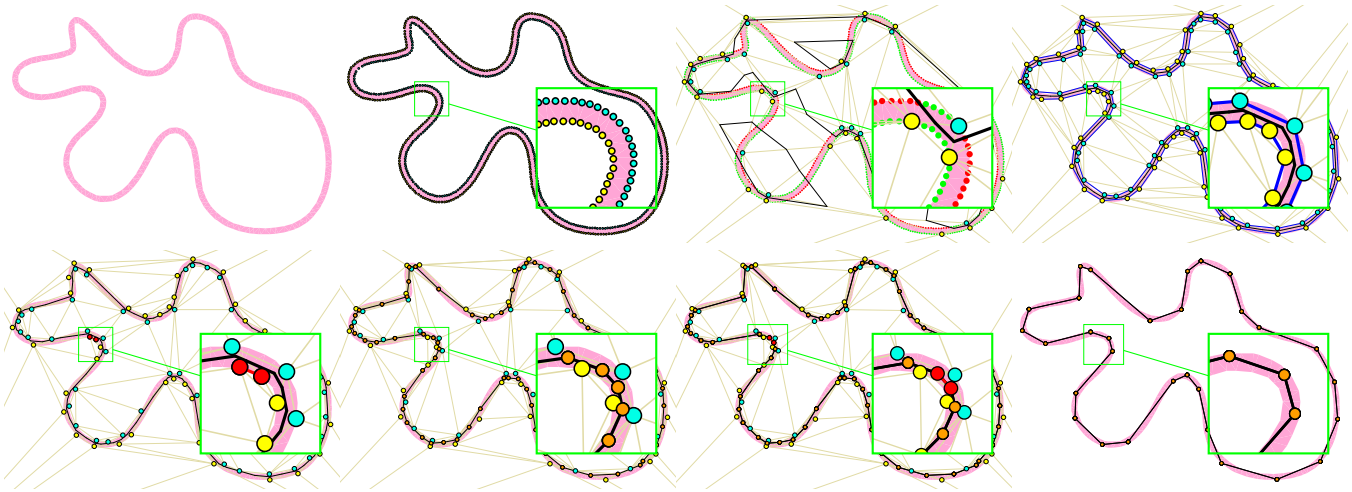


**Figure 1:** *Overview of our algorithm. Top: input tolerance $\Omega$, sampling of $\partial\Omega$, mesh refinement by inserting a subset of the sample points, and topology condition met. Samples that are well classified are depicted in green, and in red otherwise. The boundary of the simplicial tolerance volume $\partial\Gamma$ is depicted with blue edges. Bottom: simplification of $\partial\Gamma$, mutual tessellation of zero-set, simplification of zero-set, and final output.*

plex, which would require another algorithm for simplification with geometric guarantees.

### 1.2 Positioning

In $\mathbb{R}^3$, [Chazal and Cohen-Steiner 2004] showed that when seeking a homeomorphic approximation $S'$ of a connected surface $S$, a simple topological condition is sufficient to guarantee that the two surfaces are isotopic. If $S$ and $S'$ are homeomorphic, then $S$ and $S'$ are isotopic if $S'$ is contained in a *topological thickening* of $S$ and separates the boundary components of this thickening. In this paper we contribute a constructive approach for this theoretical result in the form of an algorithm that matches these conditions in order to ensure that the output surface mesh is an isotopic approximation. We state the problem as follows. The input is a tolerance volume $\Omega$ (Figure 1, top left) that is a topological thickening of a surface $S$ which we want to approximate. By topological thickening of $S$ we mean a compact subset of $\mathbb{R}^3$ homeomorphic to $S \times [0, 1]$. Our goal is to generate as output a surface triangle mesh located within $\Omega$, isotopic to the boundary components of $\Omega$, and with a low triangle count. This approximation problem was originally stated by Klee for polytopes in arbitrary dimensions. In 2D, the problem is commonly referred to as the *minimum nested polygon* problem, and has been investigated well [Aggarwal et al. 1985]. The 3D instance of this problem, referred to as *minimum nested polyhedron* problem has been shown to be NP-hard [Agarwal and Suri 1998].

Despite being a long standing problem, there is still no robust and practical solution to this enduring scientific challenge. Yet, it is both relevant to, and timely for, the increasing variety of industrial applications that involve raw geometric data. In this paper, we develop an algorithm for the above problem that yields approximations with very low triangle count, while enjoying topological guarantees under relatively mild assumptions on the tolerance volume. Note that while the assumption that $\Omega$ is a proper thickening makes the analysis easier, it is not always necessary and our approach may also work when boundary components of $\Omega$ have, for instance, additional spurious handles. We also extend our algorithm to non-closed and non-manifold surfaces. If $\Omega$ is not provided as input, we may generate it from a possibly defect-laden approximation of $S$ ($\Sigma$, *e.g.*, a point cloud or a polygon soup) using either simple offsets in the noise-free case, or sublevel sets of a robust distance function (e.g. [Chazal et al. 2011]). Hence, under relatively mild conditions, our algorithm is able to solve the problem of robust reconstruction, repair and simplification concurrently.

## 2 Base Algorithm

### 2.1 Overview

Figure 1 depicts the three main steps of our approach: First, the initialization step generates a dense point sample $\mathcal{S}$ on the boundary of the tolerance volume $\partial\Omega$. Second, we proceed *coarse-to-fine* through refinement of a 3D Delaunay triangulation by inserting one sample of $\mathcal{S}$ at a time, and while maintaining a piecewise-linear function interpolated on the triangulation. The function value at the triangulation vertices is set in accordance to the index of each boundary component $\partial\Omega_i$ ($+1$ or $-1$). The term zero-set refers to the isosurface where the interpolated function evaluates to zero. Refinement is performed until the zero-set is entirely contained into $\Omega$ and matches the topology of $\Omega$. All samples are then well classified, and the tolerance volume is approximated by $\Gamma$, referred to as the simplicial tolerance volume. Third, we proceed mainly *fine-to-coarse* through simplifying $\Gamma$, inserting the zero-set into $\Gamma$ via mutual tessellation, and simplifying the zero-set while preserving the validity of the embedding.

### 2.2 Initialization

For initialization, we generate a $\sigma$-dense set $\mathcal{S}$ sampled on the tolerance boundary $\partial\Omega$, $\sigma$ being typically set to a fixed fraction of the minimum separation $\delta$ between the $\partial\Omega_i$. That is, the balls of radius $\sigma$ centered on $\mathcal{S}$ cover $\partial\Omega$.

For the base algorithm we assume that $\partial\Omega$ has only two components $\partial\Omega_1$ and $\partial\Omega_2$. We assign to each sample $s$ of $\mathcal{S}$ a function value: $\mathcal{F}(s) = +1$ if $s \in \partial\Omega_1$, and $\mathcal{F}(s) = -1$ if $s \in \partial\Omega_2$.

We then construct an initial 3D Delaunay triangulation ($\mathcal{T}$) with the eight corners of a loose bounding box of $\mathcal{S}$. We assign to these eight vertices the same function value as that of the samples of the outer boundary of $\Omega$. We maintain a piecewise-linear function $f$ interpolated on $\mathcal{T}$, and its zero-set, denoted by $\mathcal{Z}$.

At each sample point $s \in \mathcal{S}$ we define an error $\epsilon(s)$:

$$\epsilon(s) = |\mathcal{F}(s) - f(s)|, \tag{1}$$

where $f(s)$ denotes the interpolated function at $s$ calculated using the function value $\mathcal{F}$ of the vertices of the tetrahedron containing $s$. Each sample point $s \in \mathcal{S}$ is classified as bad if $\epsilon(s) \geq 1$, and as good (or well classified) otherwise.

During refinement of $\mathcal{T}$ with a subset of $\mathcal{S}$ (described next), the classification of $\mathcal{S}$ provides us with a means to detect when $\mathcal{Z}$ lies within $\Omega$, with a safety margin (defined in Section 3). Figure 2 illustrates in 2D a refinement on $\mathcal{T}$, the corresponding zero-set $\mathcal{Z}$ and the classification of $\mathcal{S}$.

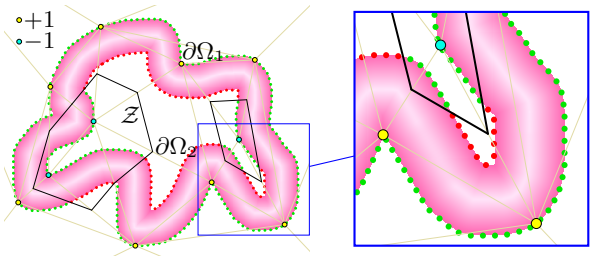



**Figure 2:** *Classification of $\mathcal{S}$. The black solid edges depict the zero-set $\mathcal{Z}$ of $f$. A sample classified as good is depicted in green, and in red otherwise.*

### 2.3 Refinement

We now refine the triangulation $\mathcal{T}$ through inserting Steiner points selected from $\mathcal{S}$, until the correct topology is met. More specifically, we insert one sample point at a time into $\mathcal{T}$ and update the Delaunay property, until $\mathcal{Z}$ classifies all samples of $\mathcal{S}$ as good, or equivalently, until $\mathcal{Z}$ separates the boundaries $\partial\Omega_i$ of $\Omega$.

Greedily inserting the sample $s$ with maximum error at each step is a natural idea for achieving the above goal with few samples. For each tetrahedron we maintain a list of sample points ($\subset \mathcal{S}$) contained in it, and a global modifiable priority queue during refinement with the maximum error points of these tetrahedra. Figure 3 illustrates several steps of a refinement sequence in 2D, until complete classification of $\mathcal{S}$.

Unfortunately, the above basic refinement algorithm is not sufficient for at least two reasons.

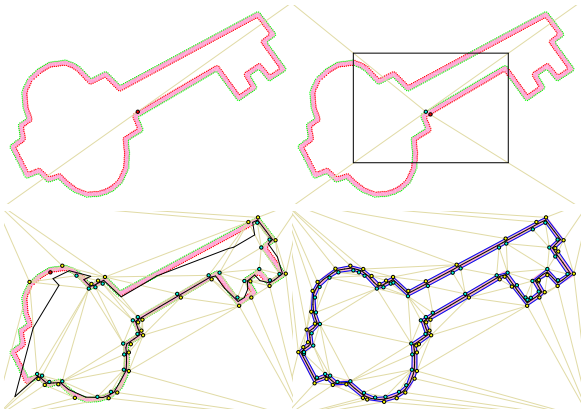


**Figure 3:** *Refinement of $\mathcal{T}$. Top: initial triangulation and one Steiner point inserted. Bottom: more Steiner points inserted, and complete classification of samples. The zero-set is depicted with black solid edges. Samples classified as good are depicted in green, and in red otherwise. A Steiner point to be inserted at the next iteration is depicted in red. Upon termination the edges of $\partial\Gamma$ are depicted in blue.*

The first reason relates to the fact that we are dealing with a finite sample of $\partial\Omega$. Even if all sample points end up being well classified, this still leaves the possibility that $\mathcal{Z}$ crosses $\partial\Omega$ in-between the samples. To prevent this from happening, we enforce that all samples are well classified with an $\alpha$ margin, as well as an upper bound on the Lipschitz constant of the piecewise-linear function.

The second reason relates to the quality of normals. In certain configurations (*e*.g., Figure 4), the normal directions are grossly wrong even in locally smooth areas. To alleviate this issue we detect so-called misoriented tetrahedra by checking that the piecewise linear function they define is locally well adapted to the geometry of $\Omega$ (condition 3 below). We note that this condition is not required for the topological correctness of the algorithm.

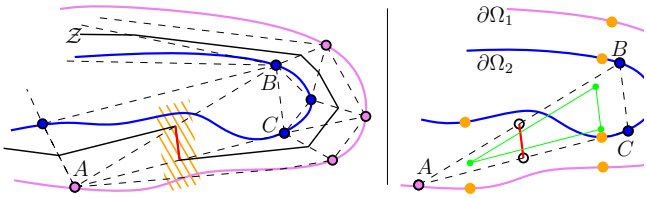



**Figure 4:** *Misoriented element. Left: The edges of $\mathcal{Z}$ are depicted with solid black lines. The zero-set of $\triangle ABC$ (red) has an incorrect normal. Right: The piecewise-linear function defined on $\triangle ABC$ should classify well the samples of $\mathcal{S}$ (on both $\partial\Omega_i$ forming $\triangle ABC$) which are nearest (orange) to the vertices of a shrunk triangle (green).*

Our modified refinement algorithm iteratively refines the triangulation until all the following criteria are met in order:

1. For some given $0 < \alpha < 1 : \forall s \in \mathcal{S}, \epsilon(s) \leq 1 - \alpha$ ($\alpha$ is set to 0.2 in all experiments).
2. The *height* of every tetrahedron contributing to $\mathcal{Z}$ is at least $2\sigma/\alpha$. The height is defined as the the distance between the supporting lines or planes of the maximal faces with different labels (Figure 5).
3. The piecewise-linear function defined by each tetrahedron $t$ classifies well the samples of $\mathcal{S}$ on both $\partial\Omega_i$ that are nearest to the vertices of a shrunk copy of $t$. The size of this shrunk copy is set to 70% of the size of $t$ in all experiments.

The term "in order" herein means that at each iteration, we look at the first condition that is violated and attempt to satisfy it by inserting a Steiner point as described below. If the condition is not satisfied after exhausting all candidate Steiner points, we move to the next condition.

Since we are dealing with a $\sigma$-dense sample, for an $\alpha$ margin (condition 1), an upper bound of $\alpha/\sigma$ on the Lipschitz constant of the piecewise-linear function suffices to ensure that the zero-set does not cross $\partial\Omega$. Noticing that the Lipschitz constant is nothing but twice the inverse height of a tetrahedron, we get an easy-to-check criterion (condition 2). The first criterion is met by adding the sample point with maximum error while the two other criteria are met by adding the sample point nearest to the circumcenter of a bad tetrahedron.

The full refinement algorithm incorporates one more condition (condition 4): while the output $\mathcal{Z}$ of the above algorithm does not have the expected genus, we refine the heterogeneous tetrahedron with the largest circumradius by adding the sample point closest to its circumcenter. This additional layer is needed to get topological guarantees on the result. However, in practice, we did not encounter a single case where the genus was not correct after the

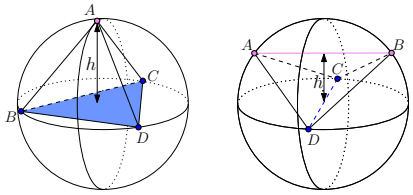



**Figure 5:** *Height of a tetrahedron contributing to $\mathcal{Z}$. The height is defined as the distance between the supporting primitive of the maximum dimension simplices formed by the tetrahedron vertices with common labels. Left: distance between point $A$ and supporting plane of $\triangle BCD$. Right: distance between supporting lines of edges $AB$ and $CD$.*

first iteration. Note also that the circumradius criterion for refining tetrahedra is blind in the sense that it does not necessarily refines the mesh where topological defects are present. While it is possible to improve the criterion from this point of view, we did not pursue this goal since it has no practical relevance.

Upon termination of the refinement step, the union of all tetrahedra of $\mathcal{T}$ which contribute to $\mathcal{Z}$, bound a *simplicial tolerance* volume $(\Gamma)$, seen as an approximation of $\Omega$. The boundary facets of $\Gamma$ are denoted by $\partial\Gamma$.

### 2.4 Simplification

The zero-set $\mathcal{Z}$ is now topologically correct. In the simplification step we reduce its complexity via the decimation of $\mathcal{T}$ combined with a mutual tessellation with $\mathcal{Z}$. Note that we stop enforcing that $\mathcal{T}$ is a Delaunay triangulation, which allows for increasingly anisotropic triangulations.

Simplification is achieved through performing a series of edge-collapse operators on $\mathcal{T}$. These operators are made conservative to preserve a valid triangulation $\mathcal{T}$, the classification of $\mathcal{S}$ and the normals achieved in previous step.

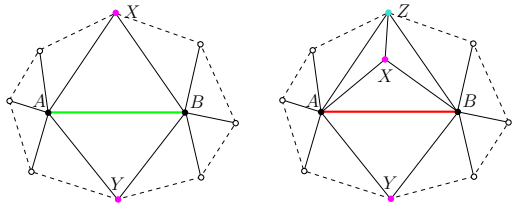



**Figure 6:** *Link condition in 2D. Left: the edge $AB$ is collapsible as $Lk(A) \cap Lk(B) = Lk(AB)$. Right: the edge $AB$ is not collapsible as $Lk(A) \cap Lk(B) \neq Lk(AB)$.*

The validity of $\mathcal{T}$ requires checking for two conditions. The combinatorial topology of $\mathcal{T}$ is preserved via the *link condition* [Dey et al. 1998] (Figure 6).

The valid embedding of $\mathcal{T}$ is preserved by computing the visibility kernel $(\mathcal{K}_\mathcal{T})(PQ)$ of a polyhedron formed by the one-ring of the edge $PQ$ (Figure 7). If the visibility kernel is non empty then locating the target vertex into this kernel preserves a valid embedding.

Preserving the classification of $\mathcal{S}$ requires further restricting the visibility kernel of an edge. As this problem is non-convex we resort to a point sampling of the kernel during the simulation of each edge-collapse operator. To obtain faithful normals locally in a smooth area, we use the same method as before (Figure 4) and check in advance whether the final solution is locally well adapted to the geometry of $\Omega$ or not.

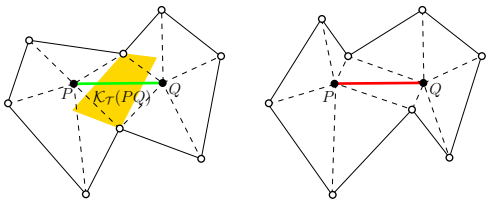



**Figure 7:** *Visibility kernel condition in 2D. Left: the edge $PQ$ is collapsible and a valid embedding is preserved when the target vertex is located within the kernel $\mathcal{K}_\mathcal{T}(PQ)$ (orange) of the polygon formed by the one-ring of the edge. Right: the kernel is empty and hence the edge $PQ$ is not collapsible.*

In order to improve efficiency we always perform simpler halfedge collapse before general edge collapse operators. A halfedge collapse operator locates the target vertex at one of the vertices of the edge. In addition, we adopt a multi-staged decimation approach with the following steps:

1. Collapse edges of $\partial\Gamma$.
2. Mutual Tessellation of $\mathcal{Z}$ into $\mathcal{T}$.
3. Collapse edges of $\mathcal{Z}$.
4. Collapse edges between $\Gamma$ and $\mathcal{Z}$, which may induce further edge collapses of $\mathcal{Z}$ (previous step).

Intuitively, we perform the steps in increasing order of computational complexity: first the operations with low number and discrete degrees of freedom, then with higher or continuous degrees of freedom. As for other decimation algorithms we need to define an error to sort the operators and to optimize the target vertex placement when performing a general edge collapse operator. In order to preserve fidelity to the initial zero-set we use as error the sum of square distances between the target vertex and the set of supporting planes of the zero-set facets located in the $2-$ring of the collapsed edge. The edge collapse operators are sorted via a priority queue sorted by increasing error.

#### 2.4.1 Simplicial Tolerance

In this step we collapse only a subset of the edges of the simplicial tolerance boundary $\partial\Gamma$. Denote by $PQ$ such an edge (Figure 8). We select as target vertex a sample point (1) from $\mathcal{S}$ (2) located within the visibility kernel $\mathcal{K}_\mathcal{T}(PQ)$ of $PQ$, (3) inducing a zero-set that preserves the classification of $\mathcal{S}$ along with normals and (4) that minimizes the aforementioned error.

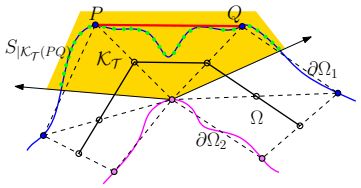



**Figure 8:** *Edge $PQ$ of $\partial\Gamma$. $PQ$ is candidate to be collapsed. The edges of $\mathcal{T}$ are depicted with dashed black lines. Visibility Kernel $\mathcal{K}_\mathcal{T}(PQ)$ is depicted (partially) in orange. The edges of $\mathcal{Z}$ (black solid lines) are not part of $\mathcal{T}$. The green dots ($S_{|\mathcal{K}_\mathcal{T}(PQ)}$) depict the subset of samples from $\mathcal{S}$ located within $\mathcal{K}_\mathcal{T}(PQ)$.*

Denote by $S_{|\mathcal{K}_\mathcal{T}(PQ)}$ the initial set of candidate sample points from $\mathcal{S}$ located within the visibility kernel $\mathcal{K}_\mathcal{T}(PQ)$. To avoid exhaustive search, we discard the sample points leading to errors in the classification of $\mathcal{S}$, as located in invalid regions, denoted $\Psi$. Figure 9 illustrates $\Psi = a \cap b \cap m$ where the point with maximum error is chosen only over $\partial\Omega_1$. A similar invalid area is computed

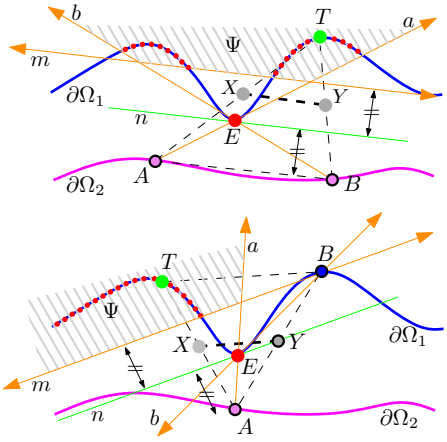



**Figure 9:** *Invalid region. Assume an edge of $\partial\Gamma$ is collapsed into the target point $T$. Line segment $XY$ denotes the zero-set of $\triangle ABT$ after collapse. Line $n$ represents the extreme zero-set of $\triangle ABT$ which preserves the classification of the point with maximum error $E$. Line $m$ delineates the corresponding locus for $T$. The intersection of the two half-spaces delineated by $a$ and $b$ represents the locus of $T$ which keeps $E$ within $\triangle ABT$. If $T$ is located in the invalid area $\Psi$ (gray) then the classification of $E$ is not preserved. Top: case where $A$ and $B$ belong to the same $\partial\Omega_i$. Line $n$ is parallel to $AB$ and passes through $E$. Bottom: case where $A$ and $B$ belong to two different $\partial\Omega_i$. Notice that $Y$ is fixed and $n$ is the supporting line of $EY$.*

by considering the point of maximum error on $\partial\Omega_2$ within $\triangle ABT$. We then collapse iteratively all edges of $\partial\Gamma$ to the point which exhibits the minimum error, as discussed above.

#### 2.4.2 Mutual Tessellation

When no more edges of $\partial\Gamma$ are collapsible, we perform a mutual tessellation between $\mathcal{Z}$ and $\mathcal{T}$ by inserting all vertices and faces of $\mathcal{Z}$ into $\mathcal{T}$. The newly inserted vertices are assigned the function value $\mathcal{F} = 0$. We then label all tetrahedra of $\mathcal{T}$ in accordance to their associated tolerance boundary component $\partial\Omega_i$. This provides us with a means to preserve the classification in the next simplification steps. A sample $s \in \partial\Omega_i$ is constrained to lie within a tetrahedron with label $i$. Intuitively, this step implements a transition from a function embedded in a volume mesh of the tolerance, to a surface mesh embedded within the 3D triangulation. Figure 10 illustrates such mutual tessellation in 2D.

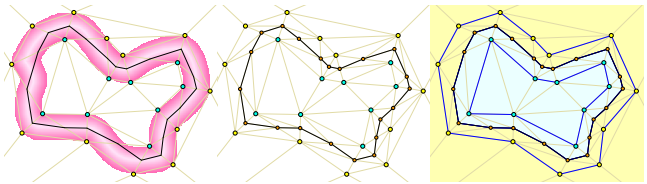


**Figure 10:** *Mutual tessellation. Left: before mutual tessellation. Middle: after mutual tessellation. Right: classification of tetrahedra in accordance to $\partial\Omega_i$. The edges of $\mathcal{Z}$ and $\partial\Gamma$ are depicted with solid black and blue lines, respectively.*

#### 2.4.3 Zero-set

After mutual tessellation we collapse the edges of $\mathcal{Z}$. Figure 11 illustrates the visibility kernel of an edge that preserves a valid embedding upon a collapse operator.

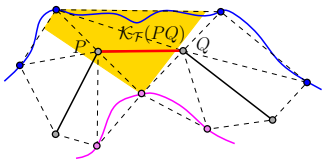



**Figure 11:** *Kernel of an edge $PQ$ of $\mathcal{Z}$. $PQ$ is candidate to be collapsed. The edges of $\mathcal{Z}$ are depicted with solid black lines. The visibility kernel $\mathcal{K}_{\mathcal{T}}(PQ)$ of $PQ$ is depicted in orange.*

Two important differences with the previous step are that we collapse an edge to an arbitrary target vertex location within the valid area ($\subset \Omega$, that minimizes the aforementioned error), and the target vertex is assigned the function value $\mathcal{F} = 0$.

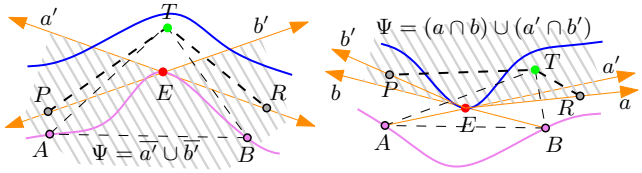
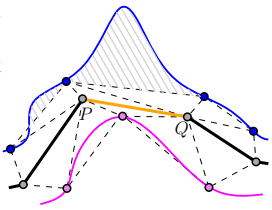



**Figure 12:** *Invalid region. Assume an edge of $\mathcal{Z}$ is collapsed into the target point $T$ ($PTR$ represents the zero-set after collapse). The intersection of the two half-spaces delineated by $a$ and $b$ represents the locus of $T$ which keeps $E$ within $\triangle ABT$. Lines $a'$ and $b'$ represent the extreme zero-set originating from $E$ that preserves the classification of point $E$. If $T \in \Psi$ (gray), the classification of $E$ is not preserved.*

To accelerate the computations, we compute invalid regions as described above. Figure 12 illustrates the invalid region by considering the point of maximum error on both $\partial\Omega_i$ for a $\triangle ABT$ when it contains one zero-set vertex. The invalid regions $\Psi$ are constructed similarly when $\triangle ABT$ contains several zero-set vertices. To further reduce the computational time when simulating general edge collapse operators, we use an octree for hierarchical sampling of $\mathcal{K}_{\mathcal{T}}$ to find the best target location and ignore further sampling of $\mathcal{K}_{\mathcal{T}}$ for the octree cells lying inside $\Psi$.

#### 2.4.4 All Edges




Due to the simplicial tolerance $\Gamma$, there may exist regions in $\Omega$ which are inaccessible (see shaded region in the inset figure). To make full use of the tolerance volume, we collapse edges between vertices of $\Gamma$ and $\mathcal{Z}$ (Figure 13). It not only helps relocating the zero-set vertices to a better location with respect to the error chosen for ordering the priority queue, but also increases the size of visibility kernel and hence, helps exploring further possibilities of an edge collapse over $\mathcal{Z}$ as discussed in 2.4.3.

Figure 14 illustrates all steps of our algorithm on a mechanical part. The input is a raw triangle soup (20k triangles). The tolerance volume is computed as the sub-level [0-0.6] of the Euclidean distance function to the input triangle soup. Note that until mutual tessellation the zero-set is made up of triangles and quadrangles before being converted into a pure triangle mesh after mutual tessellation.

## 3 Guarantees

We first derive geometric conditions under which the first three conditions of the refinement algorithm are met upon termination. Denote by $\varepsilon$ the radius of the largest ball that can fit within $\Omega$, and by

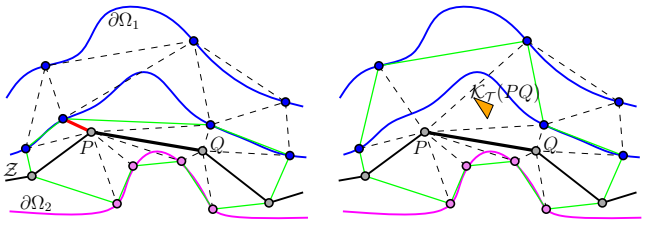


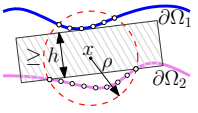

**Figure 13:** *Making an edge collapsible. Left: Edge $PQ$ is not collapsible as visibility kernel $\mathcal{K}_\mathcal{T}(PQ)$ is empty. Right: Kernel $\mathcal{K}_\mathcal{T}(PQ)$ (orange) is not empty after collapsing the red edge shown left. Collapsing an edge between a vertex of $\Gamma$ and a vertex of $\mathcal{Z}$ tends to increase the area of the one-ring of $PQ$ (green) and hence increases the probability that an edge of $\mathcal{Z}$ is collapsible.*

$\delta$ the minimum separation between the two boundary components of $\Omega$.

Condition 1 (2.3-1) is necessarily satisfied at the end of the refinement process since any sample not meeting the condition will be added to the triangulation.

Assume now that Condition 2 (2.3-2) is not satisfied upon termination. This means that there exists a heterogeneous tetrahedron $t$ with height lower than $2\sigma/\alpha$. Denote by $B$ its circumscribed ball and $r$ its radius. Ball $B$ cannot contain any sample point from $\mathcal{S}$, else that sample would have been added by the algorithm. Since $\mathcal{S}$ is a $\sigma$-sample of $\partial\Omega$, we get that the shrunk ball $B^{-\sigma}$ does not intersect $\partial\Omega$. Hence it is either empty, within $\Omega$, or outside $\Omega$. In the first case, $r \leq \sigma$. In the second case, we have that $r - \sigma \leq \varepsilon$. In the third case, because $t$ is heterogeneous, $B$ meets both boundary components of $\Omega$, hence $\sigma \geq \delta$. As a partial conclusion, upon termination, and assuming $\sigma < \delta$, the circumradius $r$ of a tetrahedron violating condition 2 cannot exceed $\varepsilon + \sigma$.

We now formulate our condition. Given two subsets $A$ and $B$ of $\mathbb{R}^3$, define the *margin* of $(A, B)$ to be the maximum thickness of a slab separating $A$ and $B$. If no such slab exists then the margin is set to zero. We say that a tolerance volume $\Omega$ is $(\rho, h)$-separated if for all $x \in \mathbb{R}^3$, the margin of $(\partial\Omega_1 \cap B(x,\rho), \partial\Omega_2 \cap B(x,\rho))$ is at least $h$.




From the above discussion, if $\sigma < \delta$, and if $\Omega$ is $(\varepsilon + \sigma, 2\sigma/\alpha)$-separated, condition 2 will be satisfied at the end of the algorithm. Similarly, condition 3 (2.3-3) will ultimately hold assuming a stronger local separation assumption on $\Omega$. However, since this condition is not essential for the topological correctness of our algorithm, we do not elaborate further on expliciting the required separation constants.

Finally, concerning condition 4, we note for future reference that if the correct genus is not met upon termination, we can bound the circumradius $r$ of any heterogeneous element as above. That is, assuming $\sigma < \delta$, we have that $r \leq \varepsilon + \sigma$.

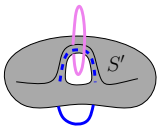
**Theorem 3.1.** *Let $\kappa$ be such that for any two points $x$ and $y$ in $\partial\Omega_i$ at distance at most $2(\varepsilon + \sigma)$, the geodesic distance between $x$ and $y$ is at most $\kappa$ times their Euclidean distance. Assuming $\Omega$ is a $((5+\kappa)(\varepsilon+\sigma)/2, 2\sigma/\alpha)$-separated topological thickening of a surface $S$, the output of the algorithm is isotopic to $S$.*

*Proof.* Our proof for isotopy utilizes the same criterion as [Chazal and Cohen-Steiner 2004]. This result states that two connected 3-dimensional surfaces $S'$ and $S$ are isotopic if:

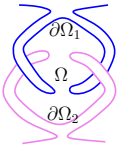
1. $S'$ is included in a topological thickening of $S$ and separates its boundary components.
2. $S'$ is connected and its genus does not exceed the genus of $S$.

In our setting, we take $\Omega$ to be a topological thickening of $S$. Being the zero-set of a piecewise-linear function, $\mathcal{Z}$ is a 2-manifold. Because condition 2 of the base algorithm is satisfied, all points in $\partial\Omega$ are well classified, meaning that $\mathcal{Z}$ is contained inside $\Omega$ and separates its boundary components.

It remains to show that the second condition holds when the refinement algorithm terminates. First, because $S'$ is a zero-set of a piecewise linear function, any component of $S'$ must enclose a vertex of the Delaunay triangulation. As there are no vertices in the interior of $\Omega$, these components must induce non-zero homology classes in $\Omega$. These components are included in the simplicial tolerance, which is fibered by line segments where the piecewise linear function is monotone. This implies that $S'$ is connected. Assume now that the genus of $S'$ exceeds the one of $S$. This means that $S'$ contains a spurious handle. Because sub and superlevel sets of piecewise linear functions are homotopy equivalent to subcomplexes of the background triangulation, we get that the two components of $\Omega \setminus S'$ contain linked homogeneous polygonal cycles in the Delaunay triangulation. For each edge in these polygonal cycles, we may form an elementary cycle by stitching the edge with a geodesic shortest path drawn on the appropriate boundary component of $\Omega$ and joining the two endpoints of the edge. Because the two polygonal cycles are linked, there must be two linked elementary cycles with different labels. Assume the correct genus is not met upon termination of the algorithm. Then we may assume that the Delaunay edges in the linked polygonal cycles are edges of heterogeneous tetrahedra. Hence their length is at most $2(\varepsilon+\sigma)$. Hence the length of the elementary cycles are at most $2(1+\kappa)(\varepsilon+\sigma)$. Because two of them are linked, the tolerance volume $\Omega$ cannot be $((5+\kappa)(\varepsilon+\sigma)/2, 0)$-separated (Appendix A.1). □




The above conditions are clearly met when the tolerance volume is a sufficiently small offset of a smooth surface, for example. Also, the algorithm can also be shown to work in situations not covered by the above theorem, as for instance tolerances bounded by convex surfaces. The numerical constants in the theorem may be further optimized, and other types of conditions, *e.g.* based on the separation $\delta$ can also be proved to be sufficient. In particular, if $2(\varepsilon+\sigma)\sqrt{\kappa^2-1} < \delta$, the algorithm is correct. The inset figure depicts an example of tolerance volume where the algorithm would fail. It is apparent from the proof above that such configurations are essentially the only way the algorithm can fail. Note that even in such situations, the output of the algorithm will be a manifold surface, albeit possibly with a too large genus. Finally, we note that if the only pursued goal was to provide topologically guaranteed output for tolerances that are topological thickenings, then a trivial solution would be to output one of the tolerance boundary surface. However, such methods would be limited to topological thickenings, while our approach may work in less favorable situations. In addition, algorithms inspired by the trivial idea above do not seem to allow successful subsequent simplifications, even in favorable cases.




## 4 Extensions

The algorithm above can be extended in order to deal with boundaries and non-manifold surfaces. While the guarantees can be extended to non-closed surfaces, for non-manifold cases, they are more difficult to formulate and are probably beyond the reach of existing tools. Still, the output is guaranteed to have the correct homotopy type in these cases also.

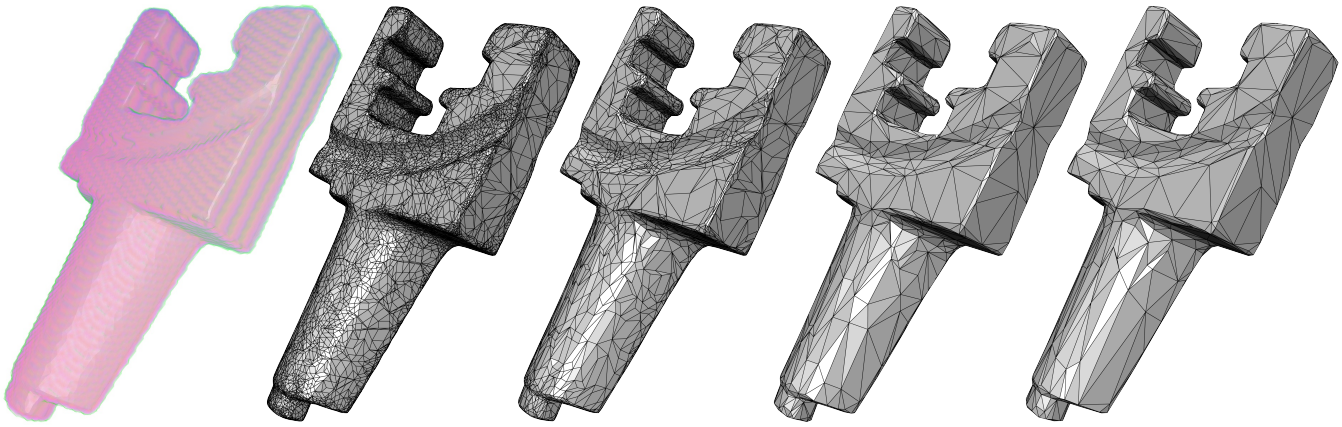


**Figure 14:** *Blade. From left to right: Input tolerance* ($\delta = 0.6\%$)*; $\mathcal{Z}$ after refinement (*$20.4k$ *vertices); simplification of* $\partial\Gamma$ ($5.3kv$)*; mutual tessellation and simplification of* $\mathcal{Z}$ ($1.01kv$)*; and the final output* ($752v$).

### 4.1 Non-closed Surfaces

Our algorithm deals with surfaces with boundaries when $\Sigma$ is provided as input along with $\Omega$. We first detect sample points corresponding to boundaries, referred to as $\mathcal{B}$, using $\gamma\delta'-$radius balls centered at $\Sigma$ (Figure 15), where $\gamma$ denotes a user defined parameter derived from the reach of input data. Parameter $\delta'$ is defined as the minimum distance between the current center of ball and $\mathcal{S}$.

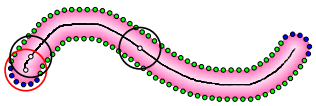


**Figure 15:** *Boundary sample points. The intersection between $\mathcal{S}$ and ball located on the input surface $\Sigma$ is composed of a single connected component near a boundary.*

When a ball contains a single connected component (samples connected to each other by paths with a maximum step distance of $2\sigma$) boundary surface of the tolerance volume then the associated samples are considered part of $\mathcal{B}$. Conversely, the sample points which correspond to a multi-component surface - with a minimum distance between the components greater or equal to $2\delta'$ - are not considered part of $\mathcal{B}$. Note that when $\Sigma$ is not provided, balls centered at $\mathcal{S}$ can also be used, but this severely limits the reach size of the input data that can be dealt with. Once the boundary is detected, we use the set $\mathcal{S} \setminus \mathcal{B}$ as the set of sample points in the initialization stage. In other words, we ignore the classification of $\mathcal{B}$ with respect to $f$. Via refinement as described in Section 2.3, we then classify all sample points of $\mathcal{S} \setminus \mathcal{B}$ and clip the zero-set by $\Omega$. We enforce during the simplification step that the two-sided Hausdorff distance between boundary of $\mathcal{Z}$ and $\mathcal{B}$ is at most $\delta$. Furthermore, in order to preserve smoothness along the boundary, we use in this last step besides Hausdorff distance an extra error term defined as the sum of squared distances between the target vertex and the set of supporting boundary edges of the zero-set located in the $2-$ring of the edge to be collapsed.

Figure 16 depicts a range scan of the *Kitten* point cloud with boundaries due to missing data. The holes are preserved by the non-closed variant of our algorithm (middle). Note that by not ignoring the parts of the zero-set outside $\Omega$, we can also fill the large hole on the nearly flat area. Nevertheless, more work is needed to reliably deal with more complicated holes.

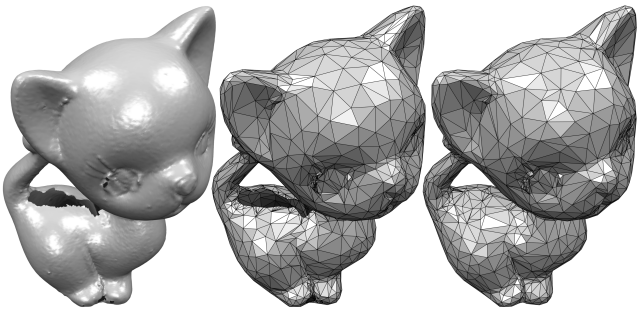


**Figure 16:** *Preserving or repairing holes. Left: Range scan of a kitten. Middle: Our output as non-closed surface (*$1.3k$ *vertices). Right: Our output with holes filled (*$1.2k$ *vertices).*

### 4.2 Non-manifold Surfaces

To handle non-manifold surfaces when computing the error of a sample, we evaluate $f$ with respect to each component of $\partial\Omega$. More specifically, to evaluate the error $\epsilon$ at a sample $s \in \partial\Omega_i$ we define

$$\forall p \in \mathcal{S}, \mathcal{F}(p) = \begin{cases} +1, & p \in \partial\Omega_i \\ -1, & p \notin \partial\Omega_i. \end{cases} \quad (2)$$

The zero-set is ignored when its end points lie outside $\Omega$; this configuration occurs when the input geometry is made up of several components. The process described above yields a zero-set (Figure 17, top left) with topological artifacts where several surfaces of $\partial\Omega$ meet. These artifacts are located inside tetrahedra containing vertices from three or more $\partial\Omega_i$. In addition, such tetrahedron may contain several zero-sets corresponding to the total number of possible permutations when assigning function values to its vertices. We remove these artifacts by joining all zero-set edges to the centroid of the zero-set vertices located on the edges of this tetrahedron. Figure 17 illustrates our algorithm at work on a non-manifold geometry.

## 5 Experiments

**Implementation.** Our algorithm is implemented in C++ using the CGAL library for the triangulation data structures and the Intel Threading Building Blocks library for parallelization. 3D tolerance volumes are rendered via 3D texture mapping using pixel shaders from the NVIDIA Cg Toolkit. All atomic operations performed

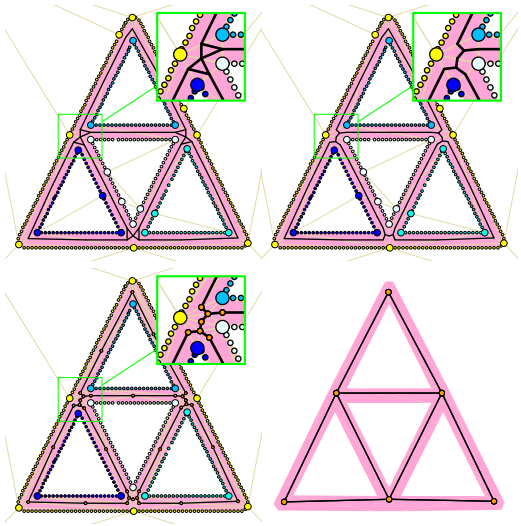


**Figure 17:** *Dealing with a non-manifold geometry. Top: refinement until matching the topology. Bottom: mutual tessellation and final output.*

over the tetrahedra or sample points are easily parallelized as they are independent. All experiments are performed on an Intel 2.4GHz 16-core machine with 128 Gb RAM. The tolerance errors are specified as a percentage of the longest edge of the bounding box of the input data. Margin $\alpha$ is set to $0.2$ in all experiments.

Though for simplicity of exposition, we assigned $\mathcal{F}$ at bounding box vertices as that of the outer boundary. In practice, we found that by multiplying this assignment by the distance to $\Omega$ significantly reduces over refinement. However this choice might lead to large interpolated values at samples of the outer tolerance boundary. Since this does not hinder classification, we do not further refine in such cases. One way to implement this idea, is to replace the error $\epsilon(s)$ for classification by $\epsilon(s) = 1 - f(s)/\mathcal{F}(s)$.

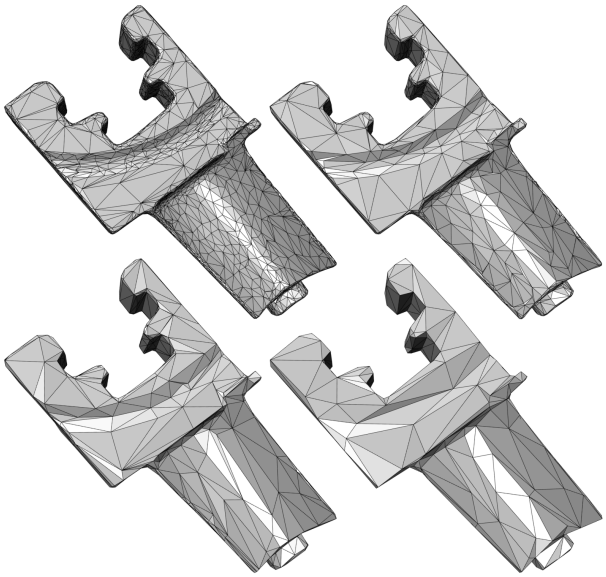


**Figure 18:** *Blade. From top left to bottom right:* $\delta$ *is set to* $0.15$*,* $0.35$*,* $0.9$ *and* $1.5\%$*. The final vertex count are* $3,020$*,* $1,015$*,* $493$ *and* $254$*, respectively.*

Figure 18 illustrates our algorithm at work on a mechanical part (*blade*), for several separation distances between the boundaries of the tolerance volume. The overall time consumed by the algorithm ranges from 34 minutes for $\delta = 1.5\%$ to around 7 hours for $\delta = 0.15\%$. Figure 19 shows outputs of our algorithm on a smooth surface (fertility) with $\delta$ ranging from $0.15\%$ to $8\%$. The one-sided Hausdorff measured from the output to the input is bounded in all cases. Note that when $\delta$ is large (bottom right), the tolerance volume is not a topological thickening anymore as the topology of the *inner* boundary of the tolerance changes. We also run our algorithm on *Armadillo* - a more general mesh made of smooth and flat parts (Figure 20).

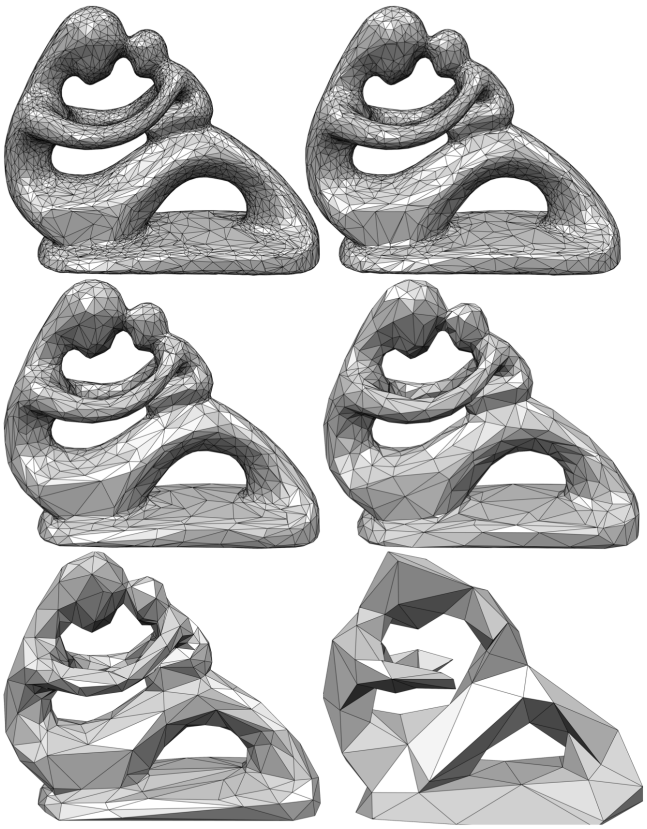


**Figure 19:** *Fertility. From top left to bottom right:* $\delta$ *is set to* $0.15$*,* $0.25$*,* $0.4$*,* $0.8$*,* $2.0$ *and* $8.0$*. The final vertex count is* $4,642$*,* $2,768$*,* $1,484$*,* $767$*,* $417$ *and* $120$*, respectively. The input model is a surface triangle mesh of the fertility model with* $14k$ *vertices.*

**Figure 20:** *Armadillo.* $\delta$ *is set to* $0.1\%$ *and* $0.9\%$*. The final vertex count is* $26,189$ *and* $1,518$*. The input model is a surface triangle mesh of the armadillo model with* $173k$ *vertices.*

**Vertex count over time.** Figure 21 plots the vertex count of $\mathcal{Z}$ against time, for the fertility model and different values for $\delta$. In all experiments the refinement step is substantially faster than the

multi-staged simplification step. The two batches of halfedge collapse operators applied to $\partial\Gamma$ and $\mathcal{Z}$ decreases the vertex count rapidly. The more general edge collapse operators are substantially slower. The time taken per operator further increases as we move from $\partial\Gamma$ to $\mathcal{Z}$, and finally to all edges. Such increase is mostly due to the transition from sampling the kernel of the edge only over $\partial\Omega$ (Figure 8) to pointwise probing of the whole kernel volumes in later stages. Another reason for the escalating time per operator is due to the progressive increasing of the kernel volumes when the mesh coarsens. In addition, each tetrahedron contains on average more samples and hence requires more time to verify the classification of these samples. In other words, discovering progressively the anisotropy in the input geometry, under the tolerance volume constraint, comes at an increasing cost for each edge collapse operator.

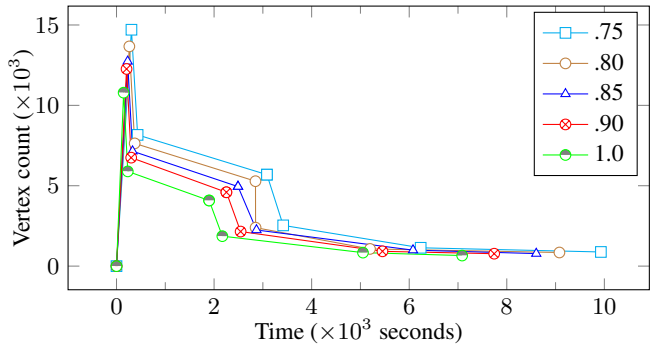



**Figure 21:** *Evolution of mesh complexity over time for different $\delta(\%)$. Each mark depicts the completion of: refinement; simplification via halfedge collapses of $\partial\Gamma$; edge collapses of $\partial\Gamma$; mutual tessellation and halfedge collapses of $\mathcal{Z}$; edge collapses of $\mathcal{Z}$; and simplification of all edges. The input is a raw surface mesh of the fertility model (14kv).*

**Comparisons.** A strict qualitative comparison with previous work is not possible as our problem statement differs. The one-sided Hausdorff distance preserved in general mesh decimation algorithms is measured from the input to the output mesh, while we guarantee the other side of the Hausdorff distance. Nevertheless, we plot our one-sided Hausdorff distance against the number of vertices of the final output mesh, for five other mesh approximation algorithms: simplification envelopes [Cohen et al. 1996], a decimation algorithm from Lindstrom-Turk [Lindstrom and Turk 1999] (without error bounds), the MMGS remeshing algorithm [Borouchaki and Frey 2005] (with Hausdorff error bound), MMGS with the mesh anisotropy option and a decimation algorithm with error bound implemented in OpenFlipper [Möbius and Kobbelt 2012].Albeit none of the other approaches except [Cohen et al. 1996] target an intersection-free output, we indicate with a dot a self-intersection of the output mesh (Figure 22). For completeness we also plot the other-sided Hausdorff distance over the same input and output datasets (Figure 23). In both cases we achieve a lower vertex count for a given tolerance error, at the price of higher computational times.

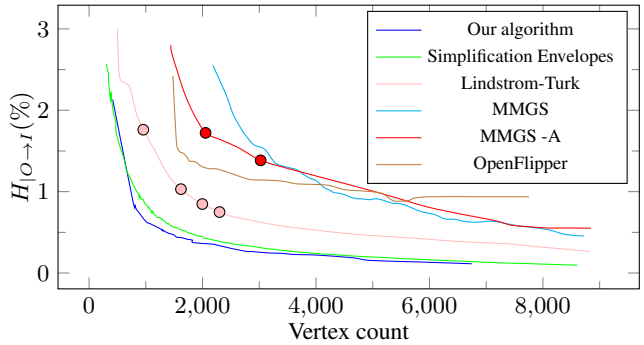



**Figure 22:** *Comparisons. We plot the one-sided Hausdorff distance (output to input) $(H_{|O\to I})$ against the final number of vertices. A dot indicates a self-intersection of the output mesh. The input mesh is a clean surface triangle mesh of the fertility model (14kv).*

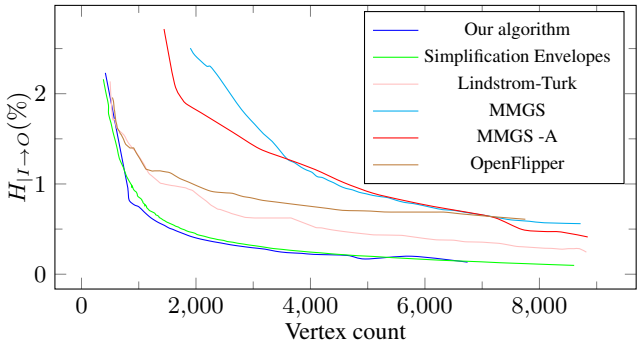



**Figure 23:** *Comparisons. We plot the other-sided Hausdorff distance (input to output) $(H_{|I\to O})$ against the final number of vertices over the same data.*

We also compare our algorithm with [Cohen et al. 1996] on *Armadillo* (Figure 24). For a very large tolerance, the vertex count for our algorithm and simplification envelopes is comparable. However, on most part of the curve, our algorithm generates on average 10% fewer vertices, for a given tolerance error. Note also that the *simplification envelopes* require a manifold mesh as input. In addition, they cannot simplify the geometry of highly undulating surfaces beyond a certain limit, due to the specific type of tolerance volume used (Figure 25).

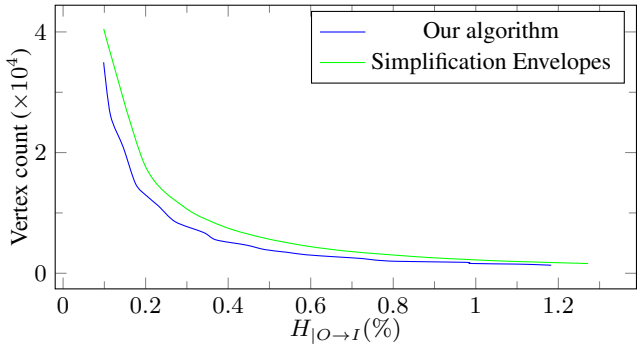



**Figure 24:** *Comparison with the Simplification envelopes [Cohen et al. 1996]. We plot the final number of vertices against the one-sided Hausdorff distance (output to input) $(H_{|O\to I})$. The input is a clean surface triangle mesh of Armadillo (173k vertices).*

**Normals.** We do not provide quantitative guarantees of faithful approximation of normals in all cases. Instead our proof (Section 3) yields good normals in smooth areas, when the sampling $\sigma$ is dense enough. Figure 26 plots the distribution of normal deviation with respect to different shrinkage factors used for condition 3 for the Fertility model. Dropping this condition may cause in practice the normal deviation to exceed $90^\circ$. Figure 27 provides on the Lucy model a visual comparison of normals of our output to the input surface triangle mesh (left).

Dealing with sharp creases subtending small angles may require an extremely dense $\sigma$, which also translates into dense refinement.

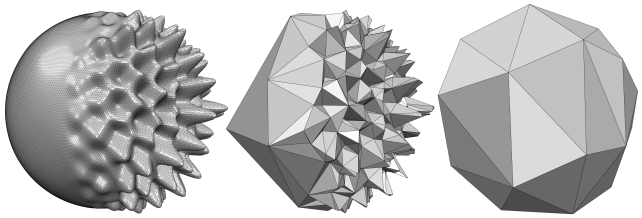


**Figure 25:** *Aggressive simplification. As the tolerance in the* Simplification Envelopes *is generated by offsetting the vertices along the normals, this approach cannot simplify the geometry of highly undulating surfaces beyond a certain Hausdorff limit. Left: input mesh. Middle: output from the* Simplification Envelopes $(H_{|O \to I} = 60\%)$. *Right: output from our algorithm* $(H_{|O \to I} = 10\%)$.

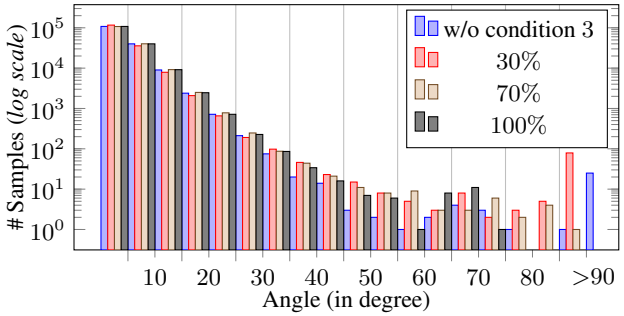



**Figure 26:** *Distribution (in log scale) of normal deviation. We plot the distribution of normal deviation of our output for the fertility model* ($\delta = 0.2\%$) *when condition* 3 *is not used, and when a* 30%, 70% *and* 100% *shrinkage factor is used. The final vertex count of the output is* $3,498$, $3,538$, $3,624$ *and* $5,715$ *respectively.*

When using a too large value for $\sigma$, activating the preservation of normals during refinement may further translate into dense refinement as the normals are ill-defined locally. We cannot always deduce whether overly dense refinement comes from the preservation of inferred normals or from the recovery of the topology. As the decimation preserves the inferred normals we may end up with overly complex meshes. However, relaxing constraint 3 during halfedge collapses can alleviate this issue.

**Robustness.** A primary virtue of our algorithm is its resilience to the type and defects of the input dataset. More specifically, and as our algorithm takes a tolerance volume as input, the robustness of our algorithm is delegated to the construction of a robust tolerance volume - then the output is guaranteed to be homotopy-equivalent to the given tolerance volume. Said differently, our algorithm is oblivious to the dataset inside the tolerance as long as the tolerance is well-behaved. Figure 28 illustrates the robustness of our algorithm on point sets and defect-laden triangle soups sampled on the elephant model, with two levels of noise. We use as tolerance volume a sub-level of the robust distance function based on distances between measures [Chazal et al. 2011].

**Limitations.** Despite its guarantees and qualitative performances, our algorithm is compute-intensive, especially when setting a small tolerance. On Figure 14 the tolerance is set to $\delta = 0.6\%$ and our algorithm runs for approximately 3h and consumes $2.1Gb$ of peak RAM memory. The time complexity is dominated by the simplification step. Table 1 lists the time taken by each step of the algorithm against the vertex count of $\mathcal{Z}$.

**Figure 27:** *Lucy. Left: Input model* $(50kv)$. *Right: Output from our algorithm* $(16.7k$ *vertices,* $\delta = 0.13\%)$.

**Table 1:** *Timing (in seconds) for each step of our algorithm for the blade model depicted in Figure 14.*

| Stage | # Vertices ($\mathcal{Z}$) | Time (s) | Time per iteration |
|---|---|---|---|
| Refinement | $20,447$ | 655 | 0.0319 |
| halfedge - $\partial\Gamma$ | $10,217$ | 326 | 0.0318 |
| general - $\partial\Gamma$ | $5,346$ | $4,658$ | 0.956 |
| halfedge - $\mathcal{Z}$ | $2,292$ | 153 | 0.050 |
| general - $\mathcal{Z}$ | $1,015$ | $1,478$ | 1.157 |
| All edges | 752 | $4,537$ | 17.185 |
| Total | | $11,807$ | 0.2941 |

Most of the time spent by the algorithm is in the exhaustive search to find the best point location for an edge collapse operator, and this time escalates as the decimation proceeds. Unlike other mesh decimation algorithms, the running time of our algorithm is decreasing with parameter $\delta$ of the specified tolerance. A small tolerance requires dense sampling and hence $|\mathcal{S}|$ increases together with the time consumed to classify the samples. Another dominating factor is the sampling density used to probe a kernel when searching for the best point of a general edge collapse operator. The halfedge collapse operators are on average two orders of magnitude faster, but are not sufficient to generate coarse meshes. On the positive side, each operation performed over all sample points and tetrahedra of the triangulation is parallelizable. Figure 29 plots the speed up in the run-time of our algorithm versus the number of CPU cores.

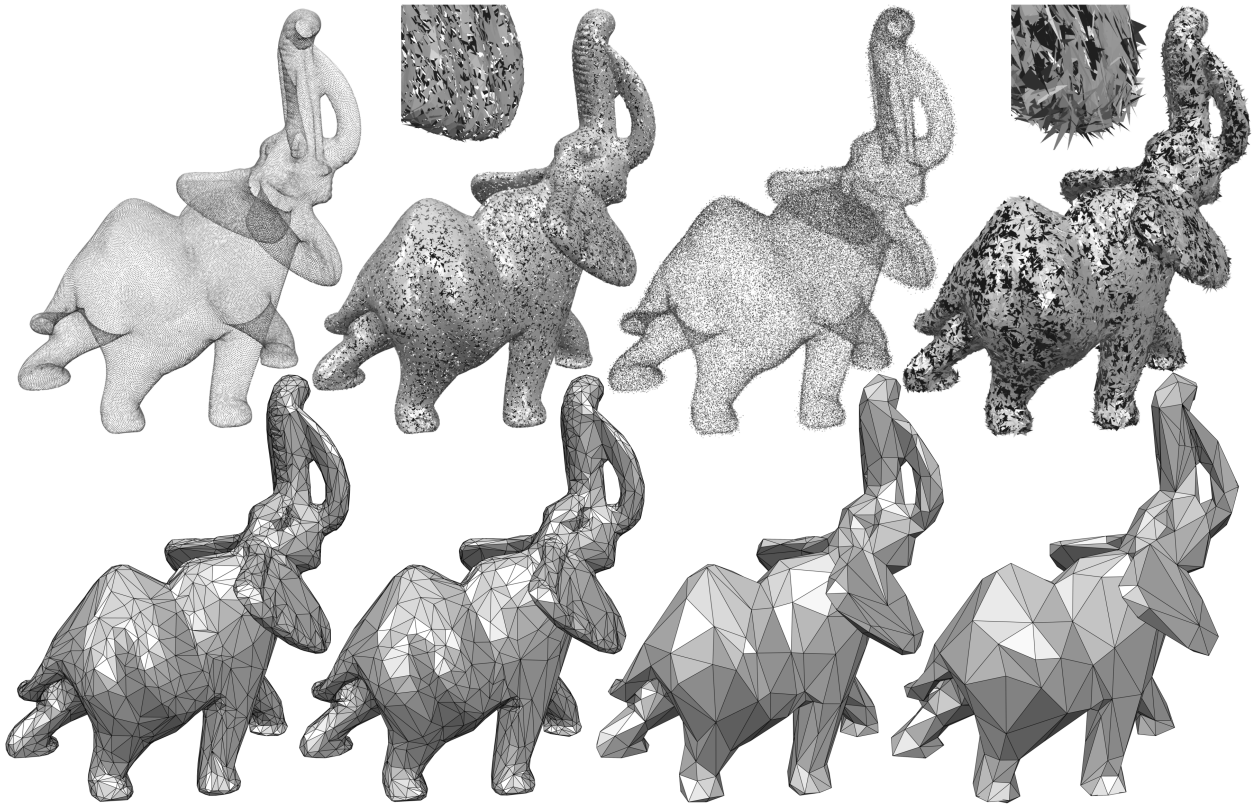


**Figure 28:** *Robustness to noise and type of input datasets. From left to right: point set, triangle soup with low noise, noisy point set, and triangle soup with high noise. The corresponding $\delta$ and output vertex count are* $0.9$*,* $1.2$*,* $2$*,* $5\%$ *and* $2,191$*,* $1,897$*,* $1,082$*,* $502$ *respectively.*

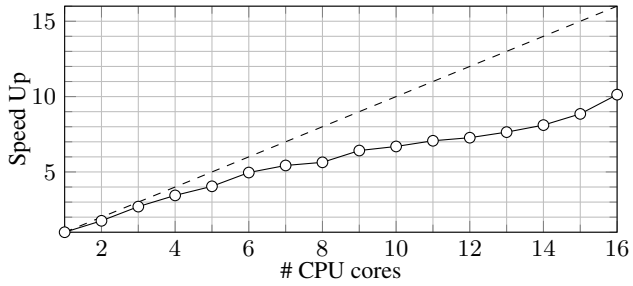



**Figure 29:** *Speed up. We plot the speed up in the run-time of our algorithm versus the number of CPU cores (input: Fertility, $\delta$ is set to* $0.4\%$*).*

## 6 Discussion

We introduced a novel approach to the problem of isotopic approximation within a tolerance volume. We depart from common approaches by leveraging on a dense point sample of the boundary of the tolerance volume. We designed a pliant meshing algorithm that first proceeds by Delaunay refinement in order to recover the correct topology through classifying the samples, and then by topology-preserving simplification in order to discover the anisotropy within the tolerance volume. A distinctive feature of our approach is its robustness to input data sets. As our approach is oblivious to the type and defects of the datasets within the tolerance volume, it can reconstruct, repair and simplify concurrently. Compared to error-driven simplification algorithms, with or without error bounds, our approach makes full use of the tolerance volume and achieves lower vertex counts for a given tolerance error, in addition to intersection-free outputs. Such lower vertex counts, however, come at a price: our current implementation is compute-intensive, and the computational times escalate when the tolerance decreases.

The current version of the algorithm is highly parallelizable: in practice we observed that the running time was inversely proportional to the number of processors. As future work we wish to extend our approach so as to make it out-of-core. A natural direction is to cut the tolerance volume into sub-parts before stitching, but it requires another line of work to preserve the guarantees during simplification. Finally, another stimulating direction is the concept of a progressive approximation algorithm, in which we could guarantee that every additional CPU cycle spent by the algorithm is making progress toward the optimal solution that matches the global minimum vertex count.

## Acknowledgements

The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007-2013) ERC Grant Agreements No.339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions) and No.257474 IRON (Robust Geometry Processing). We wish to thank Jonathan D. Cohen for providing us with the source code of the simplification envelopes and Pascal Frey for the MMGS software. We also thank the *VISIONAIR* shape repository for the datasets, Jan Möbius for the geometry processing framework *OpenFlipper*, and the reviewers for their constructive comments.

## References

AGARWAL, P. K., AND SURI, S. 1998. Surface approximation and geometric partitions. *Journal of Computing 27*, 4, 1016–1035.

AGGARWAL, A., BOOTH, H., O'ROURKE, J., SURI, S., AND YAP, C. K. 1985. Finding minimal convex nested polygons. In *Proceedings of ACM Symposium on Computational Geometry*, 296–304.

AMENTA, N., AND BERN, M. 1998. Surface reconstruction by voronoi filtering. *Discrete and Comp. Geometry 22*, 481–504.

AMENTA, N., CHOI, S., DEY, T. K., AND LEEKHA, N. 2000. A simple algorithm for homeomorphic surface reconstruction. In *Proceedings of ACM Symp. on Comp. Geometry*, 213–222.

ATTENE, M. 2010. A lightweight approach to repairing digitized polygon meshes. *The Visual Computer 26*, 1.

BISCHOFF, S., PAVIC, D., AND KOBBELT, L. 2005. Automatic restoration of polygon models. *ACM Transactions on Graphics 24*, 1332–1352.

BOISSONNAT, J.-D., AND CAZALS, F. 2000. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proceedings of ACM Symposium on Computational Geometry*, 223–232.

BOISSONNAT, J.-D., AND OUDOT, S. 2005. Provably good sampling and meshing of surfaces. *Graph. Models 67*, 5, 405–451.

BOROUCHAKI, H., AND FREY, P. 2005. Simplification of surface mesh using hausdorff envelope. *Computer Methods in Applied Mechanics and Engineering 194*, 48-49, 4864 – 4884.

BOTSCH, M., BOMMES, D., VOGEL, C., AND KOBBELT, L. 2004. GPU-based tolerance volumes for mesh processing. In *Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, 237–243.

CHAZAL, F., AND COHEN-STEINER, D. 2004. A condition for isotopic approximation. In *Proceedings of ACM Symposium on Solid Modeling and Applications*, 93–99.

CHAZAL, F., COHEN-STEINER, D., AND MÉRIGOT, Q. 2011. Geometric Inference for Measures based on Distance Functions. *Foundations of Computational Mathematics 11*, 6, 733–751.

CIAMPALINI, A., CIGNONI, P., MONTANI, C., AND SCOPIGNO, R. 1997. Multiresolution decimation based on global error. *The Visual Computer 13*, 5, 228–246.

COHEN, J., VARSHNEY, A., MANOCHA, D., TURK, G., WEBER, H., AGARWAL, P., BROOKS, F., AND WRIGHT, W. 1996. Simplification envelopes. In *Proceedings of ACM Conference on Computer Graphics and Interactive Techniques*, 119–128.

COHEN, J., MANOCHA, D., AND OLANO, M. 2003. Successive mappings: An approach to polygonal mesh simplification with guaranteed error bounds. *International Journal of Computational Geometry and Applications 13*, 1, 61–96.

DEY, T. K., AND GOSWAMI, S. 2003. Tight cocone: A watertight surface reconstructor. In *Proceedings of ACM Symposium on Solid Modeling and Applications*, 127–134.

DEY, T. K., AND SUN, J. 2005. An adaptive mls surface for reconstruction with guarantees. In *Proceedings of EUROGRAPHICS Symposium on Geometry Processing*.

DEY, T. K., EDELSBRUNNER, H., GUHA, S., AND NEKHAYEV, D. V. 1998. Topology preserving edge contraction. *Publ. Inst. Math. (Beograd) (N.S 66*, 23–45.

DEY, T. K., LI, K., RAMOS, E. A., AND WENGER, R. 2009. Isotopic reconstruction of surfaces with boundaries. *Computer Graphics Forum 28*, 5, 1371–1382.

DEY, T. K. 2006. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis (Cambridge Monographs on Applied and Comp. Mathematics)*. Cambridge University Press.

GUÉZIEC, A. 1996. Surface simplification inside a tolerance volume. Tech. Rep. 20440. IBM Research Report RC 20440.

GUMHOLD, S., BORODIN, P., AND KLEIN, R. 2003. Intersection free simplification. *International Journal of Shape Modeling 9*, 2, 155–176.

HORNUNG, A., AND KOBBELT, L. 2006. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *Proceedings of EUROGRAPHICS Symposium on Geometry Processing*, 41–50.

JU, T. 2004. Robust repair of polygonal models. *ACM Transactions on Graphics 23*, 3, 888–895.

KALVIN, A. D., AND TAYLOR, R. H. 1996. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications 16*, 3.

KAZHDAN, M., BOLITHO, M., AND HOPPE, H. 2006. Poisson surface reconstruction. In *Proceedings of EUROGRAPHICS Symposium on Geometry Processing*, 61–70.

KLEIN, R., LIEBICH, G., AND STRASSER, W. 1996. Mesh reduction with error control. In *IEEE Visualization*, 311–318.

LINDSTROM, P., AND TURK, G. 1999. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics 5*, 2, 98–115.

MÖBIUS, J., AND KOBBELT, L. 2012. Openflipper: An open source geometry processing and rendering framework. In *Proceedings of International Conference on Curves and Surfaces*, Springer-Verlag, 488–500.

OVREIU, E., RIVEROS REYES, J. G., VALETTE, S., AND PROST, R. 2012. Mesh simplification using a two-sided error minimization. In *Proceedings of International Conference on Image, Vision and Computing*, 26–30.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics 23*, 3, 896–904.

ZELINKA, S., AND GARLAND, M. 2002. Permission grids: Practical, error-bounded simplification. *ACM Transactions on Graphics 21*, 2, 207–229.

## A Interlocked loops

**Theorem A.1.** *Assume* 2 *interlocked loops, each formed by joining a segment of length $l_e$ with a continuous curve of length $l_c$. Then the continuous curves of the two loops cannot be $(\frac{l_e+l_c}{4} + l_e, 0)$-separated.*

*Proof.* Each loop is contained within a ball of radius $\frac{l_e+l_c}{4}$. Let $B$ be such a ball for the first loop $C_1$. Because the two loops are linked, the part of the second loop $C_2$ lying in $B$ cannot be linearly separated from the first loop. If this part only consists of the curve part, the conclusion follows. Else, let $B'$ be the ball obtained by enlarging $B$ by $l_e$. Now $B' \cap C_2$ must contain the two endpoints of the segment part of $C_2$. Also, $B' \cap C_2$ and $C_1$ are not linearly separable. Since the segment lies in the convex hull of the curve part of $B' \cap C_2$, the conclusion follows.