Persistence-based Handle and Tunnel Loops Computation Revisited for Speed Up

Tamal K. Dey*,[a], Kuiyu Li[a]

[a]*Department of Computer Science and Engineering, The Ohio State University*
2015 Neil Ave, Columbus, OH 43210, USA

Abstract

Loops in surfaces associated with topological features such as handles and tunnels are important entities in many applications including surface parameterization, feature identification, and topological simplification. Recently, a persistent homology based algorithm has been proposed to compute them. The algorithm has several advantages including its simplicity, combinatorial nature and independence from computing other extra structures. In this paper, we propose changes to this loop computation algorithm based on some novel observations. These changes reduce the computation time of the algorithm dramatically. In particular, our experimental results show that the suggested changes achieve considerable speed up for large data sets without sacrificing loop qualities.

Key words:
Shape analysis, topology, loops in surfaces, homology, persistent homology, topological persistence

1. Introduction

Computations of meaningful non-trivial *loops* in surfaces is a fundamental problem that crops up in various applications such as surface parameterization [2, 16, 17, 22], feature identification [1, 4, 9, 20], topological repair and simplification [5, 18, 24, 25]. As a result, considerable amount of research has been devoted in recent years to compute such loops [9, 13, 14, 15, 27].

In most applications, the loops should be linked to the topology of the surface and be small in size. To this goal, Dey, Li, Sun, and Cohen-Steiner [10] recently proposed a persistence-based algorithm to compute a special class of loops called handle and tunnel loops. This class of loops introduced in [9] captures the intuitive notion of 'handles' and 'tunnels' in a shape. To be more precise, let M denote a connected closed surface sitting in $\mathbb{R}^3$. The handle and tunnel loops in M are defined in terms of the first homology group of M and its embeddings in $\mathbb{R}^3$. One can intuitively think that a loop is a handle loop if it bounds a disk in the interior of M whereas it is a tunnel loop if it bounds a disk in the exterior of M. Figure 1 shows such loops in a 3D model *Kitten* and an iso-surface *Atom*.

Persistent homology introduced by Edelsbrunner, Letscher, and Zomorodian [12] perfectly fits the definition of handle and tunnel loops. Consequently, they could be computed with the persistence algorithm easily. Dey et al. [10] brought geometry into the loop computation by incorporating geodesic distances into the persistence algorithm. Of course, the output loops may not be optimal geometrically, but they are small as empirical results confirm. One advantage of the persistence-based algorithm is that it is mostly combinatorial in nature and thereby avoids costly, error-prone numerical computations. Furthermore, unlike many previous methods, this algorithm does not require computing any extra data structures such as Reeb graphs [4, 6, 21], medial axes [25], or curve skeletons [9].

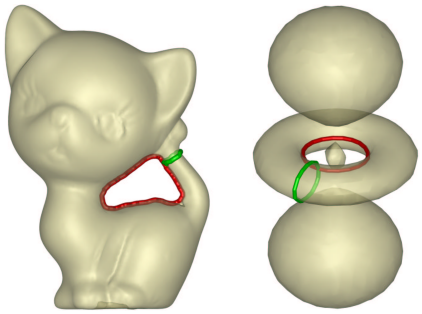


Figure 1: Handle (green) and tunnel (red) loops in 3D model *Kitten* and iso-surface *Atom*.

In the persistence algorithm, a core component is a simplex pairing algorithm which pairs simplices from an ordered sequence of simplices called *filtration* of a simplicial complex [12, 26]. This *Pairing* algorithm is used in [10] to compute handle and tunnel loops. A simplex is called either *positive* if it creates a cycle or *negative* if it destroys a cycle when it is added according to the order in the filtration. A negative p-simplex σ is always paired with a unique positive $(p-1)$-simplex σ' where σ kills a $(p-1)$-cycle created by σ'. Suppose that a surface complex $\mathbb{M}$ along with complexes that tessellate its interior and exterior volumes are given. Denote the union of the surface and volume complexes as $\mathbb{K}$. A handle or tunnel loop is obtained when a volume triangle in $\mathbb{K} \setminus \mathbb{M}$ is paired with a pos-

*Phone: 614-292-3563, Fax: 614-292-2911
Email addresses: tamaldey@cse.ohio-state.edu (Tamal K. Dey), liku@cse.ohio-state.edu (Kuiyu Li)

itive edge in $\mathbb{M}$. The authors in [10] propose two refinements to make the handle and tunnel loops geometry-aware. More details about this algorithm are given in Section 4.

Contribution: In this paper, we revisit the persistence-based method of [10] for speed up. We make some important observations (Section 5.1.1) in the persistence algorithm and several other changes (Sections 5.2 and 5.1.2) that allow us to achieve a considerable improvement in computation time. In particular, our proposed modifications make the persistence based algorithm scalable to large data without sacrificing loop qualities. The rest of the paper is organized as follows.

In Section 2, we briefly introduce homology based on which the handle and tunnel loops in closed surfaces are defined.

Section 3 describes the concepts of filtration and the *Pairing* algorithm in the theory of persistent homology [12].

Section 4 introduces the algorithm of [10] for computing handle and tunnel loops which consists of two main stages: geodesic computation for the volume simplices and loop computation using the *Pairing* algorithm [10].

In Section 5, we propose methods to improve the performance of each stage. We make a key observation based on which a modified *Pairing* algorithm and a new ordering of surface simplices in the filtration accelerate the loop computation dramatically. We also propose a method for estimating the *geodesic size* of the volume simplices for speed up.

We validate our claim of improvements with experimental results in Section 6. We finally draw conclusions in Section 7.

2. Handle and Tunnel Loops

The handle and tunnel loops in a closed surface are defined by the one dimensional homology group of the surface and its embedding in $\mathbb{R}^3$. We first review some basic facts about homology groups. For more detailed exposition, interested readers are referred to Hatcher [19].

2.1. Homology

Homology groups are algebraic structures that can be used to distinguish spaces that have different topology. In other words, homology groups are topological invariant of a space [19]. In our case, we will deal with spaces that are simplicial complexes. So, we review simplicial homology here. Let $\mathbb{K}$ denote a simplicial complex. To define the homology groups of $\mathbb{K}$, we need *chain* groups and *cycle* groups of $\mathbb{K}$ as defined below.

A *k-chain* is a finite linear combination of k-simplices. In this paper, we assume that the coefficient ring used for this combination is $\mathbb{Z}_2$, that is, coefficients are either 0 or 1. For example, $c = \sigma_1 + \sigma_2 + \sigma_3$ is a 2-chain where σ_i's are triangles. Notice that a simplex σ added to itself gives 0, that is, $\sigma + \sigma = 0$. The set of all k-chains forms a chain group C_k under addition. The *boundary* of a k-simplex σ, denoted as $\partial_k \sigma$, is the sum of its $(k-1)$ dimensional faces. Specifically, if $\sigma = [v_0, v_1, ..., v_k]$ is spanned by vertices v_0 up to v_k, then

$$\partial_k \sigma = \partial_k [v_0, v_1, \cdots, v_k] = \sum_{i=0}^{k} [v_0, \cdots, \hat{v_i}, \cdots v_k]$$

where the hat indicates that v_i is dropped. The boundary operator on k-simplices extends to the boundary operator of k-chains by linearity, that is, for a k-chain $c = \Sigma_{i=0}^{n} \sigma_i$, $\partial_k c = \Sigma_i \partial_k \sigma_i$.

A k-chain is called a *k-cycle* if its boundary is empty and a *k-boundary* if it is the boundary of a $(k+1)$-chain. Let Z_k and B_k denote the set of all k-cycles and k-boundaries respectively. It is easy to derive that B_k is a subgroup of Z_k which, in turn, is a subgroup of C_k. The kth *homology group* of $\mathbb{K}$, denoted as $\mathsf{H}_k(\mathbb{K})$, is the quotient group Z_k/B_k. The rank of the homology group $\mathsf{H}_k(\mathbb{K})$ is called the kth Betti number. Homology groups can also be defined for other topological spaces which are not necessarily simplicial. This is the study of singular homology [19]. In this paper, we will use $\mathsf{H}_k(\mathbb{X})$ to denote the kth homology group of $\mathbb{X}$ whether $\mathbb{X}$ is simplicial or not.

In this work, the space we will be interested in is a surface M embedded in $\mathbb{R}^3$ and the subspaces of $\mathbb{R}^3$ bounded by M. We are interested in loops in M which represent class of 1-cycles in the one dimensional homology group $\mathsf{H}_1(M)$.

2.2. Definitions

Let M be a connected closed surface in $\mathbb{R}^3$. Its genus g is the maximum number of disjoint simple loops whose removal does not disconnect M. The closed surface M separates $\mathbb{R}^3$ into two parts: inside space I and outside space O. Let $I_M = I \cup M$ and $O_M = O \cup M$. The handle and tunnel loops in M are defined as follows [9].

Definition 1. *A loop in M is a* handle loop *if the homology class carried by it is trivial (identity) in $\mathsf{H}_1(I_M)$ and non-trivial in $\mathsf{H}_1(O_M)$.*

Definition 2. *A loop in M is a* tunnel loop *if the homology class carried by it is trivial (identity) in $\mathsf{H}_1(O_M)$ and non-trivial in $\mathsf{H}_1(I_M)$.*

The following theorem says that the generating basis of handle and tunnel loops exist for any closed surface [9].

Theorem 1. *For any connected closed surface $M \subset \mathbb{R}^3$ of genus g, there exist g handle loops $\{h_i\}_{i=1}^{g}$ forming a basis for $\mathsf{H}_1(O_M)$ and g tunnel loops $\{t_i\}_{i=1}^{g}$ forming a basis for $\mathsf{H}_1(I_M)$. Furthermore, $\{[h_i]\}_{i=1}^{g}$ and $\{[t_i]\}_{i=1}^{g}$ form a basis for $\mathsf{H}_1(M)$.*

The handle and tunnel loops are also well defined in a simplicial 2-complex $\mathbb{M}$ that represents (homeomorphic to) M. In this case, the loops are constrained to the 1-skeleton of $\mathbb{M}$.

3. Persistent Homology

Persistent homology is a concept by which we can study the growth history of the topological features of a space as the space grows. In a discretized set-up, this growth can be implemented by a *filtration* of a simplicial complex. The persistence algorithm by Edelsbrunner, Letscher, and Zomorodian [12], which has also a root in size theory [3, 23], lets us record the birth and death of the topological features in terms of the homology groups as one proceeds through the filtration. We formalize

and describe the method below. Let $\mathbb{K}$ be a simplicial complex. A filtration of $\mathbb{K}$ is defined as a nested sequence of simplicial complexes $\mathbb{K}_i$ where

$$\emptyset = \mathbb{K}_{-1} \subset \mathbb{K}_0 \subset \mathbb{K}_1 \subset \cdots \subset \mathbb{K}_n = \mathbb{K}.$$

As simplices are added to go from $\mathbb{K}_i$ to $\mathbb{K}_{i+1}$, some topological features may appear or disappear. The longer a feature stays over the filtration, the more persistent it is. For our purpose of computing loops, we only care about how the first homology group of $\mathbb{K}$, $\mathsf{H}_1(\mathbb{K})$, changes over the filtration.

3.1. Pairing Algorithm

Assume that we add one simplex at a time, that is, $\mathbb{K}_i \setminus \mathbb{K}_{i-1} = \sigma_i$. So, we have an ordered sequence of simplices $\sigma_0, \sigma_1, ..., \sigma_n$ filtering the complex $\mathbb{K} = \bigcup_{i=0}^{n} \{\sigma_i\}$. All lower dimensional faces of a simplex are added before the simplex is added. For example, all three vertices and edges of a triangle are added before the triangle is added. For a simplex σ in the filtration we use $\mathsf{t}(\sigma)$ to denote the time it is added into the filtration, that is, $\mathsf{t}(\sigma) = i$ if $\sigma = \sigma_i$ in the filtration order. We say σ is *younger* than σ' if $\mathsf{t}(\sigma) > \mathsf{t}(\sigma')$.

We already mentioned that, as simplices are added, cycles are created or destroyed. If addition of a p dimensional simplex σ creates a p-cycle, then σ is called a *positive* p-simplex. Otherwise, σ must kill a $(p-1)$-cycle in which case it is called a *negative* p-simplex. The theory of persistent homology says that a negative p-simplex σ is always paired with a *unique* positive $(p-1)$-simplex σ' where σ kills a $(p-1)$-cycle created by σ'. This pairing can be detected by a *Pairing* algorithm which works as follows. The algorithm expands a $(p-1)$-cycle c starting from the boundary of σ. At any generic step, it checks if the youngest $(p-1)$-simplex σ' in c is already paired or not. If it is not, σ is detected as a negative simplex pairing with σ'. Otherwise, σ', which is guaranteed to be a positive simplex, must have been associated with a $(p-1)$-cycle c' in which its pair found it. We add c' to c. The process continues until either σ finds its pair or the resulting cycle is empty. In the latter case, σ is a positive simplex. The pseudo-code of the *Pairing* algorithm is described in Procedure 1. In our case, handle and tunnel loops are chosen from the loops killed by negative 2-simplices.

Procedure 1 Pair(σ)

```
c := ∂_p σ
let d be the youngest (p−1)-simplex in c
while d is paired and c ≠ ∅ do
    let c' be the cycle killed by the simplex paired with d
    c := c' + c
    let d be the youngest (p−1)-simplex in c
end while
if c ≠ ∅ then
    σ is negative p-simplex killing c and pairing with d
else
    σ is positive p-simplex
end if
```

4. Computing Handle and Tunnel Loops

Dey et al [10] proposed a persistence-based algorithm to compute handle and tunnel loops in closed surfaces such as 3D triangle meshes or iso-surfaces from volume data. Let the input surface M be presented as a simplicial 2-complex $\mathbb{M}$. To apply the persistence algorithm, we assume $\mathbb{M}$ sitting in $\mathbb{R}^3$ and tessellating its convex hull by a complex $\mathbb{K}$. We need that $\mathbb{M}$ be a subcomplex of $\mathbb{K}$. In this way, we have an explicit simplicial representation of I_M and O_M. We generate $\mathbb{K}$ which provides the triangulation of I_M and O_M by using *DelPSC* [7] software on 3D triangle meshes. For the case of iso-surfaces, they are obtained almost freely, see [10].

We will need only vertices, edges, and triangles of $\mathbb{K}$. In other words, only the 2-skeleton of $\mathbb{K}$ is relevant to our algorithm. Let $\mathbb{I} \subset \mathbb{K}$ and $\mathbb{O} \subset \mathbb{K}$ denote the 2-complexes whose edges and triangles lie in I and O respectively. Notice that edges and triangles of $\mathbb{I}$ and $\mathbb{O}$ are not in $\mathbb{M}$. We call these edges and triangles *volume edges* and *volume triangles* to distinguish them from those in $\mathbb{M}$. Figure 2 shows the the volume edges and triangles for *Botijo*.

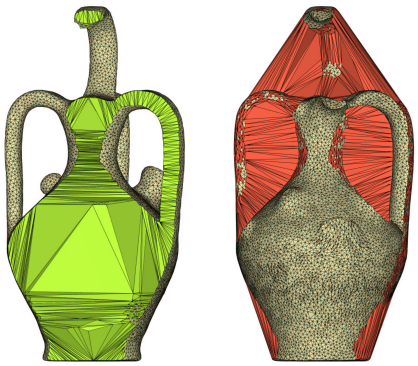


Figure 2: Interior volume complex $\mathbb{I}$ (green) and exterior volume complex $\mathbb{O}$ (red) for *Botijo*.

For computing handle and tunnel loops, a filtration of $\mathbb{K}$ is built from a sequence of simplices up to dimension two in $\mathbb{K}$. The *topological algorithm* basically runs in two steps.

- In step one, we first add all the vertices in $\mathbb{M}$, then the edges. The *Pairing* algorithm finds all positive and negative edges in $\mathbb{M}$. All triangles in $\mathbb{M}$ are then added to pair with the positive edges. Some positive edges in $\mathbb{M}$ are left unpaired. Let $\mathbb{U}$ denote the set of the unpaired edges after this step.

- In step two, we first add volume edges from $\mathbb{I}$ and $\mathbb{O}$, then the volume triangles. A handle (tunnel) loop is generated when a negative triangle from $\mathbb{I}$ ($\mathbb{O}$) is paired with an unpaired edge in $\mathbb{U}$.

It has been proved that the loops obtained in this way are topologically correct handle and tunnel loops [10]. The authors also propose the following two refinements to make the loops geometry-aware.

- For a negative triangle from $\mathbb{I}$ or $\mathbb{O}$, the *Pairing* algorithm expands a loop till it reaches an unpaired edge in $\mathbb{U}$. To get geometrically better loops, the expanding loop is declared a handle (tunnel) loop when it lies completely in the surface $\mathbb{M}$ for the first time even if it has not reached the unpaired edge.
- Each time a handle or tunnel loop is generated, a cross section of the volume bounded by $\mathbb{M}$ gets filled. To get loops of small size, small cross sections should get filled first. To this end, the volume triangles are associated with geodesic size (see definition in Section 5.2) and then are added in increasing order of the geodesic size into the filtration.

The algorithm of [10] is simple and runs fast in general, but we show in Section 5 that this algorithm could be made much more efficient with some novel changes. In particular, these modifications make the algorithm scalable without sacrificing accuracy.

5. Speed Up

Authors in [10] reported the timing of their algorithm in two parts: geodesic size computation and persistence-based loop computation. Table 1 shows the timings of these two steps[1]. We see that the time for large models could take several hours. We propose techniques that speed up each stage quite substantially.

Data	Genus	Size	Geod	Loop
Knotty cup	2	5.4k, 31.9k	2.47	0.67
Mother	4	19.5k, 117k	10.68	3.16
Molecule	1	19.9k, 115k	4.08	0.76
Botijo	5	30.0k, 176k	15.93	1.78
Casting	9	31.9k, 169k	19.34	10.63
Kitten	1	62.7k, 346k	98.09	4.97
Pegasus	5	70.8k, 403k	88.75	24.01
Buddha	9	109k, 614k	127.42	12.49
Vase	6	122k, 590k	530.92	22.03
Hip	10	173k, 996k	480.93	199.35
Children	393*	199k, 1168k	1049.74	2644.09
Filigree	49	220k, 1196k	473.03	386.41
Fusee	18	244k, 1368k	1435.67	199.42
Heptoroid	22	297k, 1757k	1249.12	396.65
Gearbox	78	478k, 2582k	1527.58	9778.63
Colon	160	854k, 4966k	9014.22	27920.15
Atom	1	3.3k, 41.8k	0.10	0.14
Aneurysm	11	23k, 5210k	6.93	116.80
Engine 1	20	157k, 2766k	392.70	148.50
Engine 2	20	629k, 20040k	5895.09	2275.82

Table 1: Times (seconds) broken into two parts, one for geodesic size computation (Geod column), and the other for loop computation (Loop column). The genus of each model is shown in the Genus column. (n,m) in Size column indicates n surface triangles in $\mathbb{M}$ and m volume triangles in $\mathbb{K}\setminus\mathbb{M}$. (* *Children* is a non-manifold mesh, 393 is its first Betti number).

5.1. Accelerating Loop Computation

A handle or tunnel loop is generated once a negative volume triangle in $\mathbb{K}\setminus\mathbb{M}$ is paired with an unpaired edge in $\mathbb{U}$. All handle and tunnel loops are obtained once all edges in $\mathbb{U}$ are paired with the corresponding negative volume triangles. The *Pairing* algorithm is thus required to run on each volume triangle. Therefore, any improvement in the pairing procedure makes a substantial improvement in total computation time.

We make a key observation in the *Pairing* algorithm in Section 5.1.1 based on which a modification of this algorithm is made for the volume triangles. We also make another observation in Section 5.1.2 which suggests a particular filtration of the surface simplices as opposed to the arbitrary filtration proposed in [10]. The new surface filtration together with the modified *Pairing* algorithm speed up the loop computation enormously as confirmed by our experiments.

5.1.1. Observation

Given a volume triangle σ, each iteration of the while loop in the *Pairing* algorithm actually expands the loop c, with initial $c = \partial_2\sigma$ being the boundary of the triangle. This expansion becomes more time consuming if more iterations are needed.

Notice that if a volume triangle σ is positive, c eventually becomes empty, resulting in a waste of time spent on expanding c. Although statistically positive volume triangles only account for roughly 5% of all the volume triangles in $\mathbb{K}\setminus\mathbb{M}$, the accumulated wasted time becomes high especially on larger models. We observe that, if a positive volume triangle σ could be detected early during the expansion process, we could exit the *Pairing* procedure earlier.

Recall that $\mathsf{t}(\sigma)$ denote the time stamp of the simplex σ when it is added to the filtration. Let d be the youngest edge in loop c, i.e., $\mathsf{t}(d)$ is the largest among all edges in c. Since the youngest edge in c must be positive, d is positive. Let u be the oldest edge in $\mathbb{U}$, that is, $\mathsf{t}(u)$ is the smallest among all unpaired edges in $\mathbb{U}$. We have the following observation.

Observation 1. *If $\mathsf{t}(d) < \mathsf{t}(u)$ at any point of time during the expansion of c, σ is positive.*

Proof 1. *We prove that c would be eventually empty when $\mathsf{t}(d) < \mathsf{t}(u)$ is verified at any point of the loop expansion process. This implies that σ is positive.*

Assume that $c \neq \emptyset$ in the end. Then, σ is a negative triangle and it is paired with a positive edge, say d'. The edge d' must be the youngest edge in c when it was discovered. In each iteration of the loop expansion process, the youngest edge in a new loop is no more younger than that of the previous loop. This means $\mathsf{t}(d') \leq \mathsf{t}(d)$. But, d' being an edge in $\mathbb{U}$ satisfies $\mathsf{t}(u) \leq \mathsf{t}(d')$ since u is the oldest unpaired surface edge. This implies $\mathsf{t}(u) \leq \mathsf{t}(d)$, contradicting the assumption that $\mathsf{t}(d) < \mathsf{t}(u)$.

Based on this observation, we modify the *Pairing* procedure for volume triangles so that less iterations are made in the

[1] All experiments in this paper are done on a DELL machine which has Intel Xeon CPU 2.66GHz and 4G Memory.

Procedure 2 Pair-Volume-Triangle(σ)

let u be the oldest edge in $\mathbb{U}$
$c := \partial_2 \sigma$
let d be the youngest edge in c
while d is paired and $c \neq \emptyset$ **do**
  let c' be the cycle killed by the simplex paired with d
  $c := c' + c$
  let d be the youngest edge in c
  if $\mathrm{t}(d) < \mathrm{t}(u)$ **then**
    set $c := \emptyset$
  end if
end while
if $c \neq \emptyset$ **then**
  σ is negative edge killing c and pairing with d
else
  σ is positive edge
end if

while-loop for the positive volume triangles. The pseudo-code is shown in Procedure 2.

5.1.2. Ordering Surface Simplices for the Filtration

The observation given in Section 5.1.1 leads us to think that if we could make all the edges in $\mathbb{U}$ as young as possible among all surface edges in the filtration, we could save more time in Procedure 2. To reach this goal, we give a method of creating a new filtration for simplices in the surface.

For simplicity, let us suppose that $\mathbb{M}$ has only one component and it is closed. In the topological algorithm [10], surface simplices are added in arbitrary order except that a simplex is added only if its lower dimensional boundary simplices appear earlier in the filtration. For instance, a triangle is added only after its vertices and edges are added.

When a surface edge is added, the *Pairing* algorithm decides whether it is positive or negative. Each positive edge generates a cycle lying in the 1-skeleton of $\mathbb{M}$. All surface triangles are added next to pair with the positive edges, with each pair (e,t) corresponding to the fact that the negative triangle t kills the cycle generated by the positive edge e. After processing all surface triangles, if $\mathbb{M}$ has genus g, then $2g$ unpaired edges are left and saved in $\mathbb{U}$. In order to make the edges in $\mathbb{U}$ as young as possible, we propose the following strategy to get a filtration for the surface simplices.

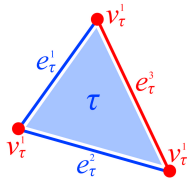
Given an arbitrary triangle $\tau \in \mathbb{M}$, we expand a simplicial 2-complex $\mathbb{M}_\tau$ maintaining the invariant that $\mathbb{M}_\tau$ is a topological disk all the time. We stop when no more triangle can be added to $\mathbb{M}_\tau$ without violating the topological disk condition. Notice that this triangle-by-triangle exploration of the surface is similar to the one used in [11, 16] though the end results are different since our conditions for growth are slightly different. We grow $\mathbb{M}_\tau$ by adding a triangle σ from $\mathbb{M} \setminus \mathbb{M}_\tau$ to $\mathbb{M}_\tau$. Before adding σ, we add each of its three vertices unless it is already in $\mathbb{M}_\tau$ and then similarly each of its edges. The criteria for choosing σ are as follows:

- σ should be adjacent to the boundary (expanding frontier) of $\mathbb{M}_\tau$.
- $\mathbb{M}_\tau$ remains a topological disk after the addition of σ.

The second criterion can be verified easily by checking the Euler number of $\mathbb{M}_\tau$ which is given by the alternating sum of the number of vertices, edges and triangles in $\mathbb{M}_\tau$. The Euler number should remain 1 after adding σ and its sub-simplices for $\mathbb{M}_\tau$ to remain a topological disk. Figure 3 shows the expansion of $\mathbb{M}_\tau$ on *Pegasus*. We have the following observation which is key to claim that unpaired edges are pushed late in the surface filtration.

Observation 2. *Let $\mathbb{M}_\tau \subset \mathbb{M}$ denote the maximal surface obtained by the above expansion procedure. Let $F_\tau = \sigma_1, \sigma_2, ..., \sigma_k$ denote the filtration of $\mathbb{M}_\tau$ induced by the expansion process. Pairing algorithm applied to the filtration F_τ of $\mathbb{M}_\tau$ does not generate any unpaired edges in $\mathbb{M}_\tau$.*

Proof 2. *We prove it by induction on the number of triangles in $\mathbb{M}_\tau$.*



For the base case when $\mathbb{M}_\tau$ contains only one triangle τ, F_τ is $\{v_\tau^1, v_\tau^2, v_\tau^3, e_\tau^1, e_\tau^2, e_\tau^3, \tau\}$ where v_τ's and e_τ's denote the vertices and edges of τ respectively. After adding simplices in F_τ, we have two negative edges e_τ^1 and e_τ^2, one positive edge e_τ^3 that is paired with τ. Thus, no unpaired edge exists in $\mathbb{M}_\tau$ in this case.

Assume inductively that there is no unpaired edge in $\mathbb{M}_\tau$ after processing the kth triangle in the filtration F_τ. Let μ be the $(k+1)$th triangle to be added next. There are two possible cases depending on how μ is adjacent to the boundary of $\mathbb{M}_\tau$, see Figure 4.

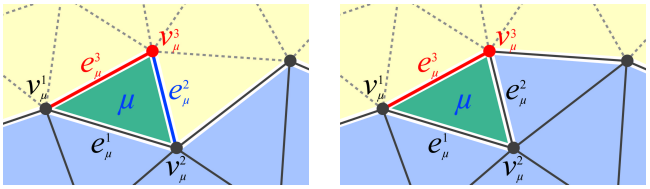


Figure 4: Two possible cases when adding triangle μ (green) to $\mathbb{M}_\tau$ (light blue).

- *Case 1: only one edge of μ is in $\mathbb{M}_\tau$. Suppose the new simplices to be added to $\mathbb{M}_\tau$ are v_μ^3, e_μ^2, e_μ^3, and μ. After these simplices (in order) are added to the filtration, we will have negative e_μ^2 and positive e_μ^3 that is paired with μ. In this case, $\mathbb{M}_\tau$ is updated as $\mathbb{M}_\tau \cup \{v_\mu^3, e_\mu^2, e_\mu^3, \mu\}$ in which no unpaired edge remains.*
- *Case 2: two edges of μ are in $\mathbb{M}_\tau$. Suppose the new simplices to be added to $\mathbb{M}_\tau$ are e_μ^3 and μ. After these simplices (in order) are added to the filtration, we will have positive edge e_μ^3 that is paired with μ. In this case, $\mathbb{M}_\tau$ is updated as $\mathbb{M}_\tau \cup \{e_\mu^3, \mu\}$ in which no unpaired edge remains.*

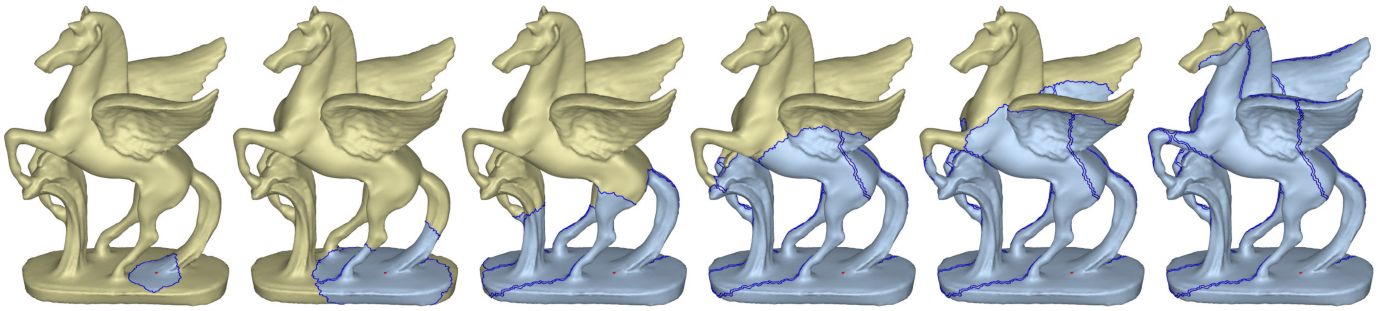


Figure 3: Expanding a surface on *Pegasus* from triangle in red. Pictures from left to right show the expanding surfaces (light blue). The expanding frontiers of the expanding surfaces are the curves in blue.

No other case is possible since otherwise Euler number of $\mathbb{M}_\tau$ changes.

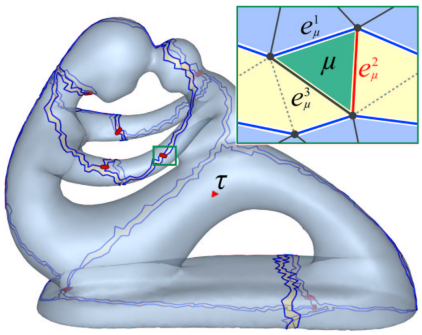


Figure 5: The maximal expanded surface $\mathbb{M}_\tau$ on *Mother* is shown in light blue bounded by the blue curve. Addition of any new triangle, say μ, and its edges e^2_τ and e^3_τ from $\mathbb{M}\setminus\mathbb{M}_\tau$ (light yellow) to $\mathbb{M}_\tau$ generate a non-trivial cycle in $\mathbb{M}_\tau \cup \{e^2_\mu, e^3_\mu, \mu\}$. $2g$ unpaired edges (red) after step one of the topological algorithm are all located in $\mathbb{M}\setminus\mathbb{M}_\tau$ and are all younger than the edges in $\mathbb{M}_\tau$.

Let $\mathbb{M}_\tau$ denote the *maximal expanded surface*, i.e., addition of any new triangle μ and its edges e^1_μ, e^2_μ, and e^3_μ to $\mathbb{M}_\tau$ violates the Euler number criterion for expansion. It can be easily verified that, to violate the Euler number condition, there must be exactly two edges, say e^2_μ and e^3_μ of μ, which are not in $\mathbb{M}_\tau$ yet. This is illustrated by the upper right picture in Figure 5. A new non-trivial cycle in $\mathbb{M}_\tau \cup \{e^2_\mu, e^3_\mu, \mu\}$ is generated by adding μ and its boundary edges e^2_μ and e^3_μ. This means one of these two edges, say e^2_μ, will be the first (oldest) unpaired edge in $\mathbb{U}$. Adding all simplices from $\mathbb{M}\setminus\mathbb{M}_\tau$ to $\mathbb{M}_\tau$ generates $2g$ unpaired edges in $\mathbb{U}$. Notice that all edges in $\mathbb{U}$ are in $\mathbb{M}\setminus\mathbb{M}_\tau$. All of them except for e^2_μ itself have time stamps larger than $\mathrm{t}(e^2_\mu)$. Since $\mathbb{M}_\tau$ covers most edges in $\mathbb{M}$, this expansion method makes the edges in $\mathbb{U}$ younger than most of the edges in $\mathbb{M}$. Figure 5 shows the edges in $\mathbb{U}$ in *Mother*.

Using the filtration of $\mathbb{M}_\tau$ and applying the modified *Pairing* algorithm on the volume triangles provide huge speed up as verified by the experimental results in Section 6.

5.2. Geodesic Size Estimation

Adding volume triangles in increasing order of the geodesic size helps generate geometrically small loops. However, exact calculation of the geodesic size for all the volume triangles becomes increasingly demanding as the input size becomes larger. As Table 1 shows, the number of triangles in $\mathbb{K}\setminus\mathbb{M}$ is roughly five times of those in $\mathbb{M}$. In this section, we present an approximation scheme to estimate the geodesic size of the volume triangles fast. Our experimental results in Section 6 show that a good approximation can effectively reduce the computation time while preserving the loop qualities.

According to [10], a volume triangle t has a geodesic size defined as $g(t) = max_{e\in t} g(e)$, where e is an edge of t and $g(e)$ is the geodesic size of e defined as follows. For two vertices v and v' in $\mathbb{M}$, let the geodesic distance between v and v' be the length of the shortest path connecting v and v' in the 1-skeleton of $\mathbb{M}$. Let a and b be the two end vertices of edge e. The geodesic size of e, $g(e)$, is defined as the geodesic distance between a' and b', where a' and b' are the closest vertices in $\mathbb{M}$ to a and b respectively. Note that if a and b are in $\mathbb{M}$, then $a' = a$ and $b' = b$. Figure 6 shows the geodesic for an outside volume triangle of iso-surface *Atom*.

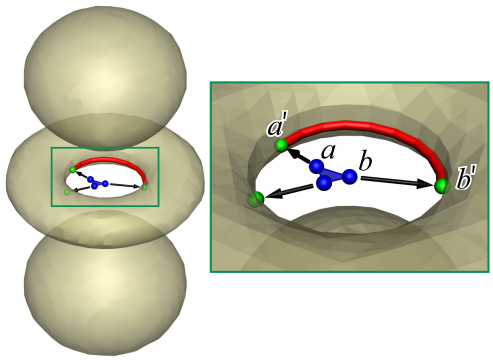


Figure 6: The geodesic of a volume triangle (blue) is shown as the geodesic path (red) between a' and b' on *Atom*. Two vertices of this volume triangle are shown as a and b whose closest vertices on *Atom* are a' and b' respectively.

Since the computation of the geodesics for the volume triangles requires computing the same for the volume edges, the computation time depends on the number of volume edges, or

the size of the set $\{(v_i, v_j) | v_i \textit{ and } v_j \textit{ are end vertices of a volume edge}\}$. We can cut down the geodesic computation time by reducing the cardinality of this set. For this purpose, we first compute a subsample S of the vertices in $\mathbb{M}$. We then divide the vertices in $\mathbb{M}$ into groups, with all vertices in a group having the same closest sample point in S. Let e, a', and b' be defined as before, and let a'' and b'' in S be the closest samples to a' and b' respectively. Then $g(e)$ can be approximated by the geodesic distance between a'' and b'' in $\mathbb{M}$.

We use a parameter δ to control the density of the subsample. Starting from an arbitrary vertex v_0 in $\mathbb{M}$, we build S incrementally using the following Procedure 3.

Procedure 3 Subsampling($\mathbb{M}$, v_0, δ)

1: unmark all vertices in $\mathbb{M}$
2: $S := \{v_0\}$
3: compute the set of vertices, S_0, whose geodesic distances to v_0 are less than δ in $\mathbb{M}$, set v_0 as the closest sample to each vertex in S_0, mark v_0 and all vertices in S_0
4: **while** there are unmarked vertices in $\mathbb{M}$ **do**
5: pick an unmarked vertex v_i
6: $S := S \cup \{v_i\}$
7: compute the set of vertices, S_i, whose geodesic distances to v_i are less than δ in $\mathbb{M}$, set v_i as the closest sample to each vertex in S_i, mark v_i and all vertices in S_i
8: **end while**

Let S_i and S_j be the set of vertices considered in step 7 of Procedure 3. For an edge (v, v') where $v \in S_i$ and $v' \in S_j$ we assign $g(v, v') = g(v_i, v_j)$ where v_i and v_j are associated to S_i and S_j according to the above subsampling procedure. Since the size of S is smaller than the total number of vertices in $\mathbb{M}$, geodesic computation becomes faster as the experiments show in Section 6. Figure 7 shows different subsamples on *Hip*. Figure 8 shows the estimated geodesic size for an outside volume edge of *Mother*.

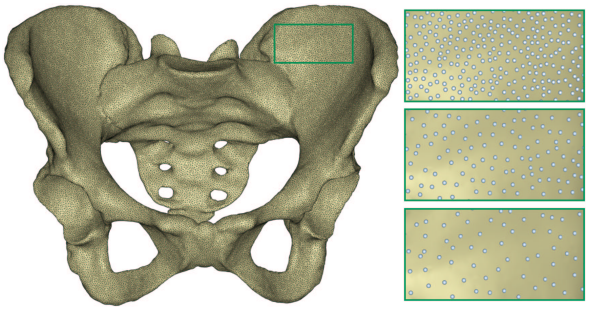


Figure 7: *Hip* has 86226 vertices. Three different subsamples, with density decreasing from top to bottom, result in 20702, 8724, and 4760 sample points respectively.

6. Results

We tested our new method on the data sets in Table 1, and the new timings are shown in Table 2. The handle and tunnel loops in the new models that are not used by [10] are shown in Figure 9.

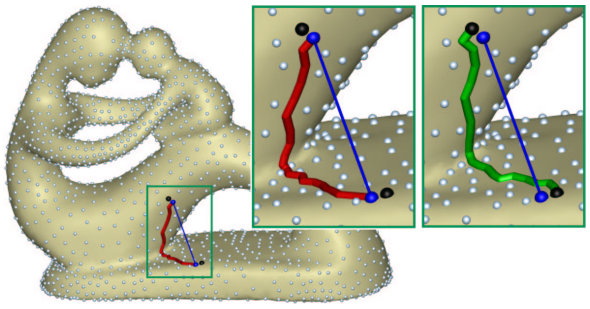


Figure 8: A subsample (light blue balls) is shown in *Mother*. The geodesic size of a volume edge (blue) is the length of the geodesic path (red) in surface. Its estimated geodesic size is the length of the geodesic path (green) between two black vertices which are the closest samples to the two end vertices (blue) of the volume edge.

Data	Geod	G-ratio	Loop	L-ratio
Knotty cup	0.06	40.00	0.14	4.78
Mother	0.50	21.36	0.41	7.70
Molecule	0.33	12.36	0.28	2.71
Botijo	0.73	21.82	0.58	3.06
Casting	1.00	19.34	0.71	14.97
Kitten	3.43	28.60	1.07	4.64
Pegasus	4.48	19.81	1.08	22.23
Buddha	6.32	20.16	2.44	5.12
Vase	14.62	36.31	4.31	5.11
Hip	21.90	21.96	3.52	56.63
Children	62.95	16.67	7.68	344.28
Filigree	17.19	27.52	9.64	40.08
Fusee	49.98	28.72	13.37	14.91
Heptoroid	40.56	30.80	20.53	19.32
Gearbox	75.08	20.34	26.30	371.81
Colon	448.84	20.08	96.62	288.96
Atom	0.01	10.00	0.12	1.17
Aneurysm	0.98	7.07	119.43	0.98
Engine 1	22.76	17.25	33.36	4.45
Engine 2	317.86	18.54	393.12	5.78

Table 2: Geod and Loop columns show the new times for geodesic size computation and loop computation respectively. G-ratio and L-ratio columns show the speed up ratios of these two computations compared with the original method given in [10]. The ratios in the G-ratio column are obtained by choosing a good subsample density that preserves good loop qualities for each model.

For the loop computation, the speed up ratios are very prominent on large models such as *Hip*, *Children*, *Gearbox*, and *Colon* as seen from Table 2. However, the gain is not so dramatic for iso-surfaces. One reason is that the loop computation in an iso-surface is generally much faster than that in a 3D model of similar size as reported in [10]. Another possible reason is that the unpaired edges in $\mathbb{U}$ for iso-surfaces are already very young even without any special ordering of the surface simplices.

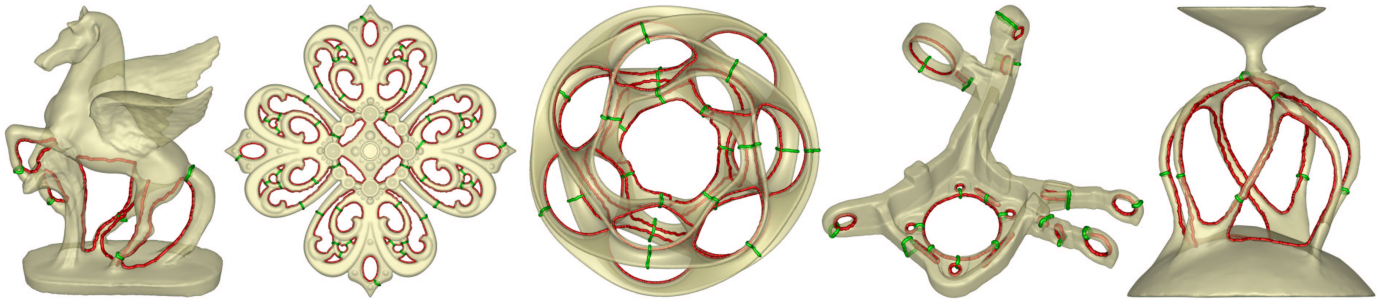


Figure 9: Handle (green) and tunnel (red) loops in *Pegasus*, *Filigree*, *Heptoroid*, *Fusee*, and *Vase* from left to right.

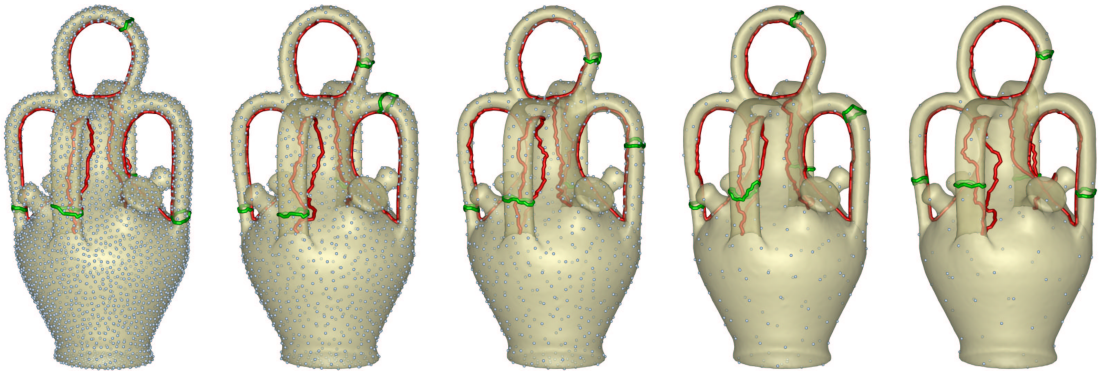


Figure 10: Handle (green) and tunnel (red) loops in *Botijo* under different subsamples, with density decreasing from left to right. The light blue balls are the sample points. The speed up ratio of the geodesic computation for each subsample is shown underneath each picture.

Data	$\mathbf{S}_1$	**G-r**$_1$	$\mathbf{S}_2$	**G-r**$_2$	$\mathbf{S}_3$	**G-r**$_3$
Mother	23.9%	3.40	9.3%	6.68	3.2%	16.95
Botijo	24.1%	3.14	10.1%	6.40	3.5%	15.77
Kitten	24.0%	3.55	10.1%	8.19	3.4%	20.96
Pegasus	23.9%	3.17	10.1%	6.18	3.4%	14.53
Buddha	24.0%	3.19	10.1%	6.44	3.5%	14.40
Vase	23.7%	3.98	10.0%	9.29	3.4%	25.21
Filigree	23.8%	3.65	10.1%	7.58	3.5%	19.82
Fusee	23.9%	3.86	10.1%	8.42	3.4%	21.26
Heptoroid	23.8%	3.75	10.0%	8.47	3.5%	22.23
Colon	23.8%	3.14	9.4%	6.39	3.2%	15.51

Table 3: Each S_i and G-r_i column pair show the percentage of the sample points on each model and the corresponding speed up ratio of the geodesic size computation respectively.

For the geodesic size computation, different subsample densities result in different speed up ratios as shown on some selected models in Table 3. The loop qualities are also influenced by subsamplings as shown by an example in Figure 10. Naturally, the denser the subsample is, the smaller is the speed up ratio for the geodesic computation and the lesser is the effect on loop qualities.

7. Conclusion

We have revisited a persistence-based method for fast computation of handle and tunnel loops recently proposed in [10]. Specifically, the loop computation is accelerated by designing a new surface filtration and applying a modified persistence algorithm on volume triangles. We use geodesic estimation to speed up the geodesic size computation. The software called HanTun has been released from authors' web pages [8].

Our experiments show that the computed loops are small and perhaps are very close to optimal. It would be nice to prove that the computed loops do approximate the optimal loops in some sense. Currently, we are focusing our research on this aspect.

Acknowledgments

We acknowledge that the many of the models used in this paper are taken from AIM@SHAPE database.

References

[1] Attene, M., Biasotti, S., Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., 2006. Computational methods for understanding 3d shapes. Computers & Graphics 30(3), 323–333.

[2] Ben-chen, M., Gotsman, C., Bunin, G., 2008. Conformal flattening by curvature prescription and metric scaling. Computer Graphics Forum (Proc. Eurographics) 27(2).

[3] Biasotti, S., Floriani, L. D., Falcidieno, B., Frosini, P., Giorgi, D., Landi, C., Papaleo, L., Spagnuolo, M., 2008. Describing shapes by geometrical-topological properties of real functions. ACM Computing Surveys 40(4), 1–87.

[4] Biasotti, S., Giorgi, D., Spagnuolo, M., Falcidieno, B., 2008. Reeb graphs for shape analysis and applications. Theoretical Comput. Sci. 392, 5–22.

[5] Bischoff, S., Kobbelt, L., 2005. Structure preserving cad model repair. Computer Graphics Forum 24(3), 527–536.

[6] Cole-McLaughlin, K., Edelsbrunner, H., Harer, J., Natarajan, V., Pascucci, V., 2004. Loops in reeb graphs of 2-manifolds. Discrete Comput. Geom. 32(2), 231–244.

[7] Dey, T. K., Levine, J. A., 2008. Delaunay meshing of piecewise smooth complexes without expensive predicates. Tech. Report: OSU-CISRC-7108-TR40.

[8] Dey, T. K., Li, K., 2008. HanTun software for computing handle and tunnel loops in 3d models: http://www.cse.ohio-state.edu/∼tamaldey/hantun.html.

[9] Dey, T. K., Li, K., Sun, J., 2007. On computing handle and tunnel loops. IEEE Proc. Internat. Conf. Cyberworlds, 357–366.

[10] Dey, T. K., Li, K., Sun, J., Cohen-Steiner, D., 2008. Computing geometry-aware handle and tunnel loops in 3d models. Proc. SIGGRAPH 2008, 45:1–45:9.

[11] Dey, T. K., Schipper, H., 1995. A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection. Discrete Comput. Geom. 14, 93–110.

[12] Edelsbrunner, H., Letscher, D., Zomorodian, A., 2002. Topological persistence and simplification. Discrete Comput. Geom. 28, 403–412.

[13] Éric Colin de Verdière, Erickson, J., 2006. Tightening non-simple paths and cycles on surfaces. Proc. 17th ACM-SIAM Sympos. Discrete Algorithms. 192–201.

[14] Éric Colin de Verdière, Lazarus, F., 2005. Optimal system of loops on an orientable surface. Discrete Comput. Geom. 33, 627–636.

[15] Erickson, J., Whittlesey, K., 2005. Greedy optimal homotopy and homology generators. Proc. 16th ACM-SIAM Sympos. Discrete Algorithms, 1038–1046.

[16] Gu, X., Gortler, S., Hoppe, H., 2002. Geometry images. Proc. SIGGRAPH 2002, 355–361.

[17] Gu, X., Yau, S.-T., 2003. Global conformal surface parameterization. ACM Sympos. Geometry Processing 2003, 127–137.

[18] Guskov, I., Wood, Z., 2001. Topological noise removal. Proc. Graphics Interface 2001, 19–26.

[19] Hatcher, A., 2002. Algebraic Topology. Cambridge University Press.

[20] Mortara, M., Patané, G., Spagnuolo, M., 2006. From geometric to semantic human body models. Computers & Graphics 30, 185–196.

[21] Pascucci, V., Scorzelli, G., Bremer, P.-T., Mascarenhas, A., 2007. Robust on-line computation of reeb graphs: simplicity and speed. Proc. SIGGRAPH 2007, 58:1–58:9.

[22] Sheffer, A., Hart, J., 2002. Seamster: inconspicuous low-distortion texture seam layout. Proc. IEEE Visualization. 291–298.

[23] Verri, A., Uras, C., Frosini, P., Ferri, M., 1993. On the use of size functions for shape analysis. Biological Cybernetics 70, 99–107.

[24] Wood, Z., Hoppe, H., Desbrun, M., Schröder, P., 2004. Removing excess topology from isosurfaces. ACM Trans. Graphics 23, 511–533.

[25] Zhou, Q.-Y., Ju, T., Hu, S.-M., 2007. Topology repair of solid models using skeletons. IEEE Transactions on Visualization and Computer Graphics 13, 675–685.

[26] Zomorodian, A., 2005. Topology for computing. Cambridge university press.

[27] Zomorodian, A., Carlsson, G., 2008. Localized homology. Computational Geometry: Theory & Applications 41(3), 126–148.