Topological Persistence and Simplification *

Herbert Edelsbrunner[†], David Letscher[‡], and Afra Zomorodian[§]

Abstract

We formalize a notion of topological simplification within the framework of a filtration, which is the history of a growing complex. We classify a topological change that happens during growth as either a feature or noise depending on its life-time or persistence within the filtration. We give fast algorithms for computing persistence and experimental evidence for their speed and utility.

Keywords. Computational geometry, computational topology, homology groups, filtrations, alpha shapes.

1 Introduction

The need for automated topological simplification has been articulated in the computer graphics and geometric modeling literature. This paper proposes a solution in which scale is used to assess the persistence of topological attributes and to prioritize simplification steps. After describing a new notion of topological simplification, we summarize the contributions of this paper and contrast them with prior work.

Topological simplification. We use homology to measure the topological complexity of a point set in $\mathbb{R}^3$. The simplest non-empty sets under this measure are the ones that contract to a point. Each such set consists of one component and has no other non-trivial homological attributes. A general set in $\mathbb{R}^3$ has β_0 components, β_1 tunnels, and β_2 voids. We consider topological complexity to be expressed by $\beta_0, \beta_1, \beta_2$, the Betti numbers of the set. As such, we understand topological simplification as a process that decreases Betti numbers. To do this in a geometrically meaningful manner, we need a way of assessing the importance of topological attributes. Once we have such a numerical assessment, we naively remove attributes in the order of increasing importance. At any moment during this process, we may call the removed attributes topological noise and the remaining ones topological features.

There are three technical difficulties with this approach. The first is the identification of subsets expressing the non-trivial topological attributes that are measured by homology groups. The second is the measurement of the importance of these subsets. The third is the elimination of a topological attribute with a minimum number of side-effects. We overcome these difficulties in this paper and describe a simplification process as envisioned above.

Approach and Results. We restrict our attention to sets represented by finite simplicial complexes in $\mathbb{R}^3$. For practical reasons, moreover, we focus on particular subcomplexes of Delaunay triangulations called alpha complexes [4]. We receive essential help in overcoming some technical difficulties by assuming a filtration which places the complex within an evolutionary growth process. Given a filtration, the main contributions of this paper are:

(i) the definition of persistence for Betti numbers and non-bounding cycles,

(ii) an efficient algorithm to compute persistence,

(iii) a simplification algorithm based on persistence.

Prior and related work. As mentioned earlier, we use homology groups and Betti numbers which were developed and refined during the first half of the twentieth century. We refer to Munkres [9] for a description that is reasonably accessible to non-specialists. Spectral sequences are the by-product of a divide-and-conquer method for computing homology groups and Betti numbers [7]. These sequences form a framework within which our result on persistent Betti numbers may be placed. The algorithm we develop for computing persistence of non-bounding cycles is based on the incremental Betti number algorithm of Delfinado and Edelsbrunner [2]. Three-dimensional alpha shapes and complexes may be found in Edelsbrunner and Mücke [4]. The problem of topological simplification was also approached by El-Sana and Varshney [5] using alpha shape inspired ideas of geometric growth.

* Research by the first and third authors is partially supported by ARO under grant DAAG55-98-1-0177. Research by the first author is also partially supported by NSF under grant CCR-97-12088.

† Department of Computer Science, Duke University, Durham, and Raindrop Geomagic, Research Triangle Park, North Carolina.

‡ Department of Mathematics, Oklahoma State University, Stillwater, Oklahoma.

§ Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

There is a large body of parallel work on iso-surfaces or level sets of 3-dimensional density functions. We refer to Milnor [8] for the mathematics and to Sethian [11] for a numerical view. A density function is a map $F : \mathbb{R}^3 \to \mathbb{R}$ and an iso-surface $F^{-1}(\alpha)$ is the preimage of a constant image value α. The sequence of iso-surfaces obtained by increasing α represents a growth process similar to that represented by a filtration. Specifically, simplices in a filter correspond to critical points of a density function. In this context, topological simplification means reducing the number of critical points. This process is related to smoothing or simplifying the graph of F, which is a 3-dimensional manifold in $\mathbb{R}^4$.

Outline. Section 2 reviews alpha complexes and homology groups. Section 3 introduces persistence for Betti numbers and non-bounding cycles. Section 4 describes an algorithm that computes persistence. Section 5 formulates simplification algorithms based on persistence. Section 6 provides experimental evidence for the speed and utility of these algorithms. Section 7 concludes the paper.

2 Background

This section introduces the background we need to define and compute topological persistence. We begin with alpha complexes, continue with homology groups for $\mathbb{Z}_2$ coefficients, and end with the incremental algorithm for computing Betti numbers.

Alpha complexes. A *spherical ball* $\hat{u} = (u, U^2) \in \mathbb{R}^3 \times \mathbb{R}$ is defined by its center u and square radius U^2. If $U^2 < 0$, the radius is imaginary and so is the ball. The *weighted distance* of a point from a ball is $\pi_{\hat{u}}(x) = \|x - u\|^2 - U^2$. Note that a point $x \in \mathbb{R}^3$ belongs to the ball iff $\pi_{\hat{u}}(x) \le 0$, and it belongs to the bounding sphere iff $\pi_{\hat{u}}(x) = 0$. Let S be a finite set of balls. The *Voronoi region* of $\hat{u} \in S$ is the set of points for which $\hat{u}$ minimizes the weighted distance,

$$V_{\hat{u}} = \{x \in \mathbb{R}^3 \mid \pi_{\hat{u}}(x) \le \pi_{\hat{v}}(x), \forall \hat{v} \in S\}.$$

The Voronoi regions decompose the union of balls into convex cells of the form $\hat{u} \cap V_{\hat{u}}$, as illustrated in Figure 1. Any two regions are either disjoint or they overlap along a shared portion of their boundary. We assume general position, where at most four (three in $\mathbb{R}^2$) Voronoi regions can have a non-empty common intersection. In practice, we simulate general position computationally [3]. Let $T \subseteq S$ have the property that its Voronoi regions have a non-empty common intersection, and consider the convex hull of the corresponding centers, $\sigma_T = \operatorname{conv}\{u \mid \hat{u} \in T\}$. General position implies that σ_T is a *k-simplex*, where $k = \operatorname{card} T - 1$. The *dual complex* of S is the collection of simplices constructed in this manner,

$$K = \{\sigma_T \mid T \subseteq S, \bigcap_{\hat{u} \in T} (\hat{u} \cap V_{\hat{u}}) \ne \emptyset\}.$$

Figure 1 illustrates a 2-dimensional example of this construction.

Any two simplices in K are either disjoint or they intersect in a common face, which is a simplex of smaller dimension. Furthermore, if $\sigma \in K$, then all faces of σ are simplices in K. A set of simplices with these two properties is a *simplicial complex* [9]. A *subcomplex* is a subset $L \subseteq K$ that is itself a simplicial complex.

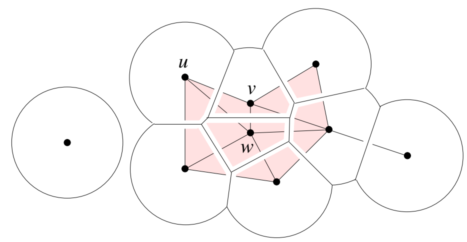

Figure 1. Union of nine disks, convex decomposition using Voronoi regions, and dual complex.

Chains, cycles, boundaries. Let K be a simplicial complex in $\mathbb{R}^3$. A *k-chain* is a subset of k-simplices in K. We define *addition* of chains with integer coefficients modulo 2. In other words, the sum of two k-chains c, d is the symmetric difference of the two sets,

$$c + d = (c \cup d) - (c \cap d),$$

which is commutative. The set of all k-chains together with addition form a group denoted as C_k. The empty set is the zero element of C_k. There is a chain group for every integer k, but for a complex in $\mathbb{R}^3$, only the ones for $0 \le k \le 3$ may be non-trivial. The *boundary* $\partial_k(\sigma)$ of a k-simplex σ is the collection of its $(k-1)$-dimensional faces, which is a $(k-1)$-chain. The *boundary* of a k-chain is the sum of the boundaries of its simplices, $\partial_k(c) = \sum_{\sigma \in c} \partial_k(\sigma)$. Each boundary operator is a homomorphism $\partial_k : \mathsf{C}_k \to \mathsf{C}_{k-1}$ and the collection of boundary operators connect the chain groups into a *chain complex*,

$$\ldots \to \emptyset \to \mathsf{C}_3 \xrightarrow{\partial_3} \mathsf{C}_2 \xrightarrow{\partial_2} \mathsf{C}_1 \xrightarrow{\partial_1} \mathsf{C}_0 \xrightarrow{\partial_0} \emptyset \to \ldots.$$

The *kernel* of ∂_k is the collection of k-chains with empty boundary and the *image* of ∂_k is the collection of $(k-1)$-chains that are boundaries of k-chains,

$$\begin{aligned} \ker \partial_k &= \{c \in \mathsf{C}_k \mid \partial_k(c) = \emptyset\}, \\ \operatorname{im} \partial_k &= \{d \in \mathsf{C}_{k-1} \mid \exists c \in \mathsf{C}_k : d = \partial_k(c)\}. \end{aligned}$$

A *k-cycle* is a k-chain in the kernel of ∂_k and a *k-boundary* is a k-chain in the image of ∂_{k+1}. The collections Z_k of k-cycles and B_k of k-boundaries together with addition form subgroups of C_k. An essential property of the boundary operators is that the boundary of every boundary is empty, $\partial_{k-1} \circ \partial_k(c) = \emptyset$. This implies that the groups are nested, $\mathsf{B}_k \subseteq \mathsf{Z}_k \subseteq \mathsf{C}_k$, as illustrated in Figure 2. The boundary of

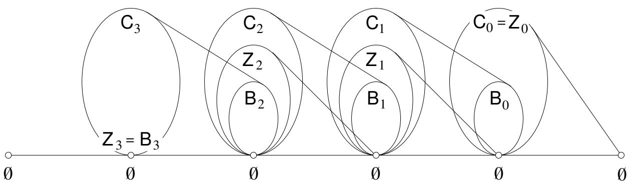

Figure 2. Chain, cycle, boundary groups and their images under the boundary operators.

a vertex is the empty set, which implies that every 0-chain is also a 0-cycle, $\mathsf{Z}_0 = \mathsf{C}_0$. Because K is a complex in $\mathbb{R}^3$, there are no non-empty 3-cycles or 3-boundaries, that is, $\mathsf{Z}_3 = \mathsf{B}_3 = \{\emptyset\}$.

Homology groups. The *k-th homology group* is the k-th cycle group factored by the k-th boundary group, $\mathsf{H}_k = \mathsf{Z}_k/\mathsf{B}_k$. Its elements are the *homology classes* $c + \mathsf{B}_k = \{c + b \mid b \in \mathsf{B}_k\}$, for all $c \in \mathsf{Z}_k$. The zero element is $\emptyset + \mathsf{B}_k = \mathsf{B}_k$, and the sum of two classes is $(c + \mathsf{B}_k) + (d + \mathsf{B}_k) = (c + d) + \mathsf{B}_k$. There is no torsion in complexes embedded in three-dimensional Euclidean space, so the homology groups are vector spaces. A subset *generates* a vector space if every element is the sum of elements in the subset. A *basis* is a minimal generating set. In general, there are no canonical bases, but all bases have the same size which is the *rank* of the group. Because taking symmetric differences is like adding modulo 2, the size of a group is 2 raised to the power of its rank. The *k-th Betti number* of K is the rank of the k-th homology group, $\beta_k = \operatorname{rank} \mathsf{H}_k$. As $\mathsf{H}_k = \mathsf{Z}_k/\mathsf{B}_k$,

$$\operatorname{rank} \mathsf{H}_k = \operatorname{rank} \mathsf{Z}_k - \operatorname{rank} \mathsf{B}_k. \tag{1}$$

There is a Betti number for each integer k, but for complexes in $\mathbb{R}^3$, only the ones for $0 \leq k \leq 2$ may be non-zero. In the absence of torsion, the Betti numbers under $\mathbb{Z}_2$ are the same as those under $\mathbb{Z}$, according to the Universal Coefficient Theorem for Homology [9]. We may still use $\mathbb{Z}_2$ coefficients in the presence of torsion, but we get Betti numbers that possibly differ from those computed using $\mathbb{Z}$ coefficients.

Intuitively, a non-bounding 0-cycle represents a collection of components of K and there is one basis element per component. It follows that β_0 is the number of components of K. A non-bounding 1-cycle represents a collection of non-contractible closed curves in K, or dually, a collection of tunnels formed by K. We can write each tunnel as a sum of tunnels in a basis, and β_1 is the size of the basis. A non-bounding 2-cycle represents a collection of non-contractible closed surfaces in K, or dually, a collection of voids, which are components of $\mathbb{R}^3 - K$. As before, β_2 is the size of a basis of voids, which is equal to the number of voids. Finally, there are no 3-cycles because K is a complex in $\mathbb{R}^3$.

Age filters. We base all formulas and algorithms in this paper on an ordering of the simplices, where each prefix of the ordering contains the simplices of a subcomplex. We call such an ordering a *filter*. The sequence of subcomplexes defined by taking successively larger prefixes is the corresponding *filtration*. Figure 3 illustrates these definitions with a filter of 18 simplices. We think of a filtration

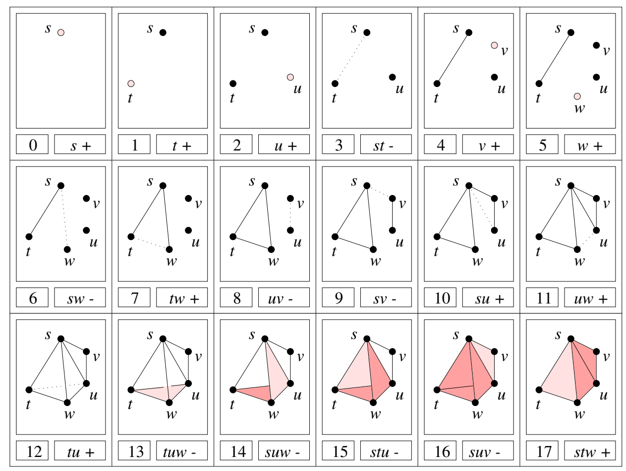

Figure 3. The filter is the sequence of light simplices. The corresponding filtration is the sequence of complexes.

as describing an evolution of a complex for which the sole element of change is growth. For dual complexes of a collection of balls, we generate a filter and a filtration by literally growing the balls. The filter is a consequence of this growth.

We now describe the growth model for spheres we use in Section 6 to apply our results to weighted point data in $\mathbb{R}^3$. For every real number $\alpha \in \mathbb{R}$, we increase the square radius of a ball $\hat{u}$ by α, giving us $\hat{u}(\alpha) = (u, U^2 + \alpha)$. We denote the collection of expanded balls $\hat{u}(\alpha)$ as $S(\alpha)$. The *α-complex* $K(\alpha)$ of S is the dual complex of $S(\alpha)$ [4]. For example, $K(-\infty) = \emptyset$, $K(0) = K$, and $K(\infty) = D$ is the dual of the Voronoi diagram, also known as the Delaunay triangulation of S. For each simplex $\sigma \in D$, there is a unique *birth time* $\alpha(\sigma)$ defined such that $\sigma \in K(\alpha)$ iff $\alpha \geq \alpha(\sigma)$. We order the simplices such that $\alpha(\sigma) < \alpha(\tau)$ implies σ precedes τ in the filter. More than one simplex may be born at a time and such cases may arise even if S is in general position. For example in Figure 1, edge uw is born at the same moment as triangle uvw. In the case of a tie, we order lower-dimensional simplices before higher-dimensional ones, breaking remaining ties arbitrarily. We call the resulting sequence the *age filter* of the Delaunay triangulation.

Incremental algorithm. The ordering of simplices in a filter permits a simple algorithm for computing Betti numbers of all complexes in a filtration [2]. We review the essential steps of the algorithm here. Suppose the sequence of σ^i, for $0 \leq i < m$, is a filter and the sequence of $K^i = \{\sigma^j \mid 0 \leq j \leq i\}$, for $0 \leq i < m$, is the corresponding filtration. Before running the algorithm, the Betti number variables are set to the Betti numbers of the empty complex, that is, $\beta_0 = \beta_1 = \beta_2 = 0$. The algorithm is shown in Figure 4.

```
integer³ BETTI-NUMBERS ()
  for i = 0 to m − 1 do
    k = dim σ^i − 1;
    if σ^i belongs to a (k + 1)-cycle in K^i
      then β_{k+1} = β_{k+1} + 1
      else β_k = β_k − 1
    endif
  endfor;
  return (β_0, β_1, β_2).
```

Figure 4. The function returns the Betti numbers of the last complex in the filtration.

But how do we decide whether a $(k+1)$-simplex σ^i belongs to a $(k+1)$-cycle in K^i? For $k+1 = 0$, this is trivial because every vertex belongs to a 0-cycle. For edges we maintain the connected components of the complex, each represented by its vertex set. An edge belongs to a 1-cycle iff its two endpoints belong to the same component. Triangles and tetrahedra are treated similarly, using the symmetry provided by complementarity, duality, and time-reversal [2].

Once we decide the cycle question for each simplex, we call a $(k+1)$-simplex σ^i *positive* if it belongs to a $(k+1)$-cycle and *negative* otherwise. Let $\beta_k = \beta_k^\ell$ be the k-th Betti number of K^ℓ, and let $\mathrm{pos}_k = \mathrm{pos}_k^\ell$ and $\mathrm{neg}_k = \mathrm{neg}_k^\ell$ be the number of positive and negative k-simplices in K^ℓ. The correctness of the incremental algorithm implies

$$\beta_k \quad = \quad \mathrm{pos}_k - \mathrm{neg}_{k+1}, \tag{2}$$

for $0 \leq k \leq 2$. In words, the Betti number β_k is the number of k-simplices that create k-cycles minus the number of $(k+1)$-simplices that destroy k-cycles by creating k-boundaries. Observe that Equation (2) is just a different way to write Equation (1). All Betti numbers are non-negative so $\mathrm{pos}_k \geq \mathrm{neg}_{k+1}$ for all ℓ. We will see in Section 3 that there exists a pairing between positive k-simplices and negative $(k+1)$-simplices. This pairing is the key to understanding the persistence of non-bounding cycles in homology groups.

3 Persistence

We wish to simplify a complex through the removal of its topological attributes. We describe a measure that ranks attributes by their life-time in a filtration — their persistence in being a feature in the face of growth. In this section we introduce the concept of persistence for Betti numbers and non-bounding cycles. We define persistence abstractly using cycle and boundary groups of complexes in a filtration. To make the abstract concrete, we give an algorithm that pairs the creation of a non-bounding cycle with its conversion to a boundary.

Algebraic formulation. Algebraically, it is easy to count the population of non-bounding cycles whose life-time exceeds a given threshold. We will see later that this statistic is sufficient for determining the life-time of individual non-bounding cycles. We define $\mathsf{Z}_k^\ell, \mathsf{B}_k^\ell$ to be the k-th cycle group and k-th boundary group, respectively, of the ℓ-th complex K^ℓ in a filtration. To capture persistent cycles in K^ℓ, we factor its k-th cycle group by the k-th boundary group of $K^{\ell+p}$, p complexes later in the filtration. Formally, the *p-persistent k-th homology group* of K^ℓ is

$$\mathsf{H}_k^{\ell,p} \quad = \quad \mathsf{Z}_k^\ell / (\mathsf{B}_k^{\ell+p} \cap \mathsf{Z}_k^\ell), \tag{3}$$

which is well-defined because $\mathsf{B}_k^{\ell+p} \cap \mathsf{Z}_k^\ell$ is the intersection of two subgroups of $\mathsf{C}_k^{\ell+p}$ and thus a group itself. The *p-persistent k-th Betti number* $\beta_k^{\ell,p}$ of K^ℓ is the rank of $\mathsf{H}_k^{\ell,p}$. Note that as we increase p, negative simplices cancel positive simplices earlier in the filtration. In other words, increasing p by one shortens the persistence of all non-bounding cycles by one. We may kill short-lived attributes, the topological noise of the complex, by increasing p sufficiently.

The p-persistent homology groups can also be defined using injective homomorphisms between ordinary homology groups. Observe that if two cycles are homologous in K^ℓ, they also exist and are homologous in $K^{\ell+p}$. Consider the homomorphism

$$\eta_k^{\ell,p} : \mathsf{H}_k^\ell \to \mathsf{H}_k^{\ell+p},$$

that maps a homology class into one that contains it. The image of the homomorphism is isomorphic to the p-persistent homology group of K^ℓ, $\mathrm{im}\, \eta_k^{\ell,p} \simeq \mathsf{H}_k^{\ell,p}$. Finally, we note that Robins has independently formulated an alternate but equivalent definition of persistence [10].

Abstract algorithm. To measure the life-time of a non-bounding cycle, we find when the cycle's homology class is created and when its class merges with the boundary group. A positive simplex creates the class and a negative one merges the class with the boundary group. To detect these events, we maintain a basis for H_k implicitly through simplex representatives.

Initially, the basis for H_k is empty. For each positive k-simplex σ^i, we first find a non-bounding k-cycle c^i that contains σ^i but no other positive k-simplices. We prove that c^i exists using induction as follows: start with an arbitrary k-cycle that contains σ^i and remove other positive k-simplices by adding their corresponding k-cycles. This method succeeds because each added cycle contains only one positive k-simplex by inductive assumption. After finding c^i, we add the homology class of c^i as a new element to the basis of H_k. In short, the class $c^i + B_k$ is represented by c^i, and c^i, in turn, is represented by σ^i. For each negative $(k+1)$-simplex σ^j, we find its corresponding positive k-simplex σ^i and remove the homology class of σ^i from the basis. A general homology class of K^i is a sum of basis classes,

$$\begin{aligned} d + \mathsf{B}_k &= \sum (c^g + \mathsf{B}_k) \\ &= \mathsf{B}_k + \sum c^g. \end{aligned}$$

The chains d and $\sum c^g$ are *homologous*, meaning they belong to the same homology class. Each c^g is represented by a positive k-simplex σ^g, $g < j$, that is not yet paired by the algorithm. The collection of positive k-simplices $\Gamma = \Gamma(d)$ is uniquely determined by d. The youngest simplex in Γ is the one with largest index and we denote this index as $y(d)$.

```
list³ PAIR-SIMPLICES ()
  L₀ = L₁ = L₂ = ∅;
  for j = 0 to m − 1 do
    k = dim σʲ − 1;
    if σʲ is negative then
(*)   d = ∂ₖ₊₁(σʲ);    i = y(d);
      Lₖ = Lₖ ∪ {(σⁱ, σʲ)}
    endif
  endfor;
  return (L₀, L₁, L₂).
```

Figure 5. The function returns three lists of paired simplices in the filter.

The algorithm, as shown in Figure 5, identifies σ^j as the culprit for turning the k-cycle created by σ^i into a k-boundary. We document this by appending (σ^i, σ^j) to the list L_k. The *persistence* of that k-cycle is one short of the difference between indices, $j - i - 1$.

Time-based persistence. Alternatively, we could define persistence as the difference in birth times of the two simplices, $\alpha(\sigma_j) - \alpha(\sigma_i)$. This view corresponds to an extension of our previous index-based formulation. Let $K^\alpha = \{\sigma^j \mid \alpha(\sigma^j) \leq \alpha\}$. Then for every real $\pi \geq 0$, we define the *π-persistent k-th homology group* of K^α to be

$$\mathsf{H}_k^{\alpha,\pi} = \mathsf{Z}_k^\alpha / (\mathsf{B}_k^{\alpha+\pi} \cap \mathsf{Z}_k^\alpha). \tag{4}$$

Note that we are not changing the ordering of the simplices, so the simplex pairs do not change. Therefore, we may use the abstract algorithm above to compute the persistence pairs. In general, however, we have fewer pair-wise different complexes, as all simplices with birth-time α enter the filtration at that time. In the index-based formulation, the simplices arrive individually with different indices.

Time-based persistence is useful in the context of isosurfaces of density functions. Index-based persistence is appropriate for alpha complexes, as most interesting activity occurs in a small range of α. We will not discuss time-based persistence in this paper any further, although all our results will be valid with little modification.

Visualization. Our pairing algorithm gives us a set of simplex pairs (σ^i, σ^j), each representing a k-cycle for $0 \leq k \leq 2$. We may visualize each pair on the index axis by a half-open interval $[i, j)$ which we call a *k-interval*. We show this in Figure 6 for the filtration in Figure 3. The incremental algorithm in Section 2 asserts that β_k^ℓ is the number of k-intervals that contain index ℓ on the axis.

We extend this visualization to two dimensions spanned by the index and persistence axes. The k-interval of (σ^i, σ^j) is extended into a *k-triangle* spanned by $(i, 0)$, $(j, 0)$, $(i, j - i)$ in the index-persistence plane. The k-triangle is closed along its vertical and horizontal edges and open along the diagonal connecting $(j, 0)$ to $(i, j - i)$, as shown in Figure 6. It represents the k-cycle created by σ^i, which is destroyed by σ^j progressively earlier as we increase p.

Assume for a moment that our algorithm is correct, which means $\beta_k^{\ell,p}$ is the number of k-triangles that contain point (ℓ, p). Then the persistent Betti numbers are non-increasing along vertical lines in the index-persistence plane. The same is true for lines in the diagonal direction and for all lines between the vertical and the diagonal directions.

MONOTONICITY LEMMA. $\beta_k^{\ell,p} \leq \beta_k^{\ell',p'}$ whenever $p' \leq p$ and $\ell \leq \ell' \leq \ell + (p - p')$.

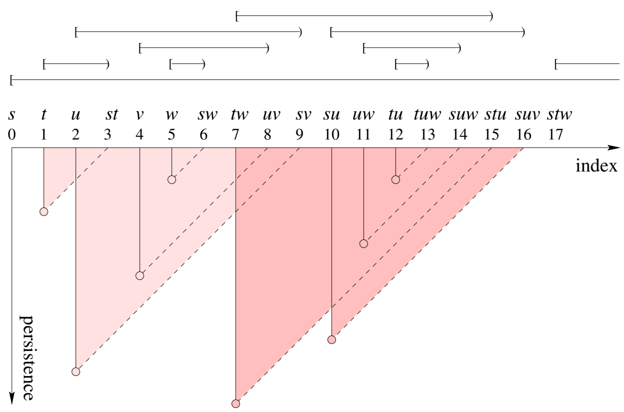

Figure 6. Visualization of the result of Function PAIR-SIMPLICES. The triangles of s and of stw are unbounded and not drawn. The light and dark triangles represent 0-cycles and 1-cycles respectively.

.

Correctness. To prove the abstract algorithm is correct, we show that the pairs it produces are consistent with the persistent Betti numbers defined by (3).

k-TRIANGLE LEMMA. The number of k-triangles containing (ℓ, p) in the index-persistence plane is $\beta_k^{\ell,p}$.

PROOF. We proceed by induction over p. For $p = 0$, the number of k-triangles that contain $(\ell, 0)$ is equal to the number of k-intervals $[i, j)$ that contain ℓ. This is equal to the number of left endpoints minus the number of right endpoints that are smaller than or equal to ℓ. Equivalently, it is the number of positive k-simplices σ^i with $i \leq \ell$ minus the number of negative $(k+1)$-simplices σ^j with $j \leq \ell$. But this is just a restatement of Equation (2), which establishes the basis of the induction.

Consider (ℓ, p) with $p > 0$ and assume inductively that the claim holds for $(\ell, p-1)$. The relevant simplex for the step from $(\ell, p-1)$ to (ℓ, p) is $\sigma^{\ell+p}$. The persistent k-th Betti number can either stay the same or decrease by 1. It will decrease only if $\sigma^{\ell+p}$ is a negative $(k+1)$-simplex, or equivalently, $(\ell+p, 0)$ is the upper right corner of a k-triangle. Indeed, no other k-triangle can possibly separate $(\ell, p-1)$ and (ℓ, p). This proves the claim if $\sigma^{\ell+p}$ is a positive $(k+1)$-simplex or a simplex of dimension different from $k+1$. Now suppose that $\sigma^{\ell+p}$ is a negative $(k+1)$-simplex and define the k-cycle $d = \partial_{k+1}(\sigma^{\ell+p})$. There are two cases, as shown in Figure 7.

`Case 1.` Assume there is a k-cycle c in K^ℓ homologous to d, that is, $c \in d + \mathsf{B}_k^{\ell+p-1}$. Then c bounds neither in K^ℓ nor in $K^{\ell+p-1}$, but it bounds in $K^{\ell+p}$. If follows that $\beta_k^{\ell,p} = \beta_k^{\ell,p-1} - 1$. We need to show that the pair $(\sigma^i, \sigma^{\ell+p})$ constructed by the algorithm satisfies $i \leq \ell$, because only in this case does the k-triangle of $\sigma^{\ell+p}$ separate $(\ell, p-1)$ from (ℓ, p). Recall that σ^i is the youngest positive k-simplex in $\Gamma(d)$. To reach a contradiction suppose $i > \ell$. Then c is a non-bounding k-cycle also in K^i, and because it is homologous to d, we have $\sigma^i \in c$. But this contradicts $c \subseteq K^\ell$ as $\sigma^i \notin K^\ell$.

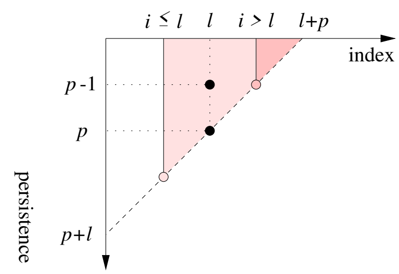

Figure 7. The light k-triangle corresponds to `Case 1` and the dark one to `Case 2`.

`Case 2.` Assume there is no k-cycle in K^ℓ homologous to d. Then $\mathsf{Z}_k^\ell \cap \mathsf{B}_k^{\ell+p-1} = \mathsf{Z}_k^\ell \cap \mathsf{B}_k^{\ell+p}$, and hence $\beta_k^{\ell,p} = \beta_k^{\ell,p-1}$. We need to show that the pair $(\sigma^i, \sigma^{\ell+p})$ constructed by the algorithm satisfies $i > \ell$, because only in this case does the k-triangle of $\sigma^{\ell+p}$ not separate $(\ell, p-1)$ from (ℓ, p). Our assumption above implies that at least one of the positive k-simplices in $\Gamma(d)$ was added after σ^ℓ. Hence $i = y(d) > \ell$. ◻

4 Computation

In this section, we complete the abstract algorithm for pairing by specifying how to implement line (*) of Function PAIR-SIMPLICES. We do this by computing the index i of the youngest positive k-simplex in $\Gamma(d)$, where $d = \partial_{k+1}(\sigma^j)$. We refer to this computation as *cycle search* for σ^j. We will first describe the data structure, then explain cycle search, prove its correctness, and analyze its running time.

Data structure. We use a linear array $\mathtt{T}[0..m-1]$, which acts similar to a hash table [1, Chapter 12]. Initially, `T` is empty. A pair (σ^i, σ^j) identified by the algorithm is stored in $\mathtt{T}[i]$ together with a list of positive simplices Λ^i defining the cycle created by σ^i and destroyed by σ^j. The simplices in that list are not necessarily the same as the ones in $\Gamma(d)$. All we guarantee is that d is homologous to the sum of cycles represented by the simplices in the list, and that the list contains the youngest simplex in $\Gamma(d)$, which is σ^i as above. The correctness proof following the algorithm will show that this property is sufficient for our purposes. The data structure is illustrated in Figure 8. Each simplex in the

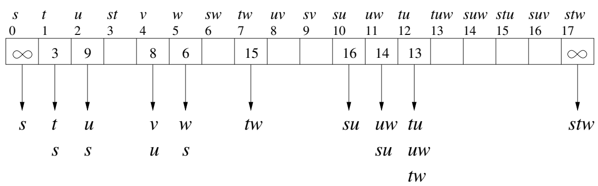

Figure 8. Hash table after running the algorithm on the filter of Figure 3.

filter has a slot in the hash table, but information is stored only in the slots of the positive simplices. This information consists of the index j of the matching negative simplex and a list of positive simplices defining a cycle. Some cycles exists beyond the end of the filter, in which case we use ∞ as a substitute for j.

Cycle search. Suppose the algorithm arrives at index j in the filter and assume σ^j is a negative $(k+1)$-simplex. Recall that $\Gamma(d)$ is the set of positive k-simplices that represent the homology class of $d = \partial\sigma^j$ in H_k^{j-1}. We search for the youngest k-simplex in $\Gamma(d)$ by successively probing slots in $\mathtt{T}$ until we find the right one. Specifically, we start with a set Λ equal to the set of positive k-simplices in d, which is necessarily non-empty, and we let $i = \max(\Lambda)$ be the index of the youngest member of Λ. We will see later that if $\mathtt{T}[i]$ is unoccupied, then $i = y(d)$. We can therefore end the search and store j and Λ in $\mathtt{T}[i]$. If $\mathtt{T}[i]$ is occupied, it contains a collection Λ^i representing a permanently stored k-cycle. At this moment, this k-cycle is already a k-boundary. We add Λ and Λ^i to get a new Λ representing a k-cycle homologous to the old one and therefore also homologous to d. The Function YOUNGEST in Figure 9 performs a cycle search for simplex σ^j.

```
integer YOUNGEST (simplex σ^j)
  Λ = {σ ∈ ∂_{k+1}(σ^j) | σ positive};
  loop
    i = max(Λ);
    if T[i] is unoccupied then
      store j and Λ in T[i]; exit
    endif;
    Λ = Λ + Λ^i
  forever;
  return i.
```

Figure 9. The function returns the index of the youngest basis cycle used in the description of the boundary of σ^j.

A *collision* is the event of probing an occupied slot of $\mathtt{T}$. It triggers the addition of Λ and Λ^i, which means we take the symmetric difference of the two collections. For example, the first collision for the filter of Figure 3 occurs for the negative edge sv. Initially, we have $\Lambda = \{s, v\}$ and i equal to 4, the index of v. $\mathtt{T}[4]$ is occupied and stores $\Lambda^4 = \{u, v\}$. The sum of the two 0-cycles is $\Lambda + \Lambda^4 = \{s, u\}$, which is the new set Λ. We now have $i = 2$, the index of u. This time, $\mathtt{T}[2]$ is unoccupied and we store the index of sv and the new set Λ in that slot.

Correctness. We first show that cycle search halts. Consider a collision at $\mathtt{T}[i]$. The list Λ^i stored in $\mathtt{T}[i]$ contains σ^i and possibly other positive k-simplices, all older than σ^i. After adding Λ and Λ^i we get a new list Λ. This list is necessarily non-empty, as otherwise d would bound. Furthermore, all simplices in Λ are strictly older than σ^i. Therefore, the new i is smaller than the old one, which implies that the search proceeds strictly from right to left in $\mathtt{T}$. It necessarily ends at an unoccupied slot $\mathtt{T}[g]$ of the hash table, for all other possibilities lead to contradictions.

It takes some more effort to prove that $\mathtt{T}[g]$ is the correct slot, or in other words, that $g = y(d)$, where $d = \partial_{k+1}(\sigma^j)$ is the boundary of the negative $(k+1)$-simplex that triggered the search. Let e be the cycle defined by Λ^g. Since e is obtained from d through adding bounding cycles, we know that e and d are homologous in K^{j-1}. A *collision-free* cycle is one where the youngest positive simplex corresponds to an unoccupied slot in the hash table. Cycle search ends whenever it reaches a collision-free cycle. For example, e is collision-free because its youngest positive simplex is σ^g and $\mathtt{T}[g]$ is unoccupied before e arrives.

COLLISION LEMMA. Let e be a collision-free k-cycle in K^{j-1} homologous to d. Then the index of the youngest positive simplex in e is $i = y(d)$.

PROOF. Let σ^g be the youngest positive simplex in e, and f be the sum of the basis cycles, homologous to d. By definition, f's youngest positive simplex is σ^i, where $i = y(d)$. This implies that there are no cycles homologous to d in K^{i-1} or earlier complexes, therefore $g \geq i$. We show $g \leq i$ by contradiction. If $g > i$, then $e = f + c$, where c bounds in K^{j-1}. $\sigma^g \notin f$ implies $\sigma^g \in c$, and as σ^g is the youngest in e, it is also the youngest in c. By assumption, $\mathtt{T}[g]$ is unoccupied as e is collision-free. In other words, the cycle created by σ^g is still a non-bounding cycle in K^{j-1}. Hence this cycle cannot be c. Also, the cycle cannot belong to c's homology class at the time c becomes a boundary. It follows that the negative $(k+1)$-simplex that converts c into a boundary pairs with a positive k-simplex in c that is younger than σ^g, a contradiction. Hence $g = i$. ⊔⊓

The cycle search continues until it finds a collision-free cycle homologous to d, and the Collision Lemma implies that that cycle has the correct youngest positive simplex. This proves the correctness of cycle search, and we may now substitute $i = \text{YOUNGEST}(\sigma^j)$ for line (*) in Function PAIR-SIMPLICES.

Running time. Let $d = \partial_{k+1}(\sigma^j)$ and let σ^i be the youngest positive k-simplex in $\Gamma(d)$. The persistence of the cycle created by σ^i and destroyed by σ^j is $p_i = j - i - 1$. The search for σ^i proceeds from right to left starting at $\mathtt{T}[j]$ and ending at $\mathtt{T}[i]$. The number of collisions is at most the number of positive k-simplices strictly between σ^i and σ^j, which is less than p_i. A collision happens at $\mathtt{T}[g]$ only if σ^g already forms a pair, which implies its k-interval $[g, h)$ is contained inside $[i, j)$. We use the nesting property to prove by induction that the k-cycle defined by Λ^i is the sum of fewer than p_i boundaries of $(k+1)$-simplices. Hence, Λ^i contains fewer than $(k+2)p_i$ k-simplices, and similarly Λ^g contains fewer than $(k+2)p_g < (k+2)p_i$ k-simplices. A collision requires adding the two lists and finding the youngest in the new list. We do this by merging, which keeps the lists sorted by age. A single collision takes time at most $\mathrm{O}(p_i)$, and the entire search for σ^i takes time at most $\mathrm{O}(p_i^2)$. The total algorithm runs in time at most $\mathrm{O}(\sum p_i^2)$, which is at most $\mathrm{O}(m^3)$.

The running time of cycle search can be improved to almost constant for dimensions $k = 0$ and $k = 2$ using a union-find data structure representing a system of disjoint sets and supporting find and union operations [1, Chapter 22]. For $k = 0$, each set is the vertex set of a connected component. Each set has exactly one yet unpaired vertex, namely the oldest one in the component. We modify standard union-find implementations in such a way that this vertex represents the set. Given a vertex, the find operation returns the representative of the set that contains this vertex. Given an edge whose endpoints lie in different sets, the union operation merges the two sets into one. At the same time, it pairs the edge with the younger of the two representatives and retains the older one as the representative of the merged set.

Cycle search is replaced by two find operations possibly followed by a union operation. If we use weighted merging for union and path compression for find, the amortized time per operation is $\mathrm{O}(A^{-1}(m))$, where $A^{-1}(m)$ is the notoriously slowly growing inverse of the Ackermann function [1, Chapter 22]. We may use symmetry to accelerate cycle search for 2-cycles using the union-find data structure for a system of sets of tetrahedra [2]. We cannot achieve the same acceleration for 1-cycles using this method, however, as there can be multiple unpaired positive edges at any time. The additional complication seems to require the more cautious and therefore slower algorithm described above.

5 Simplification

In this section, we use information about the persistence of cycles to simplify filtrations. Simplification here means reordering the simplices in such a way that only cycles whose persistence is above some threshold appear in the filtration. We get inspiration for reordering through an algorithm for computing persistent Betti numbers.

Computing persistent Betti numbers. By the k-Triangle Lemma in Section 3, the p-persistent k-th Betti number of K^ℓ is the number $\beta_k^{\ell,p}$ of k-triangles that contain the point (ℓ, p) in the index-persistence plane. To compute these

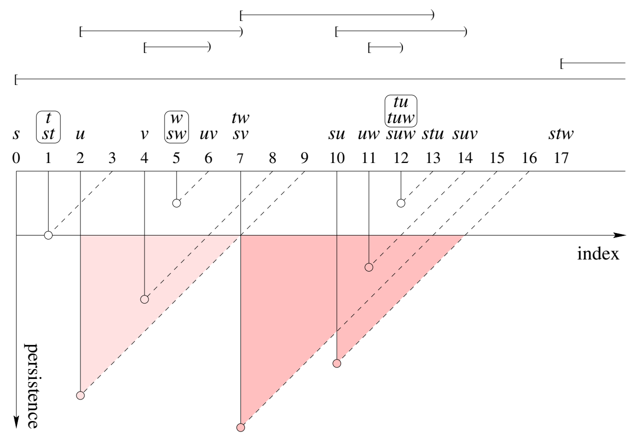

Figure 10. The k-triangles that intersect the new axis at $p = 2$ have persistence 2 or larger. The simplex pairs representing cycles of persistence less than 2 are boxed.

numbers for a fixed p, we intersect the k-triangles with a horizontal line at p. Figure 10 illustrates this operation by modifying Figure 6. The algorithm for p-persistent Betti numbers is similar to Function BETTI-NUMBERS given in Figure 4. We go through the filter from left to right and increase β_k^p whenever we encounter the left endpoint of a k-interval longer than p. Similarly, we decrease β_k^p whenever p positions ahead of us there is a right endpoint of a k-interval longer than p. Figure 11 shows the results of the algorithm applied to our example filtration of Figure 3 for $k = 0$.

Migration. The intersection of the k-triangles and the horizontal line at p is a collection of half-open intervals. We interpret these intervals as k-intervals of a simplified version of the original filtration. Our goal is to reorder the filter so that this interpretation is valid, that is, we wish to obtain a new filtration whose Betti numbers are the p-persistent Betti numbers of the original filtration. For each pair (σ^i, σ^j) we move σ^j to the left, closer or all the way to σ^i, as shown

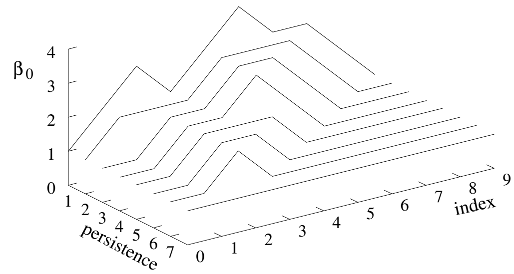

Figure 11. Persistent 0-th Betti numbers of the first ten complexes in the filtration of Figure 3 and for persistence up to 7.

in Figure 10. The new position of σ^j is $\max\{i, j-p\}$. If $j - p \leq i$, then σ^i and σ^j no longer form an interval as they both occupy the same index in the new filter. Here, we extend the notion of a filter to allow a possibly empty set of simplices at each index. We compute Betti numbers by bringing all simplices of a set into the complex at once.

There is a complication in the reordering algorithm that occurs whenever a negative simplex attempts to move past one of its faces. To maintain the ordering as a filter, we must move the face along with its coface. For example, if we increase p to 4 in Figure 10, then stu will move to index 11 past its face tu at index 12. Moving a face along with a simplex will not change any Betti numbers if the face represents a cycle whose persistence is less than p. At the time we move it, the face is already co-located with its matching negative simplex, and the two cancel each other's contributions. We may then grab the pair and move it with the simplex, moving the pair (tu, tuw) with stu in our example. For any moving simplex, however, we must also move all the necessary faces and their matching negative simplices recursively.

Conflicts. There is trouble if the face of a moving negative simplex represents a cycle whose persistence is at least p. For instance, when stu encounters the edge su, the triangle suv which is paired with su has not yet reached su. There is a conflict between our two goals of maintaining the filter property and reordering so the new Betti numbers are the old p-persistent Betti numbers. Formally, a *conflict* occurs whenever there are pairs (σ^i, σ^j) and (σ^g, σ^h) with $g < i < h < j$, where σ^i is a face of σ^h, as shown in Figure 12. There are $\binom{4}{2} = 6$ possible types of conflicts, each

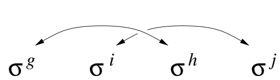

Figure 12. Basic conflict configuration.

identified by the pair $(\dim \sigma^i, \dim \sigma^h)$ of the dimensions of the main participants. The pairs (su, suv) and (tw, stu) in Figure 6 constitute a conflict of type $(1, 2)$ and show that conflicts do occur, although our experiments in Section 6 suggest that they are rather rare. We partially substantiate this finding by the following lemma.

CONFLICT LEMMA. All conflicts have type $(1, 2)$.

PROOF. Suppose a conflict exists in pairs (σ^i, σ^j) and (σ^g, σ^h), where σ^i is a vertex. When σ^h enters the filtration, it belongs to the same component as σ^g, since σ^h completes a chain whose boundary includes σ^g. Vertex σ^i, one of the vertices of σ^h, is unpaired and therefore represents the component of σ^h and σ^g. Recall that any component is represented by its oldest vertex, which implies that σ^i is older than all the vertices of σ^g. By the filter property, σ^i is older than σ^g, i.e. $i < g$, which contradicts the assumption that (σ^i, σ^j) and (σ^g, σ^h) form a conflict. This proves there are no conflicts of types $(0, 1)$, $(0, 2)$, $(0, 3)$. By complementarity and duality, there are no conflicts of types $(1, 3)$ and $(2, 3)$. ⊔⊓

Difficulties in reordering may also arise indirectly because of the recursive nature of any reordering algorithm. For example, moving a negative triangle may require moving one of its edges. This edge holds on to its matching triangle, which in turn grabs its needed faces. Some of these faces may be unpaired, and to capture this situation we define a *recursive conflict* to be a positive simplex that is moved when it is not co-located with its matching negative simplex. Extending the Conflict Lemma, we can show that all recursive conflicts are edges. We again have a situation as in Figure 12, except that σ^i is not necessarily a face of σ^h. However, the moving simplices all belong to the same component as σ^h: this is true for a face by definition, for a matching negative simplex by the reason given in the proof above, and for all moving simplices by transitivity.

Basic and recursive conflicts exist in practice, but are rather rare, as shown in Section 6. When conflicts occur, we view the maintenance of the filter property as inviolable, and attempt to approximate our secondary goal, achieving the correct Betti numbers. We do so through conflict resolution or conflict diminution, as described below.

Conflict resolution. We may resolve a conflict by subdivision. Here, we achieve the correct Betti numbers for a refined complex and its corresponding filtrations. Suppose pairs (σ^i, σ^j) and (σ^g, σ^h) form a conflict. Then, σ^i, σ^g are edges, σ^j, σ^h are triangles, and σ^i is a face of σ^h. Let $\sigma^i = bc$ and $\sigma^h = abc$ as drawn in Figure 13.

We resolve the conflict by starring from the midpoint x of edge bc, subdividing all simplices that share bc as a common face. We replace each subdivided k-simplex by one

$(k-1)$-simplex and two k-simplices. For computing persistence, the order of the three new simplices is important. As shown in Figure 13, the order of the edges bx, cx within the new filter is the opposite of the triangles acx, abx. The persistence algorithm produces new pairs (x, bx) and (ax, acx) that have no effect on Betti numbers. After acx enters, the complex is homotopy equivalent to the old complex just before abc enters. The edge cx replaces bc and the triangle abx replaces abc in the filter. Consequently, the algorithm produces pairs (σ^g, abx) and (cx, σ^h). As cx is not a face of abx, we have removed the conflict and preserved the Betti numbers of a refined filtration.

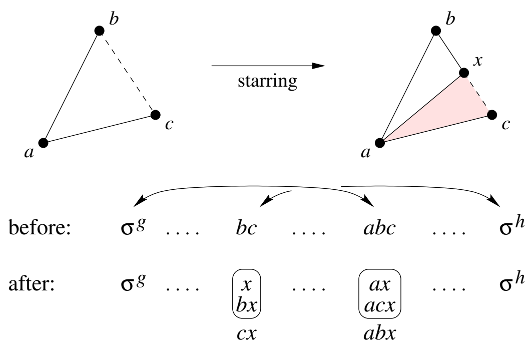

Figure 13. The conflict exists between moving abc towards σ^g and keeping bc ahead of abc. We subdivide edge bc and order the new simplices to resolve the conflict.

Conflict diminution. Often times, simplices have structural meaning in a filtration, and conflicts signal properties of the structure the simplices describe. We may not wish to tamper with this structure through subdivision, as such action may not have any meaning within our filtration. For example, in alpha complex filtrations, simplices are ordered according to a particular growth model. The ordering of the new simplices specified by subdivision in Figure 13 might not have a corresponding set of weighted balls that would generate the filtration under the growth model.

We may attempt to reduce the effect of conflicts on Betti numbers without eliminating the conflicts. Recall that a simplex pair (σ^i, σ^j) defines a k-cycle which may be visualized by a k-triangle, as in Figure 14. Whenever σ_i occurs in a conflict, we allow it to be dragged to a new location. This clearly changes the Betti numbers of the reordered filtration, so they no longer match the p-persistent Betti numbers of the original filtration. We also allow σ_j to move faster during reordering, whenever σ_i is moved. This method creates a region of the k-cycle with the same area as the k-triangle, as shown in Figure 14. Therefore, we allow each k-cycle to have the same effect on Betti numbers as it would in the absence of conflicts, but at different times.

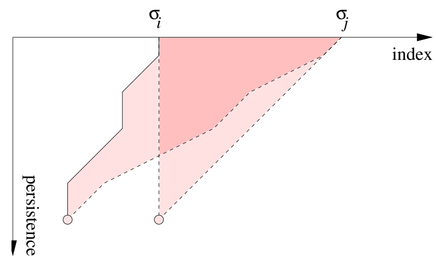

Figure 14. Reordering with conflicts.

Lazy migration. We end this section by describing an alternate method for reordering. Our motivation for formulating persistent homology in Equation (3) was to eliminate cycles with low persistence. As a consequence of the formulation, the life-time of *every* cycle is reduced regardless of its persistence, leading to the creation of k-triangles. A possibly more intuitive goal would be to eliminate cycles with low persistence without changing the life-time of cycles with high persistence. In other words, we replace k-triangles by *k-squares* as illustrated in Figure 15. In analogy

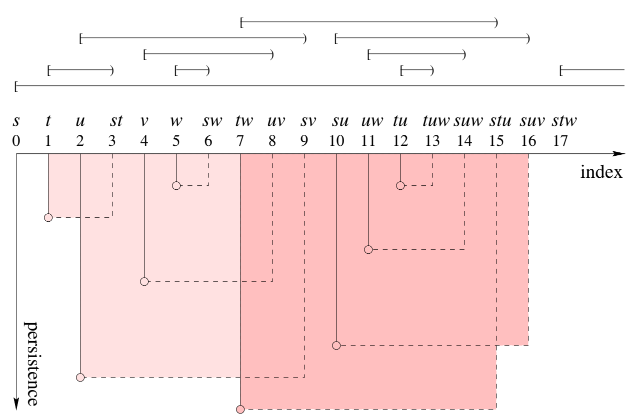

Figure 15. Alternative visualization of the result of Function PAIR-SIMPLICES. The squares of s and stw are unbounded and not shown. The light squares represent 0-cycles and the dark squares represent 1-cycles.

to p-persistent Betti numbers, we define $\gamma_k^{\ell,p}$ as the number of k-squares that contain the point (ℓ, p) in the index-persistence plane. Figure 16 illustrates how these numbers change as we increase persistence from $p = 0$ to 7. Note that we can easily read off persistent cycles from the graph. We may also simplify complexes using $\gamma_k^{\ell,p}$ by only collapsing k-intervals of length at most p, leaving other k-intervals unchanged.

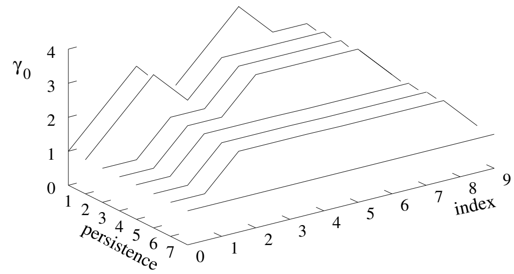

Figure 16. Numbers γ_0 for the first ten complexes in the filtration of Figure 3.

6 Experiments

We have implemented the algorithms described in this paper and created a prototype based on the Alpha Shapes software of [4]. This section presents experimental timing results and provides evidence for the claim that persistence separates topological noise from features.

Data. We have applied the software to a variety of datasets and show the results for five representative sets in this section. Three sets represent molecular structures with weighted points and two represent surfaces of macroscopic shapes with unweighted points. In each case, we first compute the possibly weighted Delaunay triangulation and then the age filter of that triangulation. The data points become vertices or 0-simplices of the triangulation. Table 1 gives the sizes of the data sets, their Delaunay triangulations, and age filters.

	# k-simplices				total
	0	1	2	3	
G	318	2,322	3,978	1,973	8,591
Z	1,296	11,401	20,098	9,992	42,787
D	7,774	60,675	105,710	52,808	226,967
B	42,311	346,664	608,445	304,091	1,301,511
S	54,262	438,134	766,893	383,020	1,642,309

Table 1. G is Gramicidin A, a small protein. Z is a portion of a periodic zeolite structure. D is a portion of DNA. B is a tiny cube of microscopic bone structure. S is a scanned and resampled Buddha statue.

Timings. We time only the portion of the software that is directly related to computing persistence. We distinguish four steps: marking simplices as positive or negative, and adding k-cycles for $k = 0, 1, 2$. Recall that the computation of persistence can be accelerated for $k = 0, 2$ using a union-find data structure. As substantiated in Table 2, this improvement subsumes adding 0- and 2-cycles in the marking process, shrinking the time for these to steps to essentially nothing. The results suggest a possible linear dependence of the running time on the size of the data, which is substantially faster than the cubic dependence proved in Section 4. Of course, we need to distinguish worst-case from average running time. After accelerating with union-find, the slowest portion of the algorithm is adding 1-cycles. We pose the detailed analysis of the algorithm as an open problem, and we also ask for a different and more efficient algorithm, if it exists.

	mark	add k-cycles			total	
		0	1	2	w/o UF	w UF
G	0.03	0.01	0.03	0.01	0.08	0.06
Z	0.19	0.02	0.17	0.07	0.45	0.36
D	1.15	0.12	1.02	0.38	2.67	2.18
B	7.35	0.73	6.79	2.58	17.45	14.12
S	11.25	0.96	9.65	8.66	30.52	19.93

Table 2. Running time in seconds. All timings were done on a Micron PC with a 266 MHz Pentium II processor and 128 MB random access memory, running the Solaris 8 operating system.

Statistics. Our cubic upper bound in Section 4 followed from the observation that the k-cycle created by σ^i goes through fewer than p_i collisions and the length of its list built up during these collisions is less than $(k + 2)p_i$. We can explain the apparently linear running time documented in Table 2 by showing that the average number of collisions and the average list length are both constant. Tables 3 and 4 provide evidence that this might indeed be the case. We do not show statistics for 0-cycles in Table 4 as every 0-cycle is represented by a list of length two.

	0-cycles		1-cycles		2-cycles	
	max	avg	max	avg	max	avg
G	17	0.58	31	0.19	91	0.13
Z	14	0.95	32	0.50	39	0.39
D	14	0.62	203	0.32	26	0.19
B	21	1.00	581	0.26	1,462	0.15
S	24	1.00	1,095	0.19	33,325	0.14

Table 3. Maximum and average number of collisions for the five data sets.

Recall that the number of collisions and the length of lists is bounded from above by the persistence of cycles. Table 5 shows that the average persistence is considerably larger than the average number of collisions and list length. Table 5 also gives evidence that conflicts are indeed rare.

Feature detection. We conclude this section by using persistence for detecting features of the dataset G. This dataset

	1-cycles			2-cycles		
	max	avg	avgf	max	avg	avgf
G	10	2.47	2.22	111	4.86	2.08
Z	46	3.91	2.52	46	2.96	2.05
D	125	3.85	2.21	15	2.26	2.02
B	1,719	9.46	2.50	510	8.15	2.07
S	4,993	19.61	2.47	7,486	374.06	2.04

Table 4. Maximum and average length of cycle lists, over all lists (avg), and all final stored lists (avgf).

	average persistence			# conflicts	
	0-cycles	1-cycles	2-cycles	basic	recursive
G	320.36	55.35	4.47	0	0
Z	1,333.86	757.03	194.56	1	128
D	7,902.83	2,706.18	462.17	0	212
B	59,383.77	5,264.05	393.92	1	187
S	95,620.31	3,828.92	45.22	0	10

Table 5. Average persistence of cycles and number of conflicts in simplifying the filtration.

contains the time-averaged molecular dynamics structure of Gramicidin A, a peptide that forms a channel for ion and water movement across lipid membranes. The primary topological feature of this data is a tunnel that runs through the molecule, as shown in Figure 17. We show the graphs

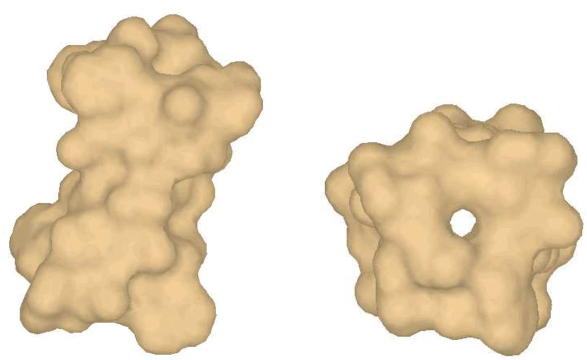

Figure 17. Side and top views of the molecular surface for Gramicidin A, presenting the channel for ion transport.

in Figure 18 for comparison. While there is considerable topological noise at $p = 0$, a simplification process which eliminates 1-cycles of persistence less than 2,688 succeeds in separating the tunnel from the remaining topological attributes detected by measuring homology. We show this simplification for three complexes in Figure 19.

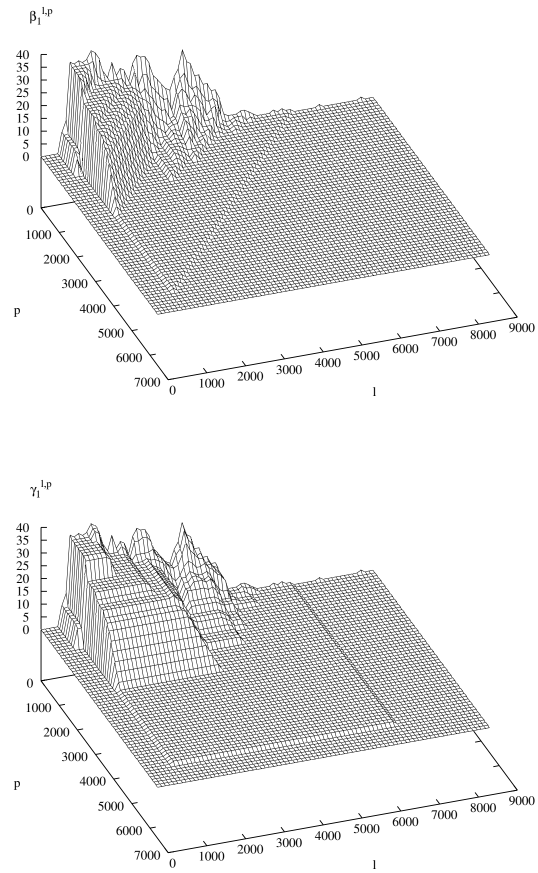

Figure 18. Graphs of $\beta_1^{\ell,p}$ and $\gamma_1^{\ell,p}$ of Gramicidin A sampled onto an 80 by 80 grid.

7 Future Directions

We introduce the notion of topological persistence for a filtration in $\mathbb{R}^3$ in this paper and give algorithms for assessing persistence and simplifying the filtration. We plan to apply these ideas to the analysis of three types of datasets.

(i) Molecular data are naturally represented by the age filters of their Delaunay triangulations. While differentiating topological noise from features is a generally useful facility, we believe it is most significant for analyzing the structure of molecules.

(ii) Simplifying shape while preserving features is at the core of the surface reconstruction problem studied in computer graphics, as well as computational geometry. We believe persistence can help in automatic reconstruction.

(iii) We plan to apply persistence to detect hierarchical or other complex types of clustering in very large

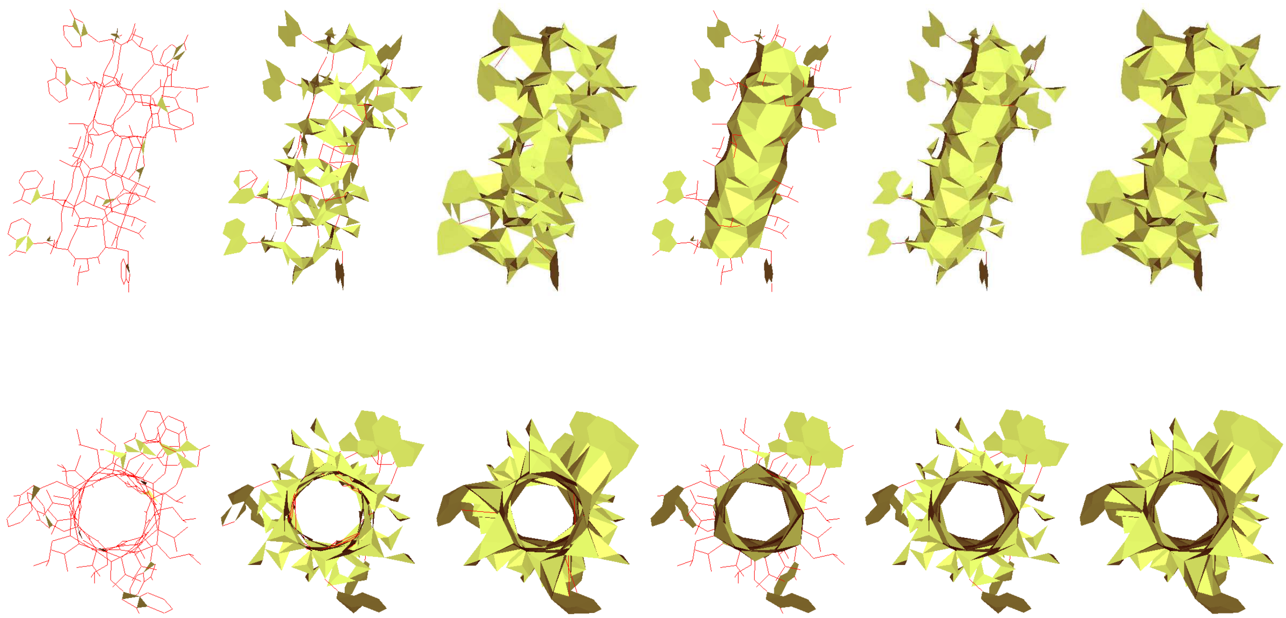

Figure 19. Side and top views of complexes $K_{715}, K_{1431}, K_{2682}$ of Gramicidin A are shown in the left three columns. The corresponding 2688-persistent complexes are shown on the right.

datasets, such as those expected from the current efforts of measuring the locations of galaxies in the universe.

There is a fourth and less direct application to surface reconstruction through iso-surface extraction. Iso-surfaces are widely used in medical imaging where volume density data is common. A smooth density function has finitely many critical points of four types, corresponding to the four different dimensions of simplices in a 3-dimensional complex. We can use the persistence algorithm to measure the significance of every critical point. A more difficult task is using this information for automatic denoising the density function and its iso-surfaces.

Acknowledgments

We thank Jeff Erickson and John Harer for helpful discussions during early stages of this paper. We also thank Daniel Huson for the zeolite dataset Z, Thomas LaBean for the DNA dataset D, and the Stanford Graphics Lab for the Buddha dataset S. To generate the bone dataset B, we sampled an iso-surface generated by Dominique Attali. The volume data was provided by Françoise Peyrin from CNRS CREATIS in Lyon and was issued from Synchrotron Radiation Microtomography from the ID19 beamline at ESRF in Grenoble. We generated Figure 17 using the Protein Explorer [6].

References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1994.

[2] C. J. A. Delfinado and H. Edelsbrunner. An incremental algorithm for betti numbers of simplicial complexes on the 3-sphere. *Comput. Aided Geom. Design*, 12:771–784, 1995.

[3] H. Edelsbrunner and E. P. Mücke. Simulation of Simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graphics*, 9:66–104, 1990.

[4] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graphics*, 13:43–72, 1994.

[5] J. El-Sana and A. Varshney. Topology simplification for polygonal virtual environments. *IEEE Trans. Visualization Comput. Graphics*, 4:133–144, 1998.

[6] E. Martz. Protein explorer 1.80b. http://www.umass.edu-/microbio/chime/explorer.

[7] J. McCleary. *User's Guide to Spectral Sequences*. Publish or Perish, Wilmington, Delaware, 1985.

[8] J. Milnor. *Morse Theory*. Princeton Univ. Press, New Jersey, 1963.

[9] J. R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, Redwood City, California, 1984.

[10] V. Robins. *Computing topology at multiple resolutions: foundations and applications to fractals and dynamics*. PhD

thesis, Dept. Appl. Math., University of Colorado, June 2000.
[11] J. A. Sethian. *Level Set Methods*. Cambridge Univ. Press, Cambridge, England, 1996.