Persistence-guided Prescribed Topological Simplification

LINXUAN RONG, Washington University in St. Louis, USA
TAO JU, Washington University in St. Louis, USA

We present a new method for simplifying the topology of a 3D shape. Unlike existing methods that either remove all topological features or offer indirect control over the target topology, our method aims at exactly preserving the user-prescribed numbers of topological features of each type (e.g., components, handles, and voids), while making minimal geometric changes. Guided by *persistent homology*, our method removes features with low persistence by performing either cutting or filling. This is achieved by an algorithm for computing candidate cuts and fills that remove only low-persistence features, an efficient algorithm for selecting an optimal subset of candidates by computing a weighted independent set, and an iterative framework that alternates between candidate computation and selection. Our method is shown to be highly successful in achieving the prescribed topology on a large test suite involving many complex 3D shapes and target topologies.

CCS Concepts: • **Computing methodologies** → **Volumetric models**; **Mesh models**; **Shape analysis**.

ACM Reference Format:
Linxuan Rong and Tao Ju. 2026. Persistence-guided Prescribed Topological Simplification. *ACM Trans. Graph.* 45, 4, Article 165 (July 2026), 16 pages. https://doi.org/10.1145/3811384

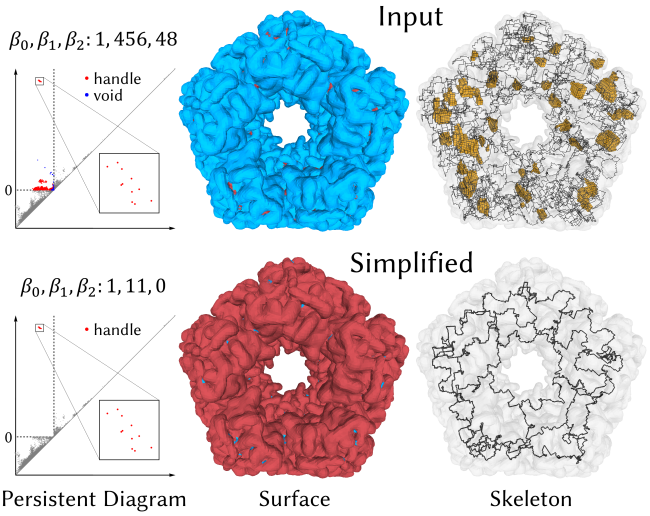


Fig. 1. Top: The *Enzyme* has 456 handles and 48 voids, as shown in the persistent diagram (left, red and blue dots are handles and voids) and the skeleton (right, yellow surfaces bound voids). Bottom: Our method simplifies it to a prescribed topology of 11 handles and 0 voids by removing all but the most persistent features (boxed in the persistent diagram). Now the skeleton reveals the dodecahedral structure of the protein. Red (blue) parts on the input (simplified) surface are fills (cuts) made by simplification.

1 Introduction

Topology is concerned with properties of a shape that are invariant under continuous deformations. A key topological property is the set of "holes", or *homology classes*, such as connected components, handles, and voids of a 3D shape. Many shapes in nature have a *trivial* topology consisting of a single component with no handles or voids, while others may have a non-trivial, and even complex topology. For example, the heart has several chambers (voids), the skeleton consists of many bones (components), and a bone may have many tunnels (handles). Man-made shapes, such as furniture and mechanical parts, are often non-trivial in topology as well.

Shapes reconstructed from real-world data, however, often have a topology different from, and usually more complex than, the topology of the target object. An incorrect topology can be detrimental to domain applications, whether they rely on a trivial topology of the shape, such as mapping cortical surfaces [Shattuck and Leahy 2001] or analyzing the branching structure of roots [Zeng et al. 2021], or are interested in specific topological features [Wu et al. 2017]. A complex topology also creates difficulties for a range of geometric processing tasks, such as mesh simplification, surface parameterization, shape matching, and physical simulations.

This work is concerned with the problem of *topological simplification*, or the removal of excess topological features from an input shape. Modifying the topology inevitably leads to changes to the geometry, either by adding more contents to (*filling*), or removing existing contents from (*cutting*), the input. Ideally, topological simplification should recover the topology of the target object while minimizing geometric changes.

Despite extensive research, a key challenge that remains is how to *control* the simplified topology (see Section 2). While some methods allow the user to specify the number or size of features to be preserved, such control is usually limited to a specific feature type, such as only handles [Chen and Wagenknecht 2006; Wood et al. 2004; Zhou et al. 2007] or only components and voids [Bauer et al. 2012; Edelsbrunner et al. 2006; Günther et al. 2014]. Methods that offer controlled simplification of all three types of features either are cumbersome to control, requiring the user to sketch [Ju et al. 2007] or provide additional shapes [Chambers et al. 2025; Zeng et al. 2020], or struggle to achieve the target topology [Kissi et al. 2024]. So far, no simplification method allows direct and precise control of all feature types.

We present a new method for topological simplification of 3D shapes that allows the user to *prescribe* the desired number of components, handles, and voids. Our method is based on *persistent homology* [Edelsbrunner et al. 2002], which identifies the topological features and equips them with a measure of importance (called *persistence*). Our goal is to remove all but the prescribed number of most

Authors' Contact Information: Linxuan Rong, Washington University in St. Louis, USA, l.rong@wustl.edu; Tao Ju, Washington University in St. Louis, USA, taoju@wustl.edu.

ACM 1557-7368/2026/7-ART165
https://doi.org/10.1145/3811384

important features while avoiding excessive geometric changes. This is achieved by three key technical contributions:

(1) An algorithm that computes candidate cuts and fills that remove topological features with low persistence (Section 5.3). Importantly, we show that using any subset of m pairwise disjoint cuts and fills removes exactly m low-persistence features (Proposition 5.1).
(2) An optimal and efficient algorithm for selecting the largest subset of disjoint cuts and fills with minimal geometric changes (Section 5.4). We formulate a *weighted independent set* problem in a bipartite graph and develop a polynomial-time algorithm using graph-cut.
(3) An iterative framework that alternates between the computation and selection of candidates to maximally simplify the topology (Section 5.5).

Our core algorithm can be applied to shapes in *any* dimension represented as a *cell complex*, which is a collection of elementary cells with trivial topology (e.g., simplices and hyper-cubes). To accommodate common representations in 3D, such as implicit surfaces and triangular meshes, we provide routines to obtain cell complexes from those representations as well as to convert the simplified cell complexes into manifold boundary meshes (Section 6).

Although lacking a theoretical guarantee of success, our method is highly effective in practice: it achieves the prescribed topology in *every* example in our test suite, which consists of more than a thousand combinations of complex 3D shapes and target topologies (Section 7). We demonstrate the potential application of our method in domains such as structural biology (Figure 1), plant phenotyping (Figure 13), and medical imaging (Figures 10 and 11).

2 Related works

2.1 Topological simplification of shapes

Many topological simplification methods are designed to remove handles on a 3D surface [Wood et al. 2004; Zhou et al. 2007], particularly the genus-0 cortical surface [Chen and Wagenknecht 2006; Han et al. 2002; Kriegeskorte and Goebel 2001; Shattuck and Leahy 2001], but they do not treat connected components or voids. Simplification of all types of topological features can be achieved by morphological operations [Nooruddin and Turk 2003], applying persistence-based cuts or fills [Chambers et al. 2018; Wu et al. 2017], or inflating (or deflating) an initial shape with simple topology while selectively permitting topological changes [Bischoff and Kobbelt 2002; Szymczak and Vanderhyde 2003]. These methods, however, either only cut or only fill, and hence they may incur excessive changes to the shape (see a comparison later in Figure 14).

We know of only a few methods capable of simplifying all types of topological features in 3D while selectively cutting or filling, and yet none provide easy and direct means for specifying the target topology. Ju et al. [2007] modify the input shape to match the topology of a user-provided 3D sketch. This requires a graphical interface for sketching, and a non-trivial topology (e.g., one with voids or many handles) can be time-consuming to sketch. Both Zeng et al. [2020] and Chambers et al. [2025] tackle the *homological simplification* [Attali et al. 2015] problem, which is to simplify an input shape so that the only remaining topological features also appear in a given inner shape (*core*) and outer shape (*neighborhood*). These methods place the burden on the user to find a suitable pair of core and neighborhood, whose shared topology is the desired topology of the input shape.

One of the problems that our method addresses is selecting candidate cuts and fills that maximally simplify topology with minimal geometric changes. This is formulated in [Zeng et al. 2020] as a Node-Weighted Steiner Tree problem, which is NP-hard. Thanks to the properties of the cuts and fills computed by our persistence-guided method (Section 5.3), we formulate the problem as finding a weighted independent set on a bipartite graph, which can be solved in polynomial time (Section 5.4).

2.2 Topological simplification of functions

A related problem is modifying a *function* to simplify the topology of *all* its level sets. Function simplification is useful, for example, for visualizing large and noise-ridden scalar fields [Heine et al. 2016]. There are two common strategies for function simplification. *Combinatorial* methods manipulate the *ordering* of vertex values, and they can offer strong guarantees of topological minimality and error bounds in 2D [Bauer et al. 2012; Edelsbrunner et al. 2006; Lukasczyk et al. 2020; Tierny and Pascucci 2012]. However, current combinatorial methods are limited to removing local extrema of the function and therefore cannot simplify handles in 3D (which involve saddle points). The same limitation applies to methods that simplify level sets using contour trees [Carr et al. 2010].

On the other hand, *numerical* methods directly optimize the function values to maximally remove the critical points. While earlier works only remove local extrema [Bremer et al. 2004; Günther et al. 2014; Patanè and Falcidieno 2009; Weinkauf et al. 2010], more recent works can act on all types of critical points (thus simplifying all types of topological features) by back-propagating gradients from the persistence of topological features, an approach often known as *persistence optimization* [Carriere et al. 2021; Kissi et al. 2024; Nigmetov and Morozov 2024; Poulenard et al. 2018; Solomon et al. 2021]. However, numerical optimization may get stuck at local minima and hence fail to find a solution with the desired topology (see Figures 11 and 13).

2.3 Topology-aware reconstruction

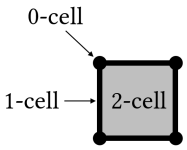
Some surface reconstruction methods allow the user to control the topology of the output, either using a graphical user interface [Sharf et al. 2007; Yin et al. 2014], by providing templates or seeds [Han et al. 2003; Ségonne 2008; Sharf et al. 2006; Zeng et al. 2008], or prescribing the number of handles [Huang et al. 2017; Lazar et al. 2018; Zou et al. 2015]. More recently, persistence optimization enables the user to specify the target topology using persistence [Brüel-Gabrielsson et al. 2020; Dong et al. 2022; Gao et al. 2022; Hu et al. 2025]. However, these numerical methods often fail to achieve the desired topology. To date, no existing method offers direct and precise control of all types of topological features. Furthermore, topological simplification is still necessary for surfaces produced by many reconstruction methods that are not topology-aware.

3 Preliminaries

We give a brief account of foundational concepts and results used in our method. Please refer to standard texts on algebraic topology [Hatcher 2002] and persistent homology [Edelsbrunner et al. 2008; Otter et al. 2017] for in-depth discussions.

3.1 Cell complex and homology

We consider a shape represented as a collection of elementary entities called *cells*. A k-dimensional cell, or a k-cell, is a shape homeomorphic to an open k-dimensional sphere. A 0-cell is called a *vertex*. The *boundary* of a k-cell ($k > 0$) σ is a set of $(k-1)$-cells bounding σ. For example, a quad is a 2-cell, whose boundary consists of four 1-cells (edges), each bounded by two 0-cells (vertices); see insert. A boundary cell ϕ of σ is called a *face* of σ, and σ a *co-face* of ϕ.



A *cell complex* is the union of cells and their faces. Some examples are the *cubical complex*, where each k-cell is a k-dimensional hypercube with 2^k vertices, and the *simplicial complex* (e.g., a triangular or tetrahedral mesh), where each k-cell is a k-dimensional simplex with $k+1$ vertices. The dimensionality of a cell complex is defined as the highest dimensionality of its cells.

Informally, *homology* characterizes the "holes" in a shape. Formally, a set of k-cells is called a *k-chain*, whose boundary is the symmetric difference of all $(k-1)$-faces. A k-chain with an empty boundary is called a *k-cycle*, and a *k-boundary* is the boundary of some $(k+1)$-chain. Since the boundary of a boundary is always empty, every k-boundary is a k-cycle, but the inverse is not always true. The *k-th homology group* of a cell complex X is the group of k-cycles factored by the group of k-boundaries. Each member of the group is an equivalent class called *homology class*, consisting of all non-boundary k-cycles that can be transformed into each other by taking algebraic sums with some k-boundaries. Intuitively, each k-th homology class represents a k-dimensional hole of X, such as a connected component ($k=0$), a handle ($k=1$), and a void ($k=2$). The rank of the k-th homology group (i.e., the number of k-th homology classes) is known as the *k-th Betti number*, β_k.

3.2 Filtration and persistent homology

While homology characterizes the holes in a shape, *persistent homology* tracks the creation and destruction of holes as the shape evolves. The evolution is represented by a sequence of expanding cell complexes, known as a *filtration*, that are subsets of a cell complex $\mathbb{X}$. Starting from a single cell, the filtration adds one cell of $\mathbb{X}$ at a time according to some cell ordering π, known as a *filter*, which has the property that every cell is preceded by its faces. The filter order ensures that each subset of $\mathbb{X}$ in the filtration is a cell complex. An example filtration is shown in Figure 2 top.

Informally, a *k-th persistent homology class* is a k-dimensional hole shared by a sequence of complexes in the filtration. Formally, it is a member of the *k-th persistent homology group*, which is the group of k-cycles in a complex X in the filtration factored by the group of k-boundaries in another complex Y later in the filtration. As the complexes grow, a k-th persistent homology class is created

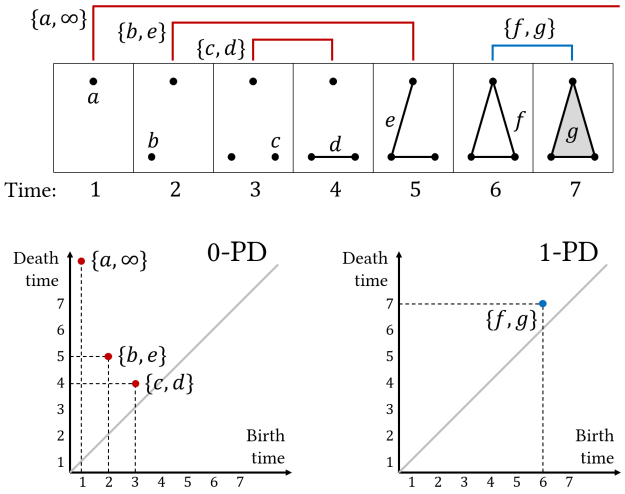


Fig. 2. Top: a filtration and the time function. Each bracket marks the duration of a 0-th (red) or 1-st (blue) persistence homology class, whose p-pair is noted by the bracket. Bottom: Visualizing the k-th ($k=0,1$) p-pairs by their birth and death times in the k-th persistence diagrams.

when some k-cell b is added and is destroyed when another $(k+1)$-cell d is added ($d=\infty$ if the class is never killed). We call b the *birth cell* and d the *death cell*, and $\{b,d\}$ a *persistent pair* (or *p-pair*). In Figure 2 top, adding cell c introduces a new connected component, which is killed by adding d. This 0-th persistent homology class is thus represented by p-pair $\{c,d\}$. Similarly, f introduces a void that is killed by g, creating a 1-st persistent homology class represented by p-pair $\{f,g\}$. The component created by a remains at the end of the filtration, which creates a p-pair $\{a,\infty\}$. Thereafter, we shall refer to a persistent homology class and its p-pair interchangeably. Note that each k-cell of $\mathbb{X}$ is either the birth cell in some k-th p-pair or the death cell in some $(k-1)$-th p-pair.

A key product of persistent homology is a measure of "importance" of topological features. Consider a *time function* f over all cells of $\mathbb{X}$ such that it is non-decreasing in the filter π. Intuitively, $f(\sigma)$ is the "time" when $\sigma \in \mathbb{X}$ is added to the filtration. Given a p-pair $\{b,d\}$, the times $f(b), f(d)$ are called the *birth time* and *death time*, respectively. The difference $f(d)-f(b)$, known as its *persistence*, measures the "life span" of the corresponding persistent homology class. A high persistence indicates a long-lasting topological feature, which is potentially more important, whereas a low persistence indicates a transient feature that is more likely to be noise.

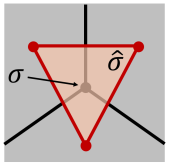
The p-pairs and their persistence can be visualized intuitively in a two-dimensional *persistence diagram* (PD), whose axes are the birth and death times of each p-pair. We call the PD of all k-th p-pairs the k-th PD. As an example, the 0-th and 1-st PDs for the filtration in Figure 2 top are shown at the bottom. Note that all p-pairs lie above the diagonal line in the PD, and those further away from the diagonal have higher persistence.

3.3 Dual complex

Our algorithms operate on both the shape and its background (the complementary space). For this purpose, it would be convenient to

represent both as a cell complex. Unfortunately, given a cell complex X (the shape) as a subset of a larger complex $\mathbb{X}$, the complement $\bar{X} = \mathbb{X} \setminus X$ is generally not a cell complex (that is, a face of some cell in $\bar{X}$ is not in $\bar{X}$). This inconvenience can be resolved by considering the *dual* of the cells.

Consider an n-dimensional cell complex $\mathbb{X}$. The *dual cell* $\hat{\sigma}$ of a k-cell $\sigma \in \mathbb{X}$ is a $(n-k)$-cell whose faces (if $k < n$) are dual cells of the co-faces of σ. For example, consider a 2D cell complex $\mathbb{X}$. The dual $\hat{\sigma}$ of a 0-cell (vertex) σ is a 2-cell (polygon). Each edge of $\hat{\sigma}$ is dual to an edge in $\mathbb{X}$ incident to σ, and each vertex of $\hat{\sigma}$ is dual to a polygon in $\mathbb{X}$ incident to σ; see insert. Let $\hat{X}$ denote the union of the dual of all cells in a subset $X \in \mathbb{X}$. It is easy to show that, if X is a cell complex, then so is *the dual of its complement*, $\tilde{X} = \hat{\bar{X}}$. This enables a cell-complex-based algorithm to be applied to both the shape (X) and its background ($\tilde{X}$).



While persistent homology captures the topology of the evolving shape, *persistent relative cohomology* does the same for the topology of the evolving background. Note that the reverse order of the filter π, denoted by $-\pi$, is a filter of the dual complex $\hat{\mathbb{X}}$. Whereas the filtration $\{\mathbb{X}, \pi\}$ expands the shape, the filtration $\{\hat{\mathbb{X}}, -\pi\}$ expands the background. By duality [De Silva et al. 2011], the two filtrations have matching persistent (co-)homology groups at complementary dimensions: $\{b, d\}$ $(d \neq \infty)$ is a k-th p-pair of the shape filtration if and only if $\{\hat{d}, \hat{b}\}$ is a $(n-k-1)$-th p-pair of the background filtration. Intuitively, the creation (destruction) of a k-dimensional hole as the shape evolves coincides with the creation (destruction) of a $(n-k-1)$-dimensional hole in the background.

4 Overview

Our method simplifies a shape to have a prescribed number of topological features. Specifically, given an input shape $S \subset \mathbb{R}^n$ and target Betti numbers $b_k \leq \beta_k(S)$ for $k = 0, \ldots, n-1$, we seek a new shape S' such that $\beta_k(S') = b_k$ for all k and the difference $|S' - S|$ is as small as possible. We proceed in three steps:

(1) Construct a filtration such that S is approximated by one of the complexes in the filtration (called the *shape complex*).
(2) Modify the filtration to remove all k-th $(0 \leq k < n)$ persistent homology classes (p-pairs) present in the shape complex, except those whose persistence is among the highest b_k.
(3) Extract a shape S' from the modified shape complex that preserves its Betti numbers.

These steps are illustrated on a 2D example in Figure 3. The input shape (a) has 2 components and 3 voids, and the target is 1 for each type (i.e., $b_0 = b_1 = 1$). The filtration is created on a cubical complex (b), whose persistent diagrams highlight the p-pairs present in the shape complex (red dots). Such p-pairs are in the top-left quadrant of coordinates $(0, 0)$. After simplification, only p-pairs with the highest persistence in each dimension remain in the shape complex. Observe in the final shape (d) that our method performs cutting or filling as needed to minimize geometric changes.

The core of our method is filtration simplification (Step 2), which will be presented first in Section 5. Our simplification algorithm is applicable to filtrations in arbitrary dimensions. Then, in Section 6, we describe routines tailored for 3D to convert a shape in a different representation (e.g., implicit surface or mesh) to a filtration (Step 1) and to extract a manifold boundary mesh from the simplified shape complex (Step 3).

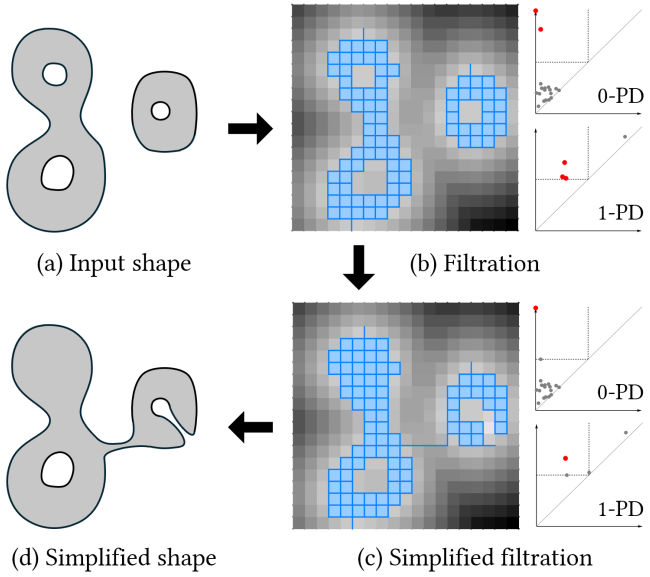


Fig. 3. Overview of our method: given an input shape (a), a filtration is created (b) and simplified (c), and the simplified shape is extracted (d). The filtrations in (b,c) are visualized by the time values of the 2-cells and overlayed with the shape complexes (blue). The persistence diagrams highlight the active p-pairs (red dots), which are reduced to the prescribed numbers in (c), with one 0-th p-pair (a component) and one 1-st p-pair (a void).

5 Filtration simplification

We consider modifying a given filtration to simplify those persistent homology classes that are *present* in one of the complexes – the shape complex. We say such classes (and their p-pairs) are *active*. Note that our problem is different from, and simpler than, the simplification of *all* persistent homology classes, which has been extensively studied but remains a challenging problem in 3D (see Section 2.2). By restricting to only active features, we show that a practically effective solution can be found.

5.1 Problem statement

Consider a filtration defined by an n-dimensional cell complex $\mathbb{X}$, a filter π, and a time function $f : \mathbb{X} \to \mathbb{R}$. We assume that $\mathbb{X}$ has a trivial topology and $f(\sigma) \neq 0$ for any $\sigma \in \mathbb{X}$. The *shape complex* $\mathbb{X}_f$ is defined as the union of cells where $f < 0$. A p-pair $\{b, d\}$ is *active* if $f(b) < 0 < f(d)$. We say a k-th p-pair is *major* if its persistence is among the top b_k in all active k-th p-pairs, and *minor* otherwise.

We seek a new filter π' and time function f' such that the new filtration defined by $\{\mathbb{X}, \pi', f'\}$ has the identical set of major p-pairs as $\{\mathbb{X}, \pi, f\}$ and as few minor p-pairs as possible. In addition, the new shape complex, $\mathbb{X}_{f'}$, should be as close to $\mathbb{X}_f$ as possible.

We say a solution pair $\{\pi', f'\}$ is *maximal* if the filtration $\{\mathbb{X}, \pi', f'\}$ has no minor p-pairs. That is, the simplified shape complex $\mathbb{X}_{f'}$ has the prescribed Betti numbers b_k. A maximal solution always exists if the target topology is trivial (e.g., $\mathbb{X}_{f'}$ may consist of a single vertex). For non-trivial target topologies, a maximal solution may not exist. Our method attempts to remove as many minor p-pairs as possible, but finding a maximal solution, if one exists, is not guaranteed.

5.2 Outline of approach

Our simplification works by identifying groups of cells, called *cuts* and *fills*, and re-ordering a selected subset of them in the filtration. The cuts consist of cells with negative times, the fills consist of cells with positive times, and both are associated with minor p-pairs. Re-ordering flips the signs of cells in the selected cuts and fills, effectively deleting the selected cuts from and adding the selected fills to the shape complex.

Given selected cuts C and fills F, re-ordering proceeds as follows. Let σ^- be the last negative cell in the filter π, and σ^+ the cell right after σ^-. The new order reads $\{\ldots, \sigma^-, F, C, \sigma^+, \ldots\}$, with cells in C and F maintaining their relative order in π. Accordingly, all cells in C are given a positive time smaller than $f(\sigma^+)$, and all cells in F a negative time greater than $f(\sigma^-)$. We call such re-ordering a *migration*; see Figure 4 for an illustration.

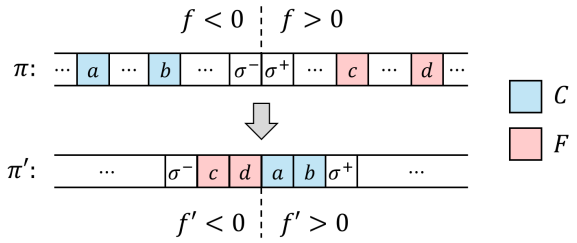


Fig. 4. Migrating sets C, F produces a new cell order π' and function f'.

What remains is finding the suitable cuts and fills whose migration yields a valid filtration with as few minor p-pairs as possible while minimizing geometric changes. We first introduce an algorithm that identifies a candidate set of cuts and fills, such that migrating any subset of m mutually disjoint candidates yields a filtration that removes *exactly* m minor p-pairs (Section 5.3). We then present an efficient and optimal algorithm to select a maximal disjoint subset of candidates with minimal geometric measures (Section 5.4). Finally, we present an iterative framework to maximally simplify the filtration (Section 5.5).

5.3 Computing cuts and fills

To remove a minor p-pair $\{b, d\}$, a natural candidate for migration is the birth cell b or death cell d: if the time sign of either cell is flipped, both cells would have the same sign, and the p-pair would no longer be active. However, the new cell order after migration may not be a filter (i.e., every cell is preceded by its faces), if b (resp. d) moves past one of its co-faces (resp. faces). In this case, removing b from or adding d to $\mathbb{X}_f$ would invalidate the cell complex. While one could migrate the (co-)faces together with b or d, doing so may affect other, or introduce new, persistent homology classes.

We present a way to "grow" a birth or death cell to a group of cells, so that migrating those groups produces a valid filtration with only the intended minor p-pairs removed. This is achieved by adopting the concept of *generating sets* introduced by Ju et al. [2007], which we shall briefly review first.

Given a cell complex X, we say a cell $\sigma \in X$ is *isolated* if it has no co-face in X. An isolated cell σ and a face ϕ form a *simple pair* if σ is the only co-face of ϕ in X. The remainder of X after removing the simple pair $\{\phi, \sigma\}$ is another cell complex that is homotopy equivalent to X (see Figure 5). Such removal is an example of a *deformation retract* [Hatcher 2002].

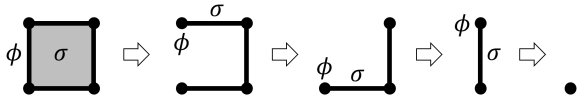


Fig. 5. Sequentially removing simple pairs $\{\phi, \sigma\}$ from a cell complex.

Consider a subset $K = X \setminus \Pi$ where Π is a sequence of simple pairs. The *generating set* $G_{X,K}(\sigma)$ of an isolated cell $\sigma \in K$ is the smallest subset of $\Pi \cup \{\sigma\}$ that includes σ and the co-faces of all cells in the set. Intuitively, $G_{X,K}(\sigma)$ is an "expansion" of σ from K to X. Importantly, the remainder $X \setminus G_{X,K}(\sigma)$ is another cell complex and homotopy equivalent to $K \setminus \sigma$. As an example, Figure 6 left shows the generating sets (green) of several isolated 1-cells (blue) on K (black).

We use generating sets to grow birth cells into cuts. We first compute a *skeleton* K of the shape complex $\mathbb{X}_f$ by sequentially removing simple pairs until no such pairs remain. Intuitively, K is the minimal subset of $\mathbb{X}_f$ that retains its topology. For technical reasons (see Appendix A), we do not remove cells that lie on the representative cycle for each active (major or minor) persistent homology class. Such cycles, called *generators*, can be obtained from standard persistent homology computations along with the p-pairs. Importantly, the generator for a p-pair $\{b, d\}$ always contains its birth cell b, and hence keeping the generators ensures that b is in the skeleton K. If b is isolated in K, we define the *cut* of b as the generating set $G_{\mathbb{X}_f,K}(b)$. Figure 6 left shows the skeleton K (black) of the shape complex in Figure 3 and the cuts (green) of three isolated birth cells (blue), which create the voids. Note that birth cells b_1, b_2, which create the connected components, are not isolated in K, and hence have no cuts.

We can similarly grow death cells to form fills by working on the background – the dual of the complement of $\mathbb{X}_f$, denoted by $\tilde{\mathbb{X}}_f$ (see definition in Section 3.3). We first compute a *background skeleton* $\tilde{K}$ by maximally removing simple pairs from $\tilde{\mathbb{X}}_f$ while preserving the representative cycles (called *co-generators*) for all active persistent *cohomology* classes. The co-generator for each class, which is also represented by some p-pair $\{b, d\}$, always contains the dual of its death cell, $\hat{d}$. If $\hat{d}$ is isolated in $\tilde{K}$, we define the *fill* of d as the dual of the generating set $G_{\tilde{\mathbb{X}}_f,\tilde{K}}(\hat{d})$. Figure 6 right shows the dual of $\tilde{K}$, consisting of 2- and 1-cells in the primal complex, and the fills (green) of three isolated death 2-cells (red quads), which kill the voids, and an isolated death 1-cell (red edge), which kills a component.

We say that a cut and a fill are *disjoint* if no cell in the cut has a co-face in the fill. Intuitively, a pair of disjoint cut and fill simplifies

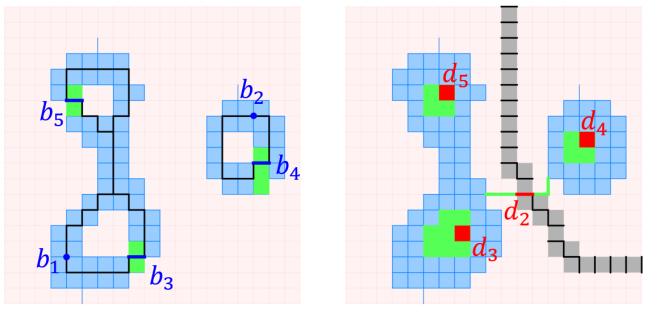


Fig. 6. Cuts and fills of active p-pairs $\{b_i, d_i\}$ of the filtration in Figure 3 (b). Left: the skeleton K (black) and the cuts (green) of isolated birth cells (blue edges). Birth cells b_1, b_2 are not isolated and hence have no cuts. Right: the dual of the background skeleton $\tilde{K}$ (black edges and gray squares, plus d_3, d_4, d_5) and the fills (green) of isolated death cells (red edges and squares).

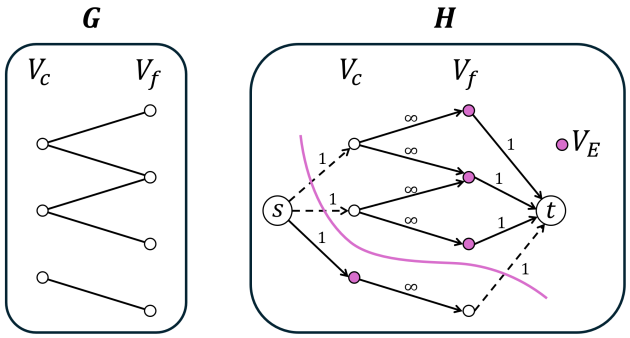


Fig. 7. Left: the bipartite graph G of cuts and fills. Right: the flow network H used in König's graph cut algorithm. Dotted edges are in a minimum s-t cut E, and purple vertices form the set V_E, an MIS of G.

different features of the shape complex. Our main result is that migrating disjoint cuts and fills removes minor p-pairs while retaining all major p-pairs and a valid filtration:

Proposition 5.1. *Let π', f' be the cell order and time function after migrating a set of m cuts and fills of minor p-pairs, such that each cut is disjoint from each fill. Then,*

(1) *$\{\mathbb{X}, \pi', f'\}$ defines a filtration.*
(2) *Filtrations $\{\mathbb{X}, \pi', f'\}$ and $\{\mathbb{X}, \pi, f\}$ have the same set of major p-pairs.*
(3) *Filtration $\{\mathbb{X}, \pi', f'\}$ has m fewer minor p-pairs than filtration $\{\mathbb{X}, \pi, f\}$.*

We prove the proposition in Appendix A. This property enables precise control of the simplified topology while only requiring a pairwise condition (disjoint-ness) between cuts and fills. As we shall see next, a maximal set of pairwise disjoint cuts and fills that minimize some geometric cost can be found in polynomial time.

5.4 Selecting cuts and fills

To remove as many minor p-pairs as possible while minimizing changes to the shape complex, we select a maximal set of mutually disjoint cuts and fills with a minimal total "size". Here, the size of a cut or fill can be defined by any geometric measure of the user's choice (see more discussion in Section 7). This selection problem can be formulated as finding a *weighted* independent set.

We represent all cuts and fills computed from minor p-pairs as an undirected graph G. The vertices of G are divided into two sets $\{V_c, V_f\}$, such that each vertex of V_c (V_f) represents a cut (fill). Two vertices $u \in V_c$ and $v \in V_f$ are connected by an edge if their corresponding cut and fill are *not* disjoint. Note that G is *bipartite*, as there is no edge connecting vertices within V_c or V_f (see Figure 7 left). Given a geometric measure w as a vertex cost function, the solution to our selection problem is a maximum independent set (MIS) of G with the minimal total vertex cost.

While finding an MIS is NP-hard on general graphs, it can be solved in polynomial-time on a bipartite graph, thanks to König's theorem. Let H be a flow network that includes all vertices and edges of G, so that each edge of G is directed from a node in V_c to a node in V_f and associated with capacity ∞. In addition, H adds two additional vertices, s, t, as well as edges from s to each vertex of V_c and from each vertex of V_f to t, all with capacity 1 (see Figure 7 right). Let E be a minimum s-t cut of H, and V_E the union of vertices in V_c connected to s and vertices in V_f connected to t after removing the cut E. Then V_E is an MIS of the original graph G.

We can modify the graph cut algorithm above to solve our weighted MIS problem. To do so, we set the capacity of each edge $\{s, v\}$ or $\{v, t\}$ in H to be $W - w(v)$ (instead of 1), where W is the total vertex cost of all vertices in $V_c \cup V_f$. Let the modified network be H^*, we can show that (see Appendix B)

Proposition 5.2. *If E^* is a minimum s-t cut of H^*, then V_{E^*} is an MIS of G with minimum total vertex cost.*

Selecting cuts and fills using graph cut has the time complexity $O(|V|(|V| + |E|)^2)$, where $V = V_c \cup V_f$ and E are the edges of G. Since $|V|$ (number of topological features to be removed) and $|E|$ (number of pairs of non-disjoint cuts and fills) are typically small in real-world data (at most in the thousands), the algorithm runs efficiently in practice.

5.5 Iterative simplification

While the algorithm above always finds the largest disjoint subset in a given set of cuts and fills, this subset may be smaller than all minor p-pairs, resulting in a non-maximal simplification. Note that maximal simplification is guaranteed if each minor p-pair has both a cut and a fill, because a possible solution is the set of all cuts (or all fills). However, a p-pair may not have a cut (fill) if its birth cell (dual death cell) is not isolated in the skeleton K ($\tilde{K}$), such as b_1, b_2 in Figure 6. With missing cuts and fills, and constrained by disjoint-ness, the selection algorithm may fail to find a cut or fill for each minor p-pair. While this happens rarely in our experiments, we propose a remedy for it.

Our key observation is that a non-isolated birth (death) cell typically lies on the generator (co-generator) of another active p-pair. For example, birth cells b_1, b_2 in Figure 6 lie on the generators of p-pairs $\{b_3, d_3\}$ and $\{b_5, d_5\}$ (see Figure 2 (e)). The birth or death cell is "blocked" by the (co-)generator from being isolated in the skeleton. To remove residue minor p-pairs after simplification, a

simple strategy is to take the simplified filtration as input and simplify again. With fewer active p-pairs, fewer (co-)generators would be preserved by the skeletons, and there would be less blocking.

The observation leads to an iterative algorithm, as shown in Algorithm 1. To see that the algorithm terminates, note that at least one cut or fill is selected if the set of cuts and fills is not empty (otherwise the iteration stops). By Proposition 5.1, the number of minor p-pairs monotonically decreases in each iteration. In our experiments, our algorithm requires only one iteration for all except two examples, among a test suite of over a thousand examples, and those two examples need only two iterations. In all test cases, our algorithm returns a maximal simplification.

ALGORITHM 1: Filtration simplification

Input: Cell complex $\mathbb{X}$, filter π, time function f
Output: Modified filter π' and time function f'
$\{\pi', f'\} \leftarrow \{\pi, f\}$;
while *True* **do**
  Compute persistent (co-)homology of filtration $\{\mathbb{X}, \pi', f'\}$;
  Compute skeletons K of $\mathbb{X}_{f'}$ and $\tilde{K}$ of $\tilde{\mathbb{X}}_{f'}$ [Section 5.3] ;
  Compute cuts $\mathbb{C}$ and fills $\mathbb{F}$ for all minor p-pairs [Section 5.3] ;
  if $\mathbb{C} \cup \mathbb{F} = \emptyset$ **then**
    break
  end
  Select cuts $C \subseteq \mathbb{C}$ and fills $F \subseteq \mathbb{F}$ using graph-cut [Section 5.4];
  Update $\{\pi', f'\}$ by migrating $C \cup F$ [Section 5.2] ;
end
return $\{\pi', f'\}$

6 Shape simplification

To apply filtration simplification to 3D shapes, which are typically represented as meshes or implicit surfaces, we first need to convert these representations into a filtration. Furthermore, the shape complex in the simplified filtration generally does not have a manifold boundary surface, which reduces its utility in downstream geometry processing. We present routines to obtain filtrations from common 3D shape representations (Section 6.1) and to extract a 2-manifold surface from the simplified shape complex (Section 6.2). While tailored to 3D, these routines can be potentially generalized to higher dimensions.

6.1 Constructing filtrations

We consider 3D shapes defined implicitly (e.g., as a zero-level set) or explicitly (e.g., as a triangular mesh). The level set of a function $g : \mathbb{R}^3 \to \mathbb{R}$ is typically discretized on a spatial grid (e.g., cubical or tetrahedral), where g is sampled at the vertices. This grid naturally serves as the cell complex $\mathbb{X}$. We define the time function f to be the same as g at each vertex, and as the maximum of all vertex g values at other cells. The filter π is any ordering of cells with ascending f such that each cell is preceded by its faces. The shape complex $\mathbb{X}_f$ defined this way is topologically equivalent to the 0-level set of the piecewise linear interpolation of g on a tetrahedral grid and the 6-connectivity of grid points where $g < 0$ on a cubical grid.

To obtain a filtration from a triangular mesh, we can first convert the mesh to the implicit form by generating a signed distance volume from the surface. However, the conversion may lose fine geometric features. Alternatively, we can adopt existing tetrahedral meshing techniques to create a tetrahedral grid where the input surface is closely approximated [Fabri and Pion 2009; Hu et al. 2018] or exactly preserved [Diazzi et al. 2023; Si and Gärtner 2005] by a subset of the tetrahedral faces. To obtain a filtration, we first compute signed distances at each tetrahedral vertex to the input mesh (subtracted by a small constant to avoid zero values at vertices) and then follow the steps above for an implicit grid.

6.2 Extracting manifold meshes

After filtration simplification, the shape complex $\mathbb{X}_{f'}$ is a sub-complex of $\mathbb{X}$ consisting of all cells with negative values in the modified time function f'. We want to extract a manifold triangular mesh approximating the boundary of $\mathbb{X}_{f'}$, such that the interior of the mesh has the same topology (i.e., homology classes) as $\mathbb{X}_{f'}$.

One solution is to use a standard contouring method, such as Marching Tetrahedra and Marching Cubes, to extract an iso-surface of f' on the cell complex $\mathbb{X}$. However, as these methods only consider the values at the 0-cells (vertices) and ignore those at higher-dimensional cells, they may produce a topology that is different from $\mathbb{X}_{f'}$. For example, suppose $\mathbb{X}$ is a tetrahedron (3-cell) that is positive in f' but whose faces (vertices, edges, and triangles) are all negative. A void exists in $\mathbb{X}_{f'}$, but it does not in the output of Marching Tetrahedra, since all vertices are negative.

Instead of designing a new contouring algorithm, we propose to *refine* the cell complex $\mathbb{X}$ so that existing contouring algorithms produce the correct topology. Unlike the uniform refinement approach taken in [Ju et al. 2007], our refinement is *localized* to only those cells where vertex-based contouring produces a different topology from that of the negative sub-complex. This leads to only small increases in the output mesh size after refinement.

We say a cell of $\mathbb{X}$ is *trivial* if it is negative in f' or at least one of its vertices is positive, and *nontrivial* otherwise. As f' is a valid time function, a trivial cell is negative if and only if all its vertices are negative. In other words, its sign is completely determined by its vertex signs. As a result, a vertex-based contouring method produces the same topology in a trivial cell as $\mathbb{X}_{f'}$.

To resolve nontrivial cells, we process the cells of $\mathbb{X}$ from low to high dimensions. In each dimension, we subdivide all cells that are either nontrivial or contain some face that has already been

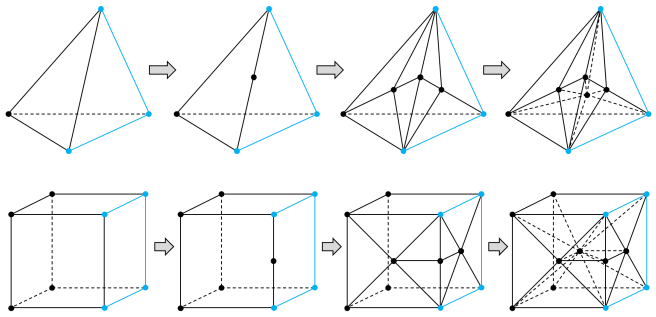


Fig. 8. Refining a tetrahedron (top) and a cube (bottom) by subdividing 1-cells, 2-cells, and 3-cells. Blue vertices and edges have negative values.

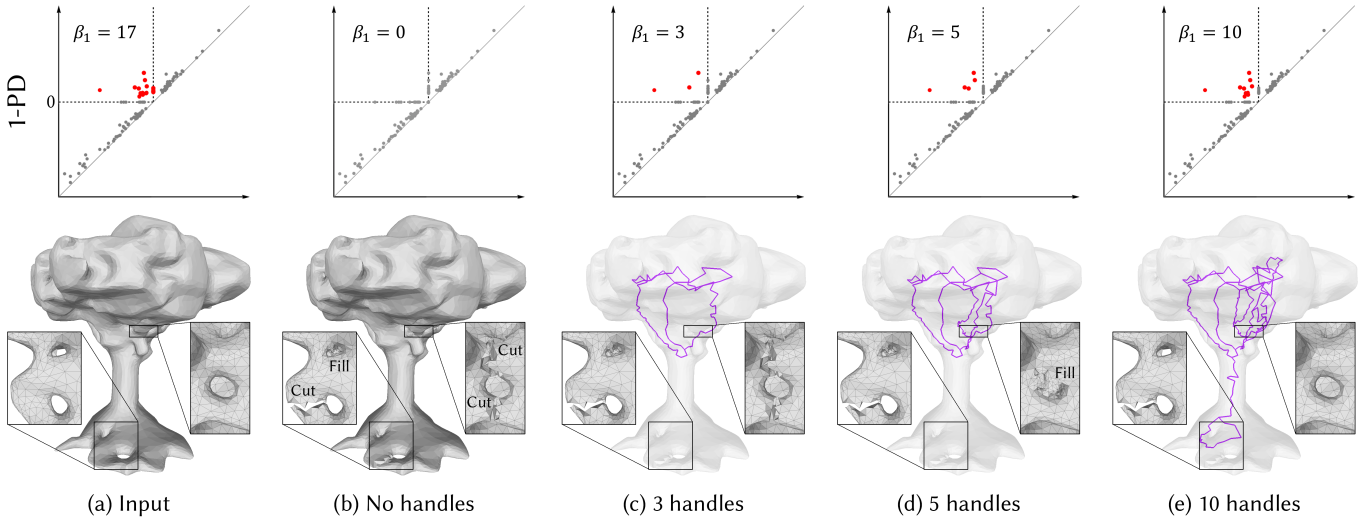


Fig. 9. Simplifying the Tree ($\beta_0/\beta_1/\beta_2$: 1/17/1) (a) while preserving 0, 3, 5, and 10 handles (b-e) and no voids. The 1-PDs of the original and simplified filtrations are shown at the top. Surfaces (some shape modifications shown in close-ups) and skeletons (purple) are shown at the bottom.

subdivided. Subdividing a k-cell σ ($k > 0$) removes σ, creates a vertex v at the centroid of σ, and creates a new cell connecting v to every face of σ. Each newly created cell inherits the same f' value as σ. It is easy to verify that refinement produces a valid cell complex and time function, and each cell in the refined complex is trivial.

Figure 8 demonstrates the process on a tetrahedron and a cube. Refining a tetrahedral complex produces another tetrahedral complex, and refining a cubical complex produces a mixed complex consisting of tetrahedra, cubes, and pyramids. We contour these cells using Marching Tetrahedra, Marching Cubes (with a modified look-up table that preserves the 6-connectivity of negative vertices), and Marching Pyramids (a simple variant of Marching Cubes).

7 Results

We demonstrate our method on a variety of 3D shapes while prescribing different target topologies (b_0, b_1, b_2).

7.1 Implementation and experiment setup

Our algorithm is implemented in C++. We adopt PHAT [Bauer et al. 2017] for computing persistent homology and use the *exhaustive* reduction algorithm [Edelsbrunner and Zomorodian 2003] to compute *canonical* generators (for computing skeletons). We observed that such generators result in better-shaped cuts and fills than those computed by the standard algorithm [Edelsbrunner et al. 2002].

We test on 3D shapes in either implicit or explicit form. For the latter, we use fTetWild [Hu et al. 2020] to obtain the tetrahedral complex from an input triangular mesh. Unless stated otherwise, the geometric measure (w in Section 5.4) of a cut (fill) is chosen as the absolute time value of the corresponding birth (death) cell in the filtration. This choice favors cuts and fills that are closer, in terms of the time function, to the shape complex. See a comparison of different choices of w in Figure 14.

We compare with the state-of-the-art methods for simplifying the topology of shapes [Zeng et al. 2020] and functions [Kissi et al. 2024]. The implementations of both methods require cubical complexes. Zeng's method requires the user to provide a pair of additional shapes (core and neighborhood) whose shared topological features exactly match the target topology. Such a pair is difficult to find for a nontrivial target topology, so we test this method using the trivial topology as the target by setting the core to be a single voxel and the neighborhood to be the entire cubical volume. Kissi's method [2024] simplifies all persistent homology features in the filtration whose persistence is lower than a user-given threshold. Our experiments showed that their method often leaves many features in the filtration, even at the highest threshold setting. We therefore use the highest threshold possible in each experiment.

7.2 Visual examples

In the following examples, the results are presented as the simplified shapes, their skeletons (to visualize handles and voids), and the k-PDs (i.e., persistent diagrams of the k-th p-pairs) of the filtrations for $k = 0, 1, 2$. Each PD highlights the active p-pairs as red dots, which lie in the top-left quadrant of $\{0, 0\}$. We omit the 0-th p-pair with an infinite death time. All examples are converted to cubical complexes except the Tree (Figure 9) and Helmet (Figure 15), which are converted to tetrahedral complexes.

We first test on the *Tree* (Figure 9), a triangular mesh with 1 component, 17 handles, and 1 void. We apply our method to remove the void ($b_2 = 0$) while preserving varying numbers of handles ($b_1 = 0, 3, 5, 10$). As shown in the 1-PDs, for each target topology, our method retains the top b_1 most persistent handles (i.e., dots furthest from the diagonal) in the original shape while removing all remaining ones. Observe from the close-up views that our method performs cutting or filling to minimize the chosen geometric measure. For example, in the left close-up in (b), the smaller tunnel (top) is filled while the larger one (bottom) is cut. The choice of cutting or filling for a feature may change with the target topology. For example, while our method decides to cut the two handles in

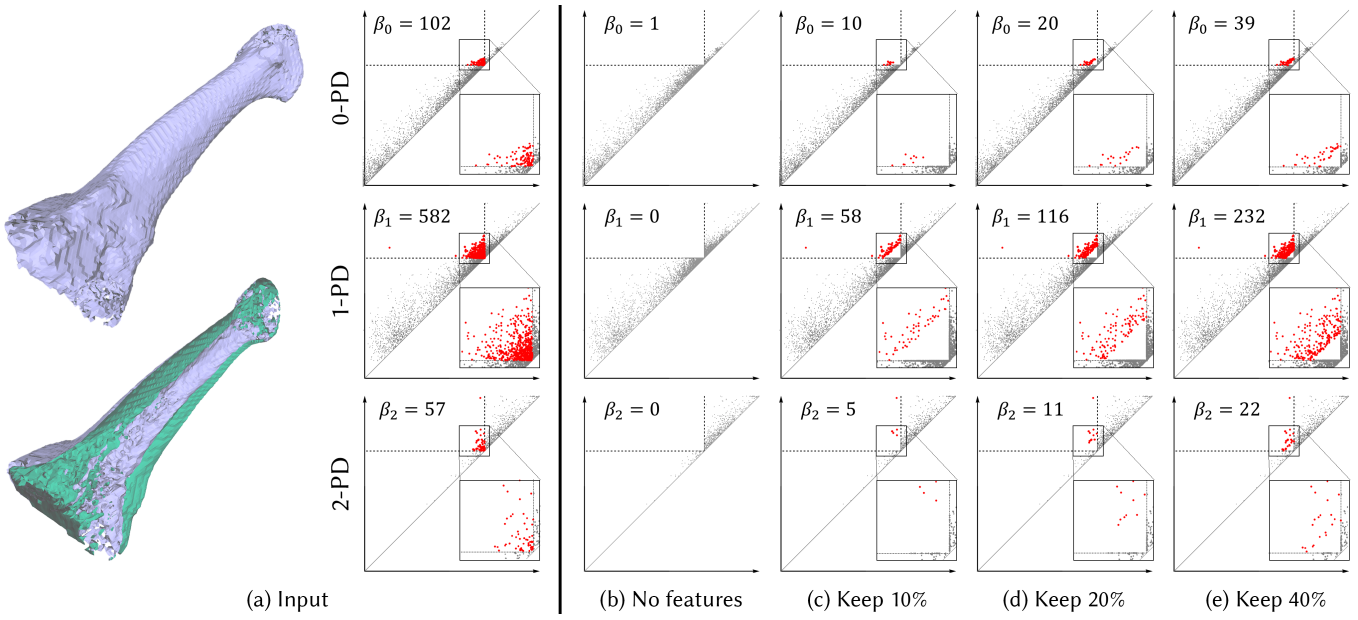


Fig. 10. Simplifying the Bone ($\beta_0/\beta_1/\beta_2$: 102/582/57) (a) while preserving the top x% most persistent features of each type for $x = 0, 10, 20, 40$ (b-e). The Bone surface and a cross-section are shown on the left.

the right close-up in (b), it switches to filling when just one of the handles needs to be removed, as shown in (d).

We next test our method on a more complex example, the *Bone*, which is an iso-surface of a CT scan (Figure 10). The shape contains 102 components, 582 handles, and 57 voids. We ask our method to simplify this shape while preserving x% of features of each type for $x = 0, 10, 20, 40$. Our method is able to realize each of these prescribed topologies. Observe from the PDs (and close-up views) that simplification strictly maintains features above some persistence level (i.e., distance to the diagonal line) while completely removing those under it. Our iterative algorithm requires only one iteration for all target topologies except the trivial topology (0%), where it uses two iterations. The first iteration removes all features except the most persistent handle, which is removed in the second iteration.

We compare our method with [Kissi et al. 2024; Zeng et al. 2020] on two iso-surfaces in Figures 11 and 13. The *Vessel* in Figure 11 (a) includes several disconnected portions of a larger vessel network, due to the small field of view. The two major portions are seen on the left and right. Due to insufficient imaging resolution, nearby vessels may get merged and form handles, and thin vessels are broken into disconnected pieces. Our method can simplify this input to a trivial topology (b), connecting broken pieces and disconnecting merged parts (see close-ups in Figure 12). However, doing so forces two major vessel parts to connect. They can be separated, with each part remaining handle-free, by prescribing two components as the target topology (c). Increasing the target component number separates the smaller vessel parts (d). In contrast, [Zeng et al. 2020] can only simplify to a trivial topology (e), and [Kissi et al. 2024] fails to remove the majority of the features (f).

The *Root* in Figure 13 (a) is our most complex example, with well over a thousand topological features. Observe from the cross-sectional view that there are large cavities inside the root stem and some of the thicker branches, which could be interesting from a biological perspective. While our method can fully simplify all topological features (b), it can also preserve a user-specified number of cavities while removing all remaining features (c,d). On this complex example, [Zeng et al. 2020] cannot achieve a full simplification, resulting in 2 handles and an extra component (e). [Kissi et al. 2024] produces a more topologically complex output, as it seems to have pulled noise from a different intensity range into the shape (f).

Our method can be used with any user-defined geometric measure of a cut or a fill. Different measures may lead to (sometimes significantly) different simplified shapes. We compare our time-based measure with a few others in Figure 14 using the Vessel example from Figure 11. The first choice, shown in (c), measures the size of a cut or fill by the number of cells in the set. While this choice minimizes the change to the volume of the input, it may introduce unnatural modifications caused by cuts or fills that are far away from the shape in the filtration (see red arrows). Another choice is to favor cuts over fills (d), by assigning a high cost to each fill, or vice versa (e). While the former leads to the loss of large components, the latter may produce overly large fills (see red arrows). In comparison, the result produced by using the time-based measure avoids excessive removals or additions (b).

We give an example where an alternative geometric measure produces a more desirable result. The triangular mesh *Helmet* in Figure 15 (a), reconstructed from a point cloud using [Xiao et al. 2023], contains many "holes" (handles) due to noise in the data. As the shape is thin and the holes are small, the birth and death

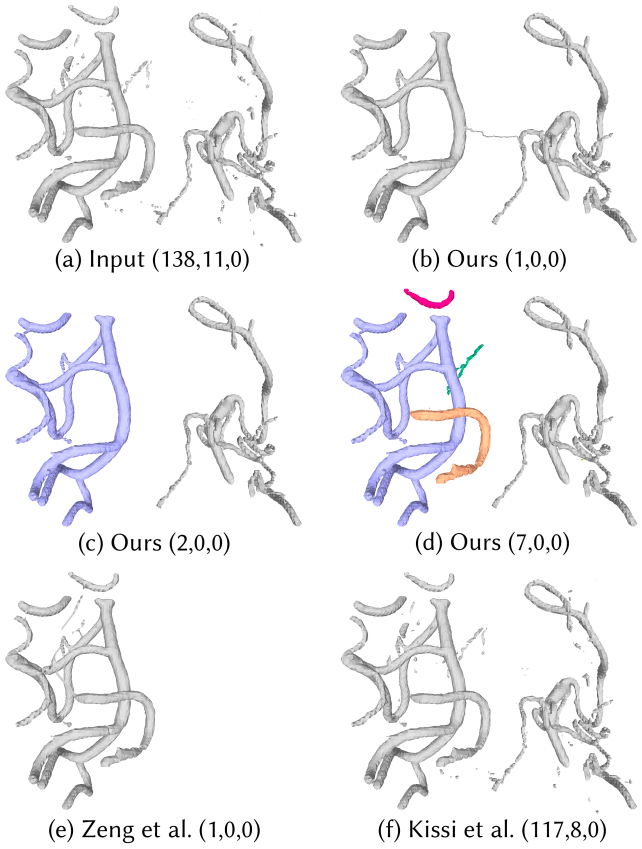


Fig. 11. Simplifying the Vessel (a) using our method while preserving 1 (b), 2 (c), and 7 (d) components and no handles, and comparing with [Kissi et al. 2024; Zeng et al. 2020]. Betti numbers of each shape are noted underneath, and connected components in (c,d) are colored.

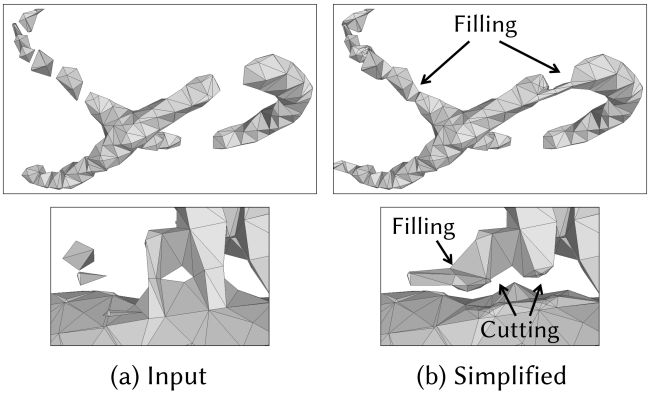


Fig. 12. Close-ups of regions in the Vessel example (Figure 11) before (a) and after (b) simplification to a trivial topology.

cells of these handles have similar and small time values in the filtration. As a result, the corresponding cuts and fills have similar costs in the time-based measure, making the choice between them

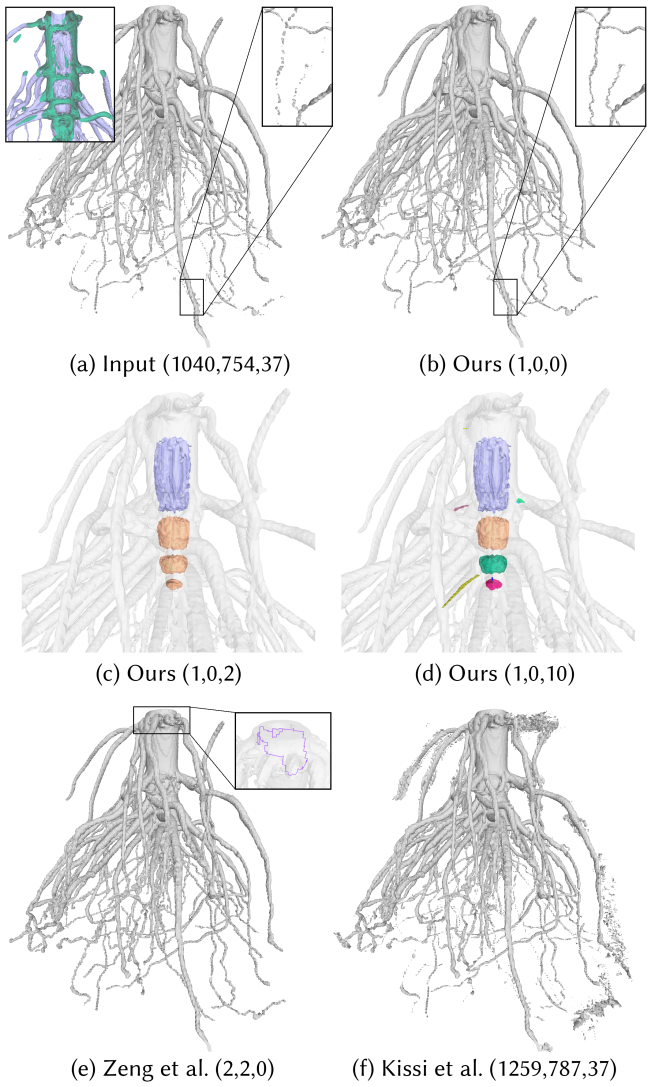


Fig. 13. Simplifying the Root (a) using our method either to a trivial topology (b) or by preserving 2 (c) or 10 (d) voids, and comparing with [Kissi et al. 2024; Zeng et al. 2020]. Betti numbers of each shape are noted underneath, and void components in (c,d) are colored. Inserts show a cross-section of the input (a), thin branches before and after simplification (a,b), and a skeleton showing handles not simplified by [Zeng et al. 2020] (e).

almost arbitrary. As seen in (b), while applying our method with 6 prescribed handles – the native topology of the Helmet – realizes that topology, it creates many cracks on the helmet due to cutting. Replacing the measure by one that favors fills over cuts, our method fills the unwanted holes without cracks (c).

While our method requires the user to prescribe the Betti numbers, this can be challenging if the shape's true topology is complex and unknown. In this case, the persistence computed in our method could help identify the desirable topology. This is demonstrated in Figure 16 for simplifying the *Skull*. We know the skull should have a single component and no void, but we do not know the correct

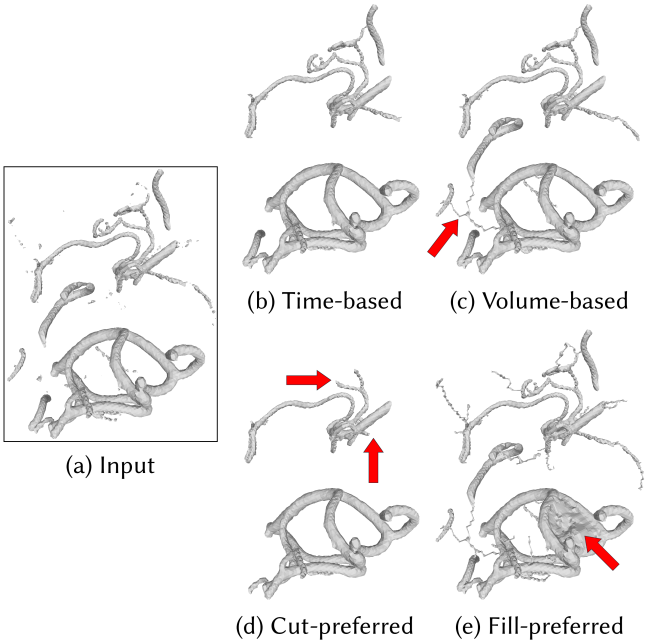


Fig. 14. Comparing different choices of geometric measures of cuts and fills in our method when simplifying the Vessel.

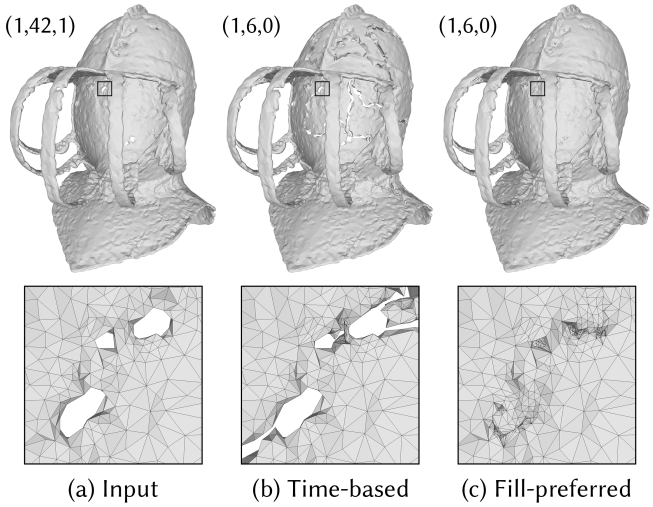


Fig. 15. Comparing the time-based (b) and fill-preferred (c) geometric measures when simplifying the Helmet (a) to preserve 6 handles and no voids. Betti numbers are shown by the shapes, and inserts show several "holes" in the Helmet removed by cutting (b) or filling (c).

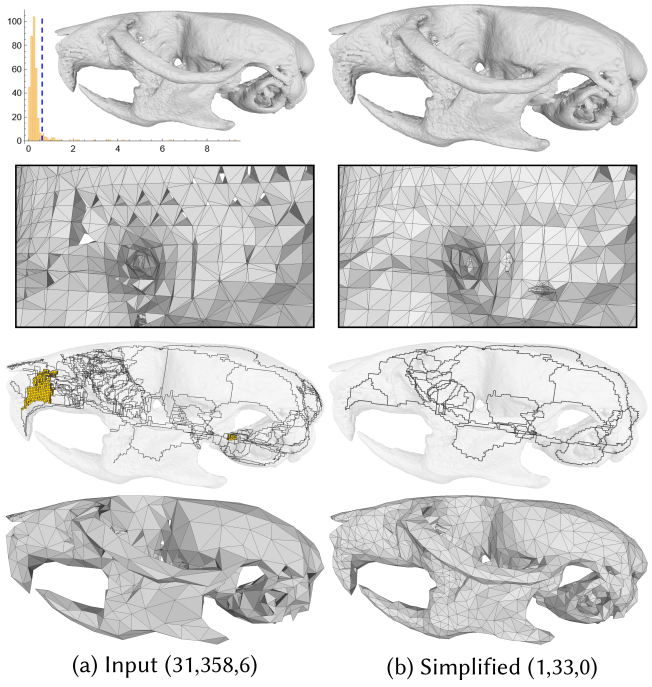


Fig. 16. Simplifying the Skull to preserve handles above a persistence threshold (0.6, blue line in the persistence histogram in top-left), showing the surfaces (top), a close-up view of a region riddled with handles in the input shape (2nd row), the skeletons (3rd row), and each mesh decimated to 5K triangles (bottom).

number of handles. By inspecting the histogram of persistence of handles (top-left), we found a plausible persistence threshold (0.6) that separates "noise" from "signal", above which there are 33 handles. Using (1,33,0) as the target, our method significantly reduces the topological complexity of the shape (as seen in the skeleton and close-up view) without losing those handles representing true anatomical features of the Skull. To further demonstrate the benefit of a simplified topology, we decimated both the input and simplified surfaces to the same mesh size (5K triangles) using [Garland and Heckbert 1997]. Observe that decimation on the topologically simplified surface preserves the geometry much better than on the input mesh, as the latter requires many triangles to maintain the topological features.

Finally, Figure 1 demonstrates a potential use case of our method in structural biology. The input shape is an iso-surface of the Pyruvate Dehydrogenase core, a large enzyme complex arranged in a dodecahedral structure. However, this structure is lost in its many topological noises (456 handles and 48 voids), as seen from its skeleton. Observe that, in the PD of the input filtration, there is a cluster of handles with much higher persistence than the rest (in the boxed region). These 11 handles capture precisely the dodecahedral connectivity. By removing all but those 11 features, our method produces a new shape that is geometrically similar to the input, but with a much more compact curve skeleton that clearly reveals the dodecahedral architecture.

7.3 Batch tests

In every example above, our method returns a maximal simplification – a simplified shape with the exact number of components, handles, and voids as prescribed. To further evaluate its robustness, we test our method on two large batches of examples.

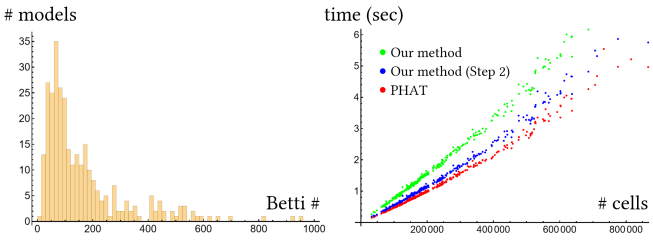


Fig. 17. Batch perturbation tests. Left: Histogram of the sum $\beta_0 + \beta_1 + \beta_2$ among the 310 perturbed shapes from Thingi10K (original sum ranging between 7-11). Right: Run time of PHAT (red), Step 2 (blue), and our entire method (green) as a function of cell complex size.

The first batch varies the target topology on the same input shape. We consider our two most complex inputs, the Enzyme (Figure 1) and the Root (Figure 13). For Enzyme ($\beta_0, \beta_1, \beta_2 = 1, 456, 48$), we first vary $b_2 = 1, \ldots, 48$ while fixing $b_1 = 0$, and then vary $b_1 = 1, \ldots, 456$ while fixing $b_2 = 0$. This creates more than 500 combinations of target topologies. For Root ($\beta_0, \beta_1, \beta_2 = 1040, 754, 37$), we vary each b_i ($i = 1, 2, 3$) independently between 0 (for b_1, b_2) or 1 (for b_0) and 10, which creates more than a thousand combinations. Our method achieves the target topology in all these 1500+ test cases, and only a single iteration is needed for each case.

The second batch adds noise to a collection of input shapes, and we use our method to recover their original topologies. We collect all models on Thingi10K [Zhou and Jacobson 2016] whose genus ranges between 6 and 10. We follow the same routine in Section 6.1 to convert each triangular mesh to a tetrahedral complex, except that a random noise up to 1% of the diagonal length of the bounding box is added to the signed distance at each vertex *before* the filtration is created. We then keep all models (310) that show an increase in topological complexity after perturbation. The increase is quite significant, as shown in the histogram of the sum of Betti numbers in Figure 17 left. We then run our method on each perturbed shape by prescribing the topology of the original shape (number of components, handles, and voids). Our method finds the maximal simplification in all 310 tests. Except for one case that requires two iterations, only a single iteration is needed.

In summary, our method achieves 100% success rate in finding a maximal simplification in our experiments. Except in rare (2 out of nearly 2000) cases where our method runs for two iterations, only a single iteration is necessary.

7.4 Performance

We report the timing for the visual examples in Table 1, listed in the order of increasing number of cells in the filtration. The timing is broken down by the steps of our algorithm (see Section 4), namely converting the input into a filtration (Step 1), simplifying the filtration (Step 2), and extracting the final surface (Step 3). The timing is only shown for a trivial target topology and a single iteration of Step 2. Different target topologies only differ in the selection of cuts of fills, and that takes a negligible amount of time compared to the rest of the algorithm. Observe that the runtime is dominated by Step 2, which in turn is dominated by the computation of persistent homology using PHAT.

Table 1. Running time (in seconds) of each algorithm step. Times in parentheses in Step 2 are those of PHAT.

	$\beta_0, \beta_1, \beta_2$	Cell count	**Step 1**	**Step 2**	**Step 3**
Tree	1,17,1	857327	1.32	6.30 (5.34)	0.86
Helmet	1,42,1	1598371	2.73	12.84 (11.48)	1.55
Bone	102,582,57	6028737	7.68	157.46 (133.10)	3.21
Vessel	138,11,0	8120601	17.51	102.32 (94.12)	0.96
Skull	31,358,6	9302799	21.21	131.48 (118.96)	4.61
Enzyme	1,456,48	13997521	33.81	196.60 (175.35)	8.53
Root	1040,754,37	34950215	83	1600.24 (1448.60)	6.72

We plot the runtime for the second batch test (on Thingi10K models) in Figure 17 right, showing also the time taken by Step 2 and specifically by PHAT. As in the visual examples, Step 2 (and particularly PHAT) dominates the overall runtime. The plot also reveals a roughly linear complexity to the size of the cell complex. All timings are recorded on a PC with Intel(R) Core(TM) i9-10900X CPU (3.70GHz) and 128GB RAM.

8 Discussion

We present a new algorithm for simplifying the topology of 3D shapes. A key benefit of our method is that it allows the user to *prescribe* the number of topological features of each type (components, handles, voids) to be preserved after simplification. This is enabled by a novel algorithm for filtration simplification that enables precise removal of persistent homological classes that are present at a specific cell complex of the filtration. Although success is not guaranteed, our method found maximal simplifications for all examples and target topologies in our experiments.

Our method has several limitations. We rely on the computation of persistent homology, which is known to be time- and space-consuming on large cell complexes (e.g., a high-resolution cubical grid). A possible solution is to apply our method to a partial filtration constructed from an "envelope" around the input shape, similar to [Zeng et al. 2020]. Furthermore, the cuts and fills made by the algorithm often have a non-smooth appearance, due in part to the structure of the underlying cell complex. This can be potentially resolved by performing grid refinement during simplification. As future work, we would like to explore how our algorithm can be extended to simplify all features in the filtration (for function simplification) and how it can be combined with surface reconstruction methods to produce topologically correct surfaces without the need for post-processing repair.

Acknowledgments

This work is supported in part by NSF grant HCC-2401224.

References

Dominique Attali, Ulrich Bauer, Olivier Devillers, Marc Glisse, and André Lieutier. 2015. Homological reconstruction and simplification in R3. *Computational Geometry* 48, 8 (2015), 606–621.

Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. 2017. Phat–persistent homology algorithms toolbox. *Journal of symbolic computation* 78 (2017), 76–90.

Ulrich Bauer, Carsten Lange, and Max Wardetzky. 2012. Optimal topological simplification of discrete functions on surfaces. *Discrete & Computational Geometry* 47, 2 (2012), 347–377.

Stephan Bischoff and Leif Kobbelt. 2002. Isosurface Reconstruction with Topology Control.. In *Pacific Conference on Computer Graphics and Applications*. 246–255.

P-T Bremer, Bernd Hamann, Herbert Edelsbrunner, and Valerio Pascucci. 2004. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics* 10, 4 (2004), 385–396.

Rickard Brüel-Gabrielsson, Vignesh Ganapathi-Subramanian, Primoz Skraba, and Leonidas J Guibas. 2020. Topology-Aware Surface Reconstruction for Point Clouds. In *Computer graphics forum*, Vol. 39. Wiley Online Library, 197–207.

Hamish Carr, Jack Snoeyink, and Michiel Van De Panne. 2010. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry* 43, 1 (2010), 42–58.

Mathieu Carriere, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hariprasad Kannan, and Yuhei Umeda. 2021. Optimizing persistent homology based functions. In *International conference on machine learning*. PMLR, 1294–1303.

Erin W Chambers, Tao Ju, David Letscher, Mao Li, and Christopher Topp. 2018. Some Heuristics for the Homological Simplification Problem. In *CCCG*.

Erin W Chambers, Tao Ju, David Letscher, Hannah Schreiber, and Dan Zeng. 2025. VHS: a package for homological simplification of voxelized plant root data for skeletonization. *Computational Geometry* (2025), 102198.

Lin Chen and Gudrun Wagenknecht. 2006. Automated topology correction for human brain segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 316–323.

David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. 2006. Vines and vineyards by updating persistence in linear time. In *Proceedings of the twenty-second annual symposium on Computational geometry*. 119–126.

Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. 2011. Dualities in persistent (co) homology. *Inverse Problems* 27, 12 (2011), 124003.

Lorenzo Diazzi, Daniele Panozzo, Amir Vaxman, and Marco Attene. 2023. Constrained delaunay tetrahedrization: A robust and practical approach. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–15.

Zhetong Dong, Jinhao Chen, and Hongwei Lin. 2022. Topology-controllable implicit surface reconstruction based on persistent homology. *Computer-Aided Design* 150 (2022), 103308.

Herbert Edelsbrunner, John Harer, et al. 2008. Persistent homology-a survey. *Contemporary mathematics* 453, 26 (2008), 257–282.

Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. 2002. Topological persistence and simplification. *Discrete & Computational Geometry* 28, 4 (2002).

Herbert Edelsbrunner, Dmitriy Morozov, and Valerio Pascucci. 2006. Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the twenty-second annual symposium on Computational geometry*. 127–134.

Herbert Edelsbrunner and Afra Zomorodian. 2003. Computing linking numbers of a filtration. (2003).

Andreas Fabri and Sylvain Pion. 2009. CGAL: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 538–539.

Depeng Gao, Jinhao Chen, Zhetong Dong, and Hongwei Lin. 2022. Connectivity-guaranteed porous synthesis in free form model by persistent homology. *Computers & Graphics* 106 (2022), 33–44.

Michael Garland and Paul S Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 209–216.

David Günther, Alec Jacobson, Jan Reininghaus, Hans-Peter Seidel, Olga Sorkine-Hornung, and Tino Weinkauf. 2014. Fast and memory-efficiently topological denoising of 2D and 3D scalar fields. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 2585–2594.

Xiao Han, Chenyang Xu, Ulisses Braga-Neto, and Jerry L. Prince. 2002. Topology Correction in Brain Cortex Segmentation Using a Multi-Scale, Graph-Based Algorithm. *IEEE Trans. Med. Imaging* 21, 2 (2002), 109–121.

Xiao Han, Chenyang Xu, and Jerry L. Prince. 2003. A topology preserving level set method for geometric deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 6 (2003), 755–768.

Allen Hatcher. 2002. *Algebraic Topology*. Cambridge University Pres.

Christian Heine, Heike Leitte, Mario Hlawitschka, Federico Iuricich, Leila De Floriani, Gerik Scheuermann, Hans Hagen, and Christoph Garth. 2016. A survey of topology-based methods in visualization. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 643–667.

Jiangbei Hu, Ben Fei, Baixin Xu, Fei Hou, Shengfa Wang, Na Lei, Weidong Yang, Chen Qian, and Ying He. 2025. TopoGen: Topology-Aware 3D Generation with Persistence Points. In *Computer Graphics Forum*, Vol. 44. Wiley Online Library, e70257.

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast tetrahedral meshing in the wild. *ACM Transactions on Graphics (ToG)* 39, 4 (2020), 117–1.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral meshing in the wild. *ACM Trans. Graph.* 37, 4 (2018), 60.

Zhiyang Huang, Ming Zou, Nathan Carr, and Tao Ju. 2017. Topology-controlled Reconstruction of Multi-labelled Domains from Cross-sections. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Tao Ju, Qian-Yi Zhou, and Shi-Min Hu. 2007. Editing the Topology of 3D Models by Sketching. *ACM Trans. Graph.* 26, 3, Article 42 (July 2007). doi:10.1145/1276377.1276430

Mohamed Kissi, Mathieu Pont, Joshua A Levine, and Julien Tierny. 2024. A Practical Solver for Scalar Data Topological Simplification. *IEEE Transactions on Visualization and Computer Graphics* (2024).

N. Kriegeskorte and R. Goebel. 2001. An efficient algorithm for topologically correct segmentation of the cortical sheet in anatomical mr volumes. *Neuroimage* 14, 2 (August 2001), 329–346.

Roee Lazar, Nadav Dym, Yam Kushinsky, Zhiyang Huang, Tao Ju, and Yaron Lipman. 2018. Robust optimization for topological surface reconstruction. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10.

Jonas Lukasczyk, Christoph Garth, Ross Maciejewski, and Julien Tierny. 2020. Localized topological simplification of scalar data. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 572–582.

Arnur Nigmetov and Dmitriy Morozov. 2024. Topological optimization with big steps. *Discrete & computational geometry* 72, 1 (2024), 310–344.

Fakir S. Nooruddin and Greg Turk. 2003. Simplification and Repair of Polygonal Models Using Volumetric Techniques. *IEEE Trans. Vis. Comput. Graph.* 9, 2 (2003), 191–205.

Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. 2017. A roadmap for the computation of persistent homology. *EPJ Data Science* 6, 1 (2017), 17.

Giuseppe Patanè and Bianca Falcidieno. 2009. Computing smooth approximations of scalar functions with constraints. *Computers & Graphics* 33, 3 (2009), 399–413.

Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. 2018. Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 13–25.

Florent Ségonne. 2008. Active contours under topology control - genus preserving level sets. *International Journal of Computer Vision* 79, 2 (2008), 107–117.

Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. 2006. Competing fronts for coarse–to–fine surface reconstruction. In *Eurographics*. Vienna, 389–398. http://www.mat.puc-rio.br/~tomlew/competing_fronts_eg.pdf

Andrei Sharf, Thomas Lewiner, Gil Shklarski, Sivan Toledo, and Daniel Cohen-Or. 2007. Interactive Topology-aware Surface Reconstruction. *ACM Trans. Graph.* 26, 3 (July 2007).

David W. Shattuck and Richard M. Leahy. 2001. Automated Graph Based Analysis and Correction of Cortical Volume Topology. *IEEE Trans. Med. Imaging* 20, 11 (2001), 1167–1177.

Hang Si and Klaus Gärtner. 2005. Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. In *Proceedings of the 14th international meshing roundtable*. Springer, 147–163.

Yitzchak Solomon, Alexander Wagner, and Paul Bendich. 2021. A fast and robust method for global topological functional optimization. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 109–117.

Andrzej Szymczak and James Vanderhyde. 2003. Extraction of topologically simple isosurfaces from volume datasets. In *IEEE Visualization*. 67–74.

Julien Tierny and Valerio Pascucci. 2012. Generalized topological simplification of scalar fields on surfaces. *IEEE transactions on visualization and computer graphics* 18, 12 (2012), 2005–2013.

Tino Weinkauf, Yotam Gingold, and Olga Sorkine. 2010. Topology-based smoothing of 2D scalar fields with C1-continuity. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 1221–1230.

Zoë J. Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. 2004. Removing excess topology from isosurfaces. *ACM Trans. Graph.* 23, 2 (2004), 190–208.

Pengxiang Wu, Chao Chen, Yusu Wang, Shaoting Zhang, Changhe Yuan, Zhen Qian, Dimitris Metaxas, and Leon Axel. 2017. Optimal topological cycles and their application in cardiac trabeculae restoration. In *International Conference on Information Processing in Medical Imaging*. Springer, 80–92.

Dong Xiao, Zuoqiang Shi, Siyu Li, Bailin Deng, and Bin Wang. 2023. Point normal orientation and surface reconstruction by incorporating isovalue constraints to Poisson equation. *Computer Aided Geometric Design* 103 (2023), 102195.

Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, Daniel Cohen-Or, and Baoquan Chen. 2014. Morfit: Interactive Surface Reconstruction from Incomplete Point Clouds with Curve-driven Topology and Geometry Control. *ACM Trans. Graph.* 33, 6 (Nov. 2014), 202:1–202:12.

Dan Zeng, Erin W Chambers, David Letscher, and Tao Ju. 2020. To cut or to fill: a global optimization approach to topological simplification. *ACM Trans. Graph.* 39, 6 (2020), 201–1.

Dan Zeng, Mao Li, Ni Jiang, Yiwen Ju, Hannah Schreiber, Erin Chambers, David Letscher, Tao Ju, and Christopher N Topp. 2021. TopoRoot: a method for computing hierarchy and fine-grained traits of maize roots from 3D imaging. *Plant methods* 17, 1 (2021),

127.
Yun Zeng, Dimitris Samaras, Wei Chen, and Qunsheng Peng. 2008. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in N-D images. *Computer Vision and Image Understanding* 112, 1 (2008), 81–90.
Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).
Qian-Yi Zhou, Tao Ju, and Shi-Min Hu. 2007. Topology Repair of Solid Models Using Skeletons. *IEEE Transactions on Visualization and Computer Graphics* 13, 4 (July 2007), 675–685.
Ming Zou, Michelle Holloway, Nathan Carr, and Tao Ju. 2015. Topology-constrained surface reconstruction from cross-sections. *ACM Trans. Graph.* 34, 4 (2015), 128.

A Proof of Proposition 5.1

We will first prove that migrating a set of disjoint cuts and fills produces a filtration. Recall that a cell ordering π and function f defines a filtration on the cell complex $\mathbb{X}$ if π is a filter (i.e., each cell is preceded by its faces) and f is non-decreasing in π. Please refer to Section 5 for the definitions of cuts, fills, and migration.

Lemma A.1. *Let C be a union of cuts and F a union of fills of a filtration $\{\mathbb{X}, \pi, f\}$ so that C and F are disjoint. The cell order π' and function f' after migrating $C \cup F$ defines a filtration on $\mathbb{X}$.*

Proof. It is easy to verify that f' is non-decreasing in π'. We will show that π' is a filter. By the definition of migration (see Figure 4), the order of a cell σ and its face ϕ in π may switch in π' after migration only in the following scenarios, which we will show to be impossible if C, F are disjoint cuts and fills:

(1) $\phi \in C$, $\sigma \notin C$ and $f(\sigma) < 0$ (i.e., ϕ moves ahead of σ^-, which is ahead of σ): C is made up of cuts, which are generating sets of the shape complex. By definition of generating sets, C must include the negative co-faces of all its cells. So $\phi \in C$ and $f(\sigma) < 0$ implies $\sigma \in C$.
(2) (Symmetric to (1)) $\sigma \in F$, $\phi \notin F$ and $f(\phi) > 0$ (i.e., σ moves to be before σ^+, which is before ϕ): Since F is made up of fills, which are the dual of generating sets of the dual complement of shape complex, F must include the positive faces of all its cells. So $\sigma \in F$ and $f(\phi) > 0$ implies $\phi \in F$.
(3) $\phi \in C$ and $\sigma \in F$ (i.e., they switch order as C and F switch): This is impossible if C and F are disjoint.

□

To understand how migrating $C \cup F$ affects the persistent homology, we shall break it down into smaller migrations, each moving only one or a pair of cells at a time. Each migration, in turn, is performed by a sequence of swaps of adjacent cells. Our analysis builds on the seminal work of [Cohen-Steiner et al. 2006], which details how to update persistent homology by cell swapping. We first summarize the results that are most relevant to our proof.

Consider two cells $a, b \in \mathbb{X}$ that are adjacent in the filter π such that $a < b$ (that is, a precedes b) and a is not a face of b. We assume that $\mathbb{X}$ has a trivial topology, meaning all p-pairs have a finite death cell except for the first cell in π, which is not a. Swapping a, b modifies the p-pairs and their (co-)generators only in the following scenarios (case numbers are from [Cohen-Steiner et al. 2006]):

(1) Both a, b are birth cells. Let $\{a, a'\}, \{b, b'\}$ be their p-pairs. Furthermore, a is on the generator of $\{b, b'\}$.
   (a) [Case 1.1.1] $a' < b'$: No changes to p-pairs, but the generator of $\{b, b'\}$ and the co-generator of $\{a, a'\}$ are modified.
   (b) [Case 1.1.2] $a' > b'$: The birth cells are switched to form new p-pairs $\{b, a'\}$ and $\{a, b'\}$. P-pair $\{b, a'\}$ inherits the co-generator, and modifies the generator, of $\{a, a'\}$, while p-pair $\{a, b'\}$ inherits the generator, and modifies the co-generator, of $\{b, b'\}$.
(2) Both a, b are death cells. Let $\{a^*, a\}, \{b^*, b\}$ be their p-pairs. Furthermore, b is on the co-generator of $\{a^*, a\}$.
   (a) [Case 2.1.1] $a^* < b^*$: No changes to p-pairs, but the generator of $\{b^*, b\}$ and the co-generator of $\{a^*, a\}$ are modified.
   (b) [Case 2.1.2] $a^* > b^*$: The death cells are switched to form new p-pairs $\{b^*, a\}$ and $\{a^*, b\}$. P-pair $\{b^*, a\}$ inherits the generator, and modifies the co-generator, of $\{b^*, b\}$, while p-pair $\{a^*, b\}$ inherits the co-generator, and modifies the generator, of $\{a^*, a\}$.
(3) [Case 3.1] a is a death cell and b is a birth cell. Let $\{a^*, a\}, \{b, b'\}$ be their p-pairs. Furthermore, a is on the generator of $\{b, b'\}$ and b is on the co-generator of $\{a^*, a\}$: The birth and death cells are switched to form new p-pairs $\{a^*, b\}$ and $\{a, b'\}$, which respectively inherit the (co-)generators of $\{a^*, a\}$ and $\{b, b'\}$.

These cases are illustrated in Figure 18. We shall refer to them using their respective case numbers in [Cohen-Steiner et al. 2006].

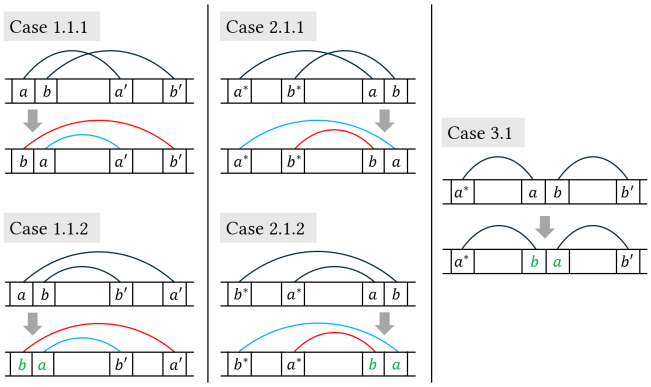


Fig. 18. Cases in [Cohen-Steiner et al. 2006] in which swapping a cell pair $\{a, b\}$ modifies some p-pairs (green letters), generators (red arcs), or co-generators (blue arcs).

We start with a technical lemma about birth and death cells. Recall that a cell $\sigma \in X$ is *isolated* in cell complex X if it has no co-face in X, and it forms a *simple pair* with a face ϕ if σ is the only co-face of ϕ in X. We call σ *simple* in X if it can form a simple pair with one of its faces. Also recall that the complement of a sub-complex $X \in \mathbb{X}$ is $\bar{X} = \mathbb{X} \setminus X$, and the dual complement is $\tilde{X} = \hat{\mathbb{X}} \setminus \hat{X}$, where $\hat{}$ is the dual operator.

Lemma A.2. *For any cell complex X in a filtration:*

(1) *Any simple cell in X cannot be a birth cell in the filtration. Symmetrically, the dual of any simple cell in $\tilde{X}$ cannot be a death cell.*
(2) *If a birth cell is isolated in X, then its paired death cell is in $\bar{X}$. Symmetrically, if the dual of a death cell is isolated in $\tilde{X}$, then the paired birth cell is in X.*

Proof. We only need to prove the first part of each statement; the second part holds by the duality between persistent homology and persistent relative co-homology (see Section 3.1).

(1) Suppose, on the contrary, that a k-cell b is simple in X and also the birth cell in a p-pair $\{b, d\}$. Then b lies in the generator of $\{b, d\}$, which is a k-cycle, and b is the last k-cell in the cycle to enter the filtration. As a result, this k-cycle is a subset of any cell complex in the filtration (such as X) that contains b. On the other hand, any k-chain in X containing b has an "open" boundary at the face of b that it forms a simple pair with, contradicting that b is in a k-cycle.
(2) Consider a k-cell b that is isolated in X and the birth cell in a p-pair $\{b, d\}$. Following the argument above, b lies in a k-cycle in X. If, on the contrary, that $d \in X$, then the k-cycle must be a k-boundary in X; that is, b must be a face of some $(k+1)$-cell in X. This contradicts the fact that b is isolated.

□

Let us first consider migrating a single cell. The following lemma shows that, when the cell is an isolated birth cell of a minor p-pair, migration removes exactly one minor p-pair.

Lemma A.3. *Let b be the birth cell in a minor p-pair of a filtration $\{\mathbb{X}, \pi, f\}$. If b is isolated in the shape complex $\mathbb{X}_f$, then migrating b results in a filtration that has the same set of major p-pairs as $\{\mathbb{X}, \pi, f\}$ and one fewer minor p-pair.*

Proof. We will decompose the migration into a sequence of steps. Except for the last step, each step swaps b with its current next cell and adjusts its time to be the same as the cell it swaps with. The steps are repeated until b is swapped with σ^-, the last negative cell in π. The final step raises the time of b to a positive value smaller than $f(\sigma^+)$. Since b is isolated in $\mathbb{X}_f$, all its co-faces have positive time, and hence each step produces a valid filtration.

If b maintains the same paired death cell as it moves forward, the persistence of its p-pair never increases, and the p-pair remains minor. Since b is a birth cell, swapping with another cell c makes changes to p-pairs only in Case 2.1.2, where c is another birth cell and the p-pair of c has a lower persistence than the p-pair of b. Furthermore, by Lemma A.2 (2), b must be paired with a death cell outside the shape complex (i.e., with a positive time). As a result, c must be the birth cell of a minor p-pair, and swapping b with c does not change any major p-pair or the total number of minor p-pairs. The final step retains all p-pairs but only makes the p-pair of b inactive, thus reducing the number of minor p-pairs by one. □

As a corollary, we have the following characterization of the p-pair of b after migration, which follows directly from the proof above and Lemma A.2 (2):

Corollary A.4. *Let b be the birth cell in a minor p-pair of a filtration $\{\mathbb{X}, \pi, f\}$ that is isolated in $\mathbb{X}_f$, and $\{\pi', f'\}$ the new filter and time function after migrating b. Then:*

(1) *The birth cells of minor p-pairs of the filtration $\{\mathbb{X}, \pi', f'\}$ are the same as those of the filtration $\{\mathbb{X}, \pi, f\}$ but without b.*
(2) *The death cells of minor p-pairs of the filtration $\{\mathbb{X}, \pi', f'\}$ are the same as those of the filtration $\{\mathbb{X}, \pi, f\}$ but without d, such that $\{b, d\}$ is a p-pair in the filtration $\{\mathbb{X}, \pi', f'\}$ and d is either a co-face of b or the dual of a non-isolated cell in $\tilde{\mathbb{X}}_{f'}$.*

Moving on to migrating a pair of cells, the next lemma shows that migrating a simple pair, under certain conditions, retains all p-pairs and their (co-)generators.

Lemma A.5. *Let $\{\phi, \sigma\}$ be a simple pair of the shape complex $\mathbb{X}_f$ of a filtration $\{\mathbb{X}, \pi, f\}$ such that neither cell lies in the generator of any active p-pair. Migrating $\{\phi, \sigma\}$ results in a filtration that retains all active p-pairs and their (co-)generators in $\{\mathbb{X}, \pi, f\}$.*

Proof. We follow the multi-step approach in the proof of Lemma A.3 to migrate one cell at a time. We first migrate the simple cell σ. Since σ is isolated, each step of migration produces a valid filtration. Note that σ remains simple in the shape complex, until the final step when it exits the shape complex. By Lemma A.2 (1), σ must be a death cell and remains so during migration. As a result, before the final step, migration makes changes to p-pairs and (co-)generators only in Cases 2.1.1 and 2.1.2. In both cases, σ swaps with another death cell and the swap does not affect any active p-pairs or their (co-)generators. The final step creates a new active p-pair – the p-pair with σ as the death cell – as σ's time turns positive. Let π', f' be the new filter and time function.

Next, we migrate ϕ, which is now isolated in the new shape complex $\mathbb{X}_{f'}$, since its only co-face (σ) is no longer in the shape complex. As a result, each step of migration produces a valid filtration. By Lemma A.2 (2), an isolated ϕ cannot be the birth cell of a non-active p-pair. On the other hand, since ϕ avoids all generators of active p-pairs of the original filtration, which are preserved by the migration of σ, ϕ cannot be the birth cell of any of those active p-pairs. We conclude that, before migration, ϕ is either a death cell or the birth cell paired with σ. We will show that each step of migration (before the final step) maintains this characterization of ϕ, as well as all active p-pairs (except σ's) and their (co-)generators. There are two cases to consider when swapping ϕ with its next cell ϕ':

(1) ϕ is a death cell: Cases 2.1.1, 2.1.2 and 3.1 are possible. In the first two cases, only non-active p-pairs and (co-)generators are affected, and ϕ remains a death cell. In the last case, ϕ becomes a birth cell. Following the same argument above, ϕ can only be paired with σ, and no other active p-pairs or their (co-)generators are affected.
(2) $\{\phi, \sigma\}$ is a p-pair: Swapping ϕ, ϕ' may change p-pairs or their (co-)generators only in Cases 1.1.1 and 1.1.2, and we will show that neither case would happen. The former requires ϕ to be on the generator of the p-pair of ϕ', which is an active p-pair inherited from the original filtration. By our assumption, such generators do not contain ϕ. The latter, if it happens, would make ϕ the birth cell of a non-active pair, which is not possible as ϕ remains isolated during migration.

Note that $\{\phi, \sigma\}$ must be a p-pair before the final step (when ϕ's time turns positive). To see this, note that the migrated $\{\phi, \sigma\}$ are consecutive in the filter. Therefore, ϕ is the last face of σ to enter the filtration, and σ is the first co-face of ϕ to enter the filtration. In the algorithm for computing persistent homology classes, the column of σ with pivot ϕ is already reduced in the boundary matrix, making $\{\phi, \sigma\}$ a p-pair. This p-pair turns inactive after the final step,

removing the only new active p-pair introduced by migrating σ. As a result, the resulting filtration retains all original active p-pairs and their (co-)generators. □

This lemma is the reason why our algorithm asks the skeleton to preserve the generators of active p-pairs: doing so ensures that the simple pairs removed during skeletonization avoid the generators. As the removed simple pairs make up the generating sets, which define the cuts, we can leverage this lemma to examine the migration of simple pairs in the cuts. Similarly, preserving the co-generators of active p-pairs in the background skeleton allows us to utilize the lemma, applied to the background filtration $\{\hat{\mathbb{X}}, -\pi, -f\}$, to examine the migration of simple pairs in the fills.

We are now ready to prove the rest of Proposition 5.1.

Lemma A.6. *Let $\{C, F\}$ be unions of m disjoint cuts and fills of a filtration $\{\mathbb{X}, \pi, f\}$. The new filtration after migrating $C \cup F$ has the same set of major p-pairs as $\{\mathbb{X}, \pi, f\}$ and m fewer minor p-pairs.*

Proof. Let B be the set of birth cells whose cuts make up C, and D the set of death cells whose fills make up F. We will migrate $C \cup F$ in several stages, as detailed below, while tracking the changes to the p-pairs and their (co-)generators.

We first migrate $C \setminus B$. By the definition of generating sets, cells in $C \setminus B$ can be grouped into an ordered list of pairs Π such that the i-th pair is a simple pair in the shape complex after removing the first $i-1$ pairs. Furthermore, as discussed above, all pairs in Π avoid the generators of active p-pairs. Therefore, by Lemma A.3, sequentially migrating the pairs in Π produces a filtration that leaves all active p-pairs and their (co-)generators unchanged.

Next, we migrate $F \setminus D$ in a symmetric way. We group cells in $F \setminus D$ into an ordered list of pairs Π such that the dual of the i-th pair is a simple pair in the dual complement after adding the first $i-1$ pairs to the shape complex. Furthermore, all pairs in Π avoid the co-generators of active p-pairs. By applying Lemma A.3 to the background filtration, we see that sequentially migrating the pairs in Π produces a filtration that retains all active p-pairs and their (co-)generators.

We then migrate B. Since the active p-pairs are unchanged after the previous migrations, cells in B remain birth cells of minor p-pairs. Furthermore, each cell in B is now isolated in the shape complex, as its generating set has been removed. By Lemma A.3, migrating a cell $b \in B$ produces a filtration that keeps all major p-pairs and reduces the number of minor p-pairs by one. By Corollary A.4 (1), the remaining cells in $B \setminus \{b\}$ are still birth cells of the minor p-pairs, and they remain isolated. As a result, migrating B, one cell at a time, produces a filtration that retains all major p-pairs and removes exactly $|B|$ minor p-pairs.

Next, we migrate D in a similar way, one cell at a time. Note that the dual cells $\hat{D}$ are isolated in the dual complement of the shape complex before the migration of B. Furthermore, no cell in D is a co-face of a cell in B, because C and F are disjoint. By Corollary A.4 (2), migrating B does not affect D, meaning that each cell $d \in D$ remains a death cell of a minor p-pair and $\hat{d}$ is isolated in the dual complement. Applying Lemma A.3 to the background filtration shows that the migrating D results in a filtration that retains all major p-pairs and removes exactly $|D|$ minor p-pairs.

At this point, the cell groups C and F are migrated to the desired position in the filter, such that $\sigma^- < F < C < \sigma^+$. Let the current filter be π'. However, the relative order within each group is not the same as in the original filter π. In the final step, we rearrange the cells within C and F to recover their relative order in π, via adjacent cell swaps. Since both π, π' are filters, a sequence of cell swaps that maintains the filter property always exists. Furthermore, Corollary A.4 and the argument within the proof of Lemma A.5 show that cells in C and F are only involved in inactive p-pairs. As a result, cell swapping within C (or F) will not affect any active p-pairs, based on the case analysis earlier. Therefore, rearrangement produces a filtration whose active p-pairs are exactly what they were before rearrangement, which has the same set of major p-pairs as the original filtration and $|B| + |D| = m$ fewer minor p-pairs. □

B Proof of Proposition 5.2

Proof. We will first show that a min cut E^* of H^* is also a min cut of H, which would imply that V_{E^*} is an MIS of G. Denote by $s_H(E)$ the size of a cut E in graph H, defined as the sum of edge capacity in H over all edges in E.

Suppose, on the contrary, that E^* is not a min cut of H. As H^* shares the same vertices and edges with H, E^* is also a cut in H. Then there must exist another cut E of H such that $s_H(E) < s_H(E^*)$. Note that a cut with finite size exists in H^* (e.g., the set of all edges incident to s or t), and so E^* must consist of only edges with finite capacity in H^*, implying it (and E) consists of only edges with finite capacity in H. Since the only finite capacity in H is 1, we have $s_H(E) = |E|$ and $s_H(E^*) = |E^*|$, and hence

$$|E| \le |E^*| - 1.$$

On the other hand, E is also a cut in H^*. The cut sizes of E and E^* in H^* are:

$$s_{H^*}(E) = |E| * W - w(U_E) \text{ and } s_{H^*}(E^*) = |E^*| * W - w(U_{E^*}),$$

where $w(V) = \sum_{v \in V} w(v)$ for any node set V, and U_E consists of all vertices in V_c disconnected from s and vertices in V_f disconnected from t by the cut E. Note that both U_E and U_{E^*} are non-empty (otherwise E or E^* is not an s-t cut) and are subsets of $V_c \cup V_f$. So we derive

$$s_{H^*}(E) - s_{H^*}(E^*) \le w(U_{E^*}) - w(U_E) - W < w(U_{E^*}) - W \le 0,$$

meaning E has a smaller cut size in H^* than E^*. This contradicts the assumption that E^* is a min cut of H^*.

We next show that, among all min cuts of H (which correspond to MIS's of G), the node set V_{E^*} defined by the min cut E^* has the smallest total cost. Note that, for any cut E of H,

$$w(V_E) = W - w(U_E) = s_{H^*}(E) - (|E| - 1) * W.$$

The arguments above showed that all min cuts of H must contain the same number of edges. Therefore, the min cut E that minimizes $w(V_E)$ is the one that minimizes $s_{H^*}(E)$, its cut size in H^*, implying that E^* is such a min cut. □