A Practical Solver for Scalar Data Topological Simplification

Mohamed Kissi, Mathieu Pont, Joshua A. Levine, Julien Tierny

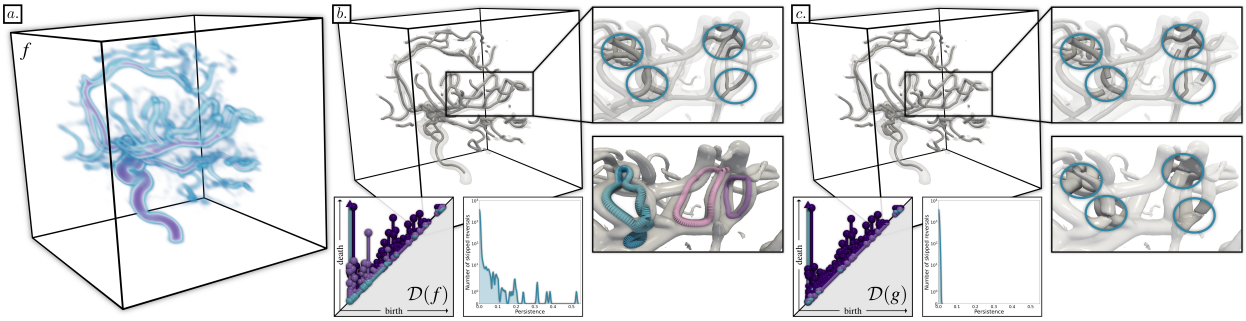

Fig. 1: Given an acquired scalar field f of a network of arteries (a), the core structure of the blood vessels can be extracted (grey filaments, (b)) with upward discrete integral lines, started at 2-saddles above 0.1 (isovalue representing the geometry of the vessels, transparent isosurface). However, as shown in the persistence diagram $\mathcal{D}(f)$, f contains many saddle-pairs (light purple bars), corresponding to persistent 1-dimensional generators [38, 53] (curves, colored by persistence, bottom zoom, (b)), yielding incorrect loops in the filament structures (top zoom). In this example, standard techniques for gradient field simplification (i.e., saddle connector reversal) cannot simplify these spurious loops while maintaining a valid gradient (b), as shown in the bottom histogram (number of *skipped* reversals as a function of persistence). Our approach efficiently generates a function g which is close to f and which optimizes saddle pair cancellation while maintaining the other features ($\mathcal{D}(g)$). This enables the direct visualization and analysis of the simplified data (c), where isosurface handles have been cut (bottom zoom) and most spurious filament loops have been simplified (top zoom).

Abstract—This paper presents a practical approach for the optimization of topological simplification, a central pre-processing step for the analysis and visualization of scalar data. Given an input scalar field f and a set of *"signal"* persistence pairs to maintain, our approaches produces an output field g that is close to f and which optimizes *(i)* the cancellation of *"non-signal"* pairs, while *(ii)* preserving the *"signal"* pairs. In contrast to pre-existing simplification algorithms, our approach is not restricted to persistence pairs involving extrema and can thus address a larger class of topological features, in particular saddle pairs in three-dimensional scalar data. Our approach leverages recent generic persistence optimization frameworks and extends them with tailored accelerations specific to the problem of topological simplification. Extensive experiments report substantial accelerations over these frameworks, thereby making topological simplification optimization practical for real-life datasets. Our approach enables a direct visualization and analysis of the topologically simplified data, e.g., via isosurfaces of simplified topology (fewer components and handles). We apply our approach to the extraction of prominent filament structures in three-dimensional data. Specifically, we show that our pre-simplification of the data leads to practical improvements over standard topological techniques for removing filament loops. We also show how our approach can be used to repair genus defects in surface processing. Finally, we provide a C++ implementation for reproducibility purposes.

Index Terms—Topological Data Analysis, scalar data, simplification, feature extraction.

1 Introduction

As acquisition devices and computational resources are getting more sophisticated and efficient, modern datasets are growing in size. Consequently, the features of interest contained in these datasets gain in geometric complexity, which challenge their interpretation and analysis. This motivates the design of tools capable of robustly extracting the structural patterns hidden in complex datasets. This task is the purpose of Topological Data Analysis (TDA) [27, 93], which provides a family of techniques for the generic, robust and multi-scale extraction of structural features. It has been successfully applied in a number of data analysis problems [50], in various applications, including turbulent combustion [17, 42], material sciences [47, 82], nuclear energy [63], fluid dynamics [55, 67], bioimaging [3, 15], quantum chemistry [12, 71, 72], or astrophysics [80, 84]. TDA provides a variety of *topological data abstractions*, which enable the extraction of specific types of features of interest. These abstractions include critical points [6], persistence diagrams [8, 28, 38], merge [20, 35, 60] and contour trees [19, 36], Reeb graphs [13, 37, 73], or Morse-Smale complexes [44, 45, 62, 78, 79]. A central aspect of TDA is its ability to analyze data at multiple scales. Thanks to various importance measure [21, 28], these abstractions can be iteratively simplified, to reveal the prominent structures in a dataset.

In practice, this topological simplification can be achieved in two fashions: either by *(i)* a post-process simplification of the abstractions, or by *(ii)* a pre-process simplification of the data itself. While the post-process approach *(i)* requires specific simplification mechanisms tailored to the abstraction at hand [21, 41, 73], the pre-process strategy offers a generic framework which is independent of the considered abstraction. This generic aspect eases software design, as simplification needs to be implemented only once [14, 85]. Pre-process simplification has also the advantage of being reusable by multiple abstractions when these are combined within a single data analysis pipeline (see [88] for real-life examples). Also, pre-process simplification enables the direct visualization of the simplified data itself (e.g. with isosurfaces). Finally, it is also compatible with further post-process simplification if needed. For these reasons, we focus on pre-process simplification in this work.

Several combinatorial approaches [2, 5, 10, 29, 59, 81, 86] have been proposed for the pre-simplification of persistence pairs involving ex-

- *Mohamed Kissi, Mathieu Pont, and Julien Tierny are with the CNRS and Sorbonne University. E-mail: {firstname.lastname}@sorbonne-universite.fr*
- *Joshua A. Levine is with the University of Arizona. E-mail: josh@cs.arizona.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

trema. However, no efficient combinatorial algorithm has been proposed for the pre-simplification of saddle pairs, hence preventing a more advanced simplification of 3D datasets. In fact, as pointed out by Chambers et al. [24], *optimal simplification* (i.e. finding a scalar field g which is δ-away from an input field f, such that g has a minimum number of critical points [10]) is a more general, hence more difficult, version of the *sublevel set simplification* problem, itself being NP-hard in 3D [4]. Then, there may not even exist polynomial time algorithms for directly solving this problem. This theoretical limitation requires a shift in strategy. A recent alternative consists in considering persistence optimization frameworks [22, 70, 83], which optimize the data in a *best effort* manner, given criteria expressed with persistence diagrams. However, while one could leverage these frameworks for data pre-simplification (i.e., to cancel noisy features while preserving the features of interest, *as much as possible*), current frameworks can require up to days of computation for regular grids of standard size (Sec. 3), making them impractical for real-life datasets.

This paper addresses this issue and introduces a practical solver for the optimization of the topological simplification of scalar data. Our approach relies on a number of pragmatic observations from which we derived specific accelerations, for each sub-step of the optimization (Secs. 4.2 and 4.3). Our accelerations are simple and easy to implement, but result in significant gains in terms of runtimes. Extensive experiments (Sec. 5.1) report $\times 60$ accelerations on average over state-of-the-art frameworks (with both fewer and faster iterations), thereby making topological simplification optimization practical for real-life datasets. We illustrate the utility of our contributions in two applications. First, our work enables the direct visualization and analysis of topologically simplified data (Sec. 5.2). This reduces visual clutter in isosurfaces by simplifying their topology (fewer components and handles). We also investigate filament extraction in three-dimensional data, where we show that our approach helps standard topological techniques for removing filament loops. Second, we show how to use our approach to repair genus defects in surface processing (Sec. 5.3).

1.1 Related work

Beyond post-process simplification schemes tailored for specific topological abstractions (e.g. for merge/contour trees [18], Reeb graphs [73] or Morse-Smale complexes [40, 41, 43]), the literature related to pre-process simplification can be classified into two categories.

Combinatorial methods: the first combinatorial approach for the topological simplification of scalar data on surfaces has been proposed by Edelsbrunner et al. [29]. This work can be seen as a generalization of previous approaches in terrain modeling where only persistence pairs involving minima were removed [2, 81]. Attali et al. [5] extended this framework to generic filtrations, while Bauer et al. [10] extended it to discrete Morse theory [31]. Tierny et al. presented a generalized approach [86], supporting a variety of simplification criteria, which was later extended by Lukasczyk et al. [59] with an efficient shared-memory parallel algorithm. Such combinatorial simplification algorithms can be used directly within optimization procedures [69], to remove noise in the solution at each iteration. While most of the above approaches were specifically designed for scalar data on surfaces, they can be directly applied to domains of higher dimensions. However, they can only simplify persistence pairs involving extrema. For instance, this means that they cannot remove saddle pairs in three-dimensional scalar fields, thus preventing an advanced simplification of this type of datasets. *Optimal simplification* [10] of scalar data is a more general variant of the *sublevel set simplification* problem, itself being NP-hard in 3D [4]. Then, a polynomial time algorithm solving this problem may not even exist. This theoretical limitation requires a shift in strategy.

Numerical methods: in contrast to combinatorial methods, which come with strong guarantees on the result, numerical approaches aim at providing an approximate solution in a best effort manner. In other words, these methods may not *fully* simplify three-dimensional scalar fields up to the desired tolerance either, but they will do their best to provide a result as close as possible to the specified simplification. As such, this type of approaches appear as a practical alternative overcoming the theoretical limitation of combinatorial approaches discussed above. In geometric modeling, several techniques have been described to generate smooth scalar fields on surfaces, with a minimal number of critical points [34, 68, 75]. Bremer et al. [16] proposed a method based on Laplacian smoothing to reconstruct a two-dimensional scalar field corresponding to a pre-simplified Morse-Smale complex. This work has been extended by Weinkauf et al. [90] to bi-Laplacian optimization, with an additional enforcement of gradient continuity across the separatrices of the Morse-Smale complex. While an extension of this work has been documented for the 3D case [39], it only addresses the simplification of persistence pairs involving extrema, without explicit control on the saddle pairs. Recently, a new class of methods dedicated to *persistence optimization* has been documented. Specifically, these approaches introduce a framework for optimizing a dataset, according to criteria expressed with persistence diagrams, with applications in various tasks including surface matching [76], point cloud processing [33], classification [22] and more. Solomon et al. [83] presented an approach based on stochastic subsampling applied to 2D images. Carriere et al. [22] presented an efficient and generic persistence optimization framework, supporting a wide range of criteria and applications, exploiting the convergence properties of stochastic sub-gradient descent [57] for tame functions [26]. Nigmetov et al. [70] presented an alternative method, drastically reducing the number of optimization iterations, but at the cost of significantly more computationally expensive steps. As described in Sec. 3, one can leverage these frameworks for the problem of topological simplification, however, with impractical runtimes for three-dimensional datasets of standard size (e.g. up to days of computation). We address this issue in this work by proposing a practical approach for topological simplification optimization, with substantial accelerations over state-of-the-art frameworks for persistence optimization [22].

1.2 Contributions

This paper makes the following new contributions:

1. **Algorithm:** We introduce a practical solver for the optimization of topological simplification for scalar data (Sec. 4). Our algorithm is based on two accelerations, which are tailored to the specific problem of topological simplification:
   - We present a simple and practical procedure for the fast update of the persistence diagram of the data along the optimization (Sec. 4.2), hence preventing a full re-computation at each step.
   - We describe a simple and practical procedure for the fast update of the pair assignments between the diagram specified as target, and the persistence diagram of the optimized data (Sec. 4.3), also preventing a full re-computation at each step.

   Overall, the combination of these accelerations makes topological simplification optimization tractable for real-life datasets.
2. **Applications:** Thanks to its practical time performance, our work sets up ready-to-use foundations for several concrete applications:
   - *Visualization of topologically simplified data (Sec. 5.2):* we illustrate the utility of our framework for the direct visualization and analysis of topologically simplified data. Our approach reduces visual clutter in isosurfaces by simplifying connected components as well as, in contrast to previous work, surface handles. We also investigate prominent filament extraction in 3D data, where we show that our approach helps standard topological techniques for removing filament loops.
   - *Surface genus repair (Sec. 5.3):* we show how to use our framework to repair genus defects in surface processing, with an explicit control on the employed primitives (cutting or filling).
3. **Implementation:** We provide a C++ implementation of our algorithm that can be used for reproducibility purposes.

2 Preliminaries

This section presents the background to our work. We refer the reader to textbooks [27, 93] for introductions to computational topology.

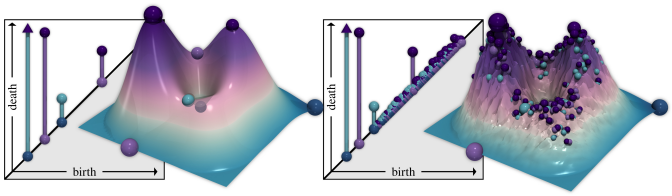

Fig. 2: Persistent diagrams for the lexicographic filtration of a clean (left) and a noisy (right) terrain example. Minimum-saddle persistence pairs are show with cyan bars in the birth-death space, while saddle-maximum pairs are shown with purple bars. Generators with infinite persistence are marked with an upward arrow. The persistence of each topological feature is given by the height of its bar. Critical simplices are shown in the data with spheres, with a radius proportional to their persistence.

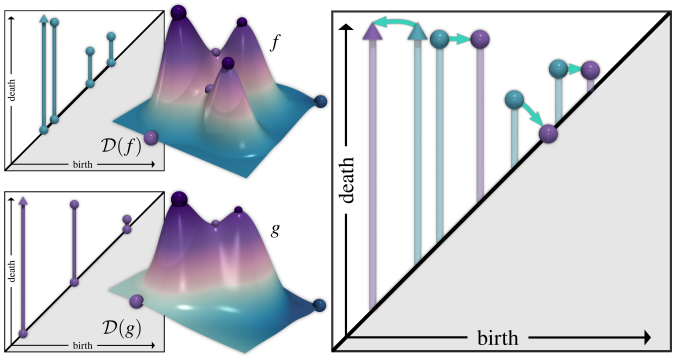

Fig. 3: The Wasserstein distance $\mathcal{W}_2$ between $\mathcal{D}(f)$ (top) and $\mathcal{D}(g)$ (bottom) is computed by assignment optimization (Eq. 1) in the 2D birth-death space (right). The optimal assignment ϕ^* (arrows) encodes a minimum cost transformation of $\mathcal{D}(f)$ into $\mathcal{D}(g)$, which displaces persistence pairs in the birth-death space or *cancel* them by sending them to the diagonal.

2.1 Input data

The input data is provided as a piecewise-linear (PL) scalar field $f : \mathcal{K} \rightarrow \mathbb{R}$ defined on a d-dimensional simplicial complex $\mathcal{K}$ (with $d \leq 3$ in our applications). If the data is provided on a regular grid, we consider for $\mathcal{K}$ the implicit Freudenthal triangulation of the grid [49, 52]. In practice, the data values are defined on the n_v vertices of $\mathcal{K}$, in the form of a *data vector*, noted $v_f \in \mathbb{R}^{n_v}$. f is assumed to be injective on the vertices (i.e., the entries of v_f are all distinct), which can be easily obtained in practice via a variant of simulation of simplicity [30].

2.2 Persistence diagrams

Persistent homology has been developed independently by several research groups [7, 28, 32, 77]. Intuitively, persistent homology considers a sweep of the data (i.e., a *filtration*) and estimates at each step the corresponding topological features (i.e., *homology generators*), as well as maps to the features of the previous step. This enables the identification of the topological features, along with their lifespan, during the sweep.

In this work we consider the *lexicographic filtration* (as described in [38]), which we briefly recall here for completeness. Given the input data vector $v_f \in \mathbb{R}^{n_v}$, one can sort the vertices of $\mathcal{K}$ by increasing data values, yielding a *global vertex order*. Based on this order, each d'-simplex $\sigma \in \mathcal{K}$ (with $d' \in [0, d]$) can be represented by the sorted list (in decreasing values) of the $(d'+1)$ indices in the global vertex order of its $(d'+1)$ vertices. Given this simplex representation, one can now compare two simplices σ_i and σ_j via simple lexicographic comparison, which induces a *global lexicographic order* on the simplices of $\mathcal{K}$. This order induces a nested sequence of simplicial complexes $\emptyset = \mathcal{K}_0 \subset \mathcal{K}_1 \subset \ldots \subset \mathcal{K}_{n_\sigma} = \mathcal{K}$ (where n_σ is the number of simplices of $\mathcal{K}$), which we call the *lexicographic filtration* of $\mathcal{K}$ by f.

At each step i of the filtration, one can characterize the p^{th} homology group of $\mathcal{K}_i$, noted $\mathcal{H}_p(\mathcal{K}_i)$, for instance by counting its number of *homology classes* [27, 38] (i.e., the *order* of the group) or its number of *homology generators* (i.e., the *rank* of the group, a.k.a. the p^{th} *Betti number*, noted β_p). Intuitively, in 3D, the first three Betti numbers (β_0, β_1, and β_2) respectively provide the number of connected components, of independent cycles and voids of the complex $\mathcal{K}_i$. For two consecutive steps of the filtration i and j, the corresponding simplicial complexes are nested ($\mathcal{K}_i \subset \mathcal{K}_j$). This inclusion induces homomorphims between the homology groups $\mathcal{H}_p(\mathcal{K}_i)$ and $\mathcal{H}_p(\mathcal{K}_j)$, mapping homology classes at step i to homology classes at step j. Intuitively, for the 0^{th} homology group, one can precisely map a connected component at step i to a connected component at step j because the former is included in the latter. In general, a p-dimensional homology class γ_i at step i can be mapped to a class γ_j at step j if the p-cycles of γ_i and γ_j are *homologous* in $\mathcal{K}_j$ [27, 38]. Then, one can precisely track the homology generators between consecutive steps of the filtration. In particular, a *persistent generator* is born at step j (with $j = i+1$) if it is not the image of any generator by the homomorphims mapping $\mathcal{H}_p(\mathcal{K}_i)$ to $\mathcal{H}_p(\mathcal{K}_j)$. Symmetrically, a persistent generator dies at step j if it merges with another, *older* homology class, which was born before it (this is sometimes called the *Elder rule* [27]). Each p-dimensional persistent generator is associated to a *persistence pair* (σ_b, σ_d), where σ_b is the p-simplex introduced at the *birth* of the generator (at step b) and where σ_d is the $(p+1)$-simplex introduced at its *death* (at step d). A p-simplex which is involved in the birth or the death of a generator is called a *critical simplex* and, in 3D, we call it a minimum, a 1-saddle, a 2-saddle, or a maximum if p equals 0, 1, 2, or 3 [38, 78], respectively. The *persistence* of the pair (σ_b, σ_d), noted $p(\sigma_b, \sigma_d)$, is given by $p(\sigma_b, \sigma_d) = v_f(v_d) - v_f(v_b)$, where v_b and v_d (the *birth* and *death vertices* of the pair) are the vertices with highest global vertex order of σ_b and σ_d. We call *zero-persistence pairs* the pairs with $v_b = v_d$. Some p-simplices of $\mathcal{K}$ may be involved in no persistence pair. These mark the birth of persistent generators with *infinite persistence* (i.e., which never die during the filtration) and they characterize the homology groups of the final step of the filtration ($\mathcal{K}_{n_\sigma} = \mathcal{K}$).

The set of persistence pairs induced by the lexicographic filtration of $\mathcal{K}$ by f can be organized in a concise representation called the *persistence diagram* (Fig. 2), noted $\mathcal{D}(f)$, which embeds each non zero-persistence pair (σ_b, σ_d) as a point in a 2D space (called the birth-death space), at coordinates $\big(v_f(v_b), v_f(v_d)\big)$. By convention, generators with infinite persistence are reported at coordinates $\big(v_f(v_b), v_f(v_{max})\big)$, where v_{max} is the last vertex in the global vertex order.

2.3 Wasserstein distance between persistence diagrams

Two diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$ can be reliably compared in practice with the notion of *Wasserstein distance*. For this, the two diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$ need to undergo an *augmentation* pre-processing phase. This step ensures that the two diagrams admit the same number of points, which will facilitate their comparison. Given a point $p = (p_b, p_d) \in \mathcal{D}(f)$, we note $\Delta(p)$ its diagonal projection: $\Delta(p) = \big(\frac{1}{2}(p_b + p_d), \frac{1}{2}(p_b + p_d)\big)$. Let Δ_f and Δ_g be the sets of the diagonal projections of the points of $\mathcal{D}(f)$ and $\mathcal{D}(g)$ respectively. Then, $\mathcal{D}(f)$ and $\mathcal{D}(g)$ are *augmented* by appending to them the set of diagonal points Δ_g and Δ_f respectively. After this augmentation, we have $|\mathcal{D}(f)| = |\mathcal{D}(g)|$.

Then, given two augmented persistence diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$, the L^q Wasserstein distance between them is defined as:

$$\mathcal{W}_q\big(\mathcal{D}(f), \mathcal{D}(g)\big) = \min_{\phi \in \Phi} \Big(\sum_{p \in \mathcal{D}(f)} c\big(p, \phi(p)\big)^q \Big)^{\frac{1}{q}}, \tag{1}$$

where Φ is the set of all bijective maps between the augmented diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$, which specifically map points of finite (respectively infinite) r-dimensional persistent generators to points of finite (respectively infinite) r-dimensional persistent generators. For this distance, the cost $c(p, p')$ is set to zero when both p and p' lie on the diagonal (i.e., matching dummy features has no impact on the distance). Otherwise, it is set to the Euclidean distance in birth-death space $||p - p'||_2$.

The Wasserstein distance induces an optimal assignment ϕ^* from $\mathcal{D}(f)$ to $\mathcal{D}(g)$ (Fig. 3), which depicts how to minimally transform $\mathcal{D}(f)$ into $\mathcal{D}(g)$ (given the considered cost). This transformation may induce

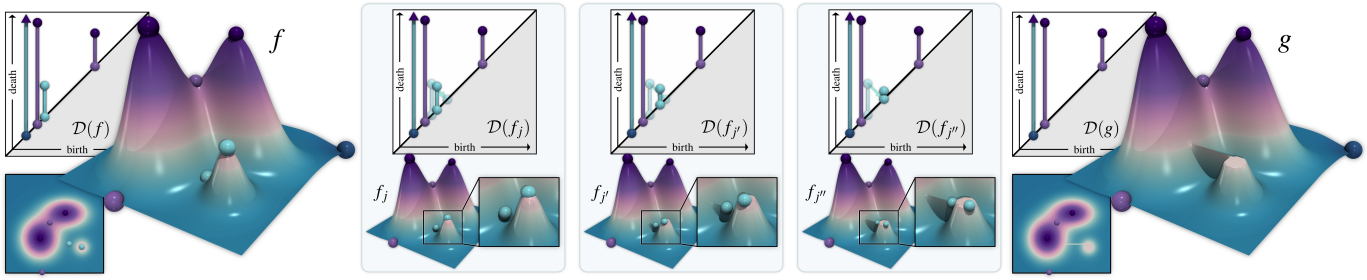

Fig. 4: Optimizing the simplification of an input scalar field $f = f_0$ into a field $g = f_{final}$ for the removal of a user selected saddle-maximum pair (cyan). At each iteration ($j < j' < j''$), given the point $p_i \in \mathcal{D}(f_j)$ to cancel, the data values of its birth and death vertices v_{i_b} and v_{i_d} (cyan spheres in the data) are modified to project p_i to the diagonal. In this example, this results in a scalar field g which is close to f, with the prescribed topology.

point displacements in the birth-death space, as well as projections to the diagonal (encoding the *cancellation* of a persistence pair).

2.4 Persistence optimization

Several frameworks have been introduced for persistence optimization (Sec. 1.1). We review a recent, efficient, and generic framework [22].

Given a scalar data vector $v_f \in \mathbb{R}^{n_v}$ (Sec. 2.1), the purpose of persistence optimization is to modify v_f such that its persistence diagram $\mathcal{D}(f)$ minimizes a certain loss $\mathcal{L}$, specific to the considered problem. Then the solution space of the optimization problem is $\mathbb{R}^{n_v}$.

Let $\mathcal{F} : \mathbb{R}^{n_v} \to \mathbb{R}^{n_\sigma}$ be the *filtration map*, which maps a data vector v_f from the solution space $\mathbb{R}^{n_v}$ to a filtration represented as a vector $\mathcal{F}(v_f) \in \mathbb{R}^{n_\sigma}$, where the i^{th} entry contains the index of the i^{th} simplex σ_i of $\mathcal{K}$ in the global lexicographic order (Sec. 2.2). For convenience, we maintain a *backward filtration map* $\mathcal{F}^+ : \mathbb{R}^{n_\sigma} \to \mathbb{R}^{n_\sigma}$, which maps a filtration vector $\mathcal{F}(v_f)$ to a vector in $\mathbb{R}^{n_\sigma}$, whose i^{th} entry contains the index of the highest vertex (in global vertex order) of the i^{th} simplex in the global lexicographic order.

Given a persistence diagram $\mathcal{D}(f)$, the *critical simplex persistence order* can be introduced as follows. First, the points of $\mathcal{D}(f)$ are sorted by increasing birth and then, in case of birth ties, by increasing death. Let us call this order the *diagram order*. Then the set of persistence pairs can also be sorted according to the diagram order, by interleaving the birth and death simplices corresponding to each point. This results in an ordering of the critical simplices, called the *critical simplex persistence order*, where the $(2i)^{th}$ and $(2i+1)^{th}$ entries correspond respectively to the birth and death simplices of the i^{th} point p_i in the diagram order. Critical simplices which are not involved in a persistence pair (i.e., corresponding to homology classes of infinite persistence) are appended to this ordering, in increasing order of birth values.

Let us now consider the *persistence map* $\mathcal{P} : \mathbb{R}^{n_\sigma} \to \mathbb{R}^{n_\sigma}$, which maps a filtration vector $\mathcal{F}(v_f)$ to a persistence image $\mathcal{P}\big(\mathcal{F}(v_f)\big)$, whose i^{th} entry contains the *critical simplex persistence order* (defined above) for the i^{th} simplex in the global lexicographic order. For convenience, the entries corresponding to filtration indices which do not involve critical simplices are set to -1.

Now, to evaluate the relevance of a given diagram $\mathcal{D}(f)$ for the considered optimization problem, one needs to define a loss term. Let $\mathcal{E} : \mathbb{R}^{n_\sigma} \to \mathbb{R}$ be an energy function, which evaluates the diagram energy given its critical simplex persistence order. Then, given an input data vector v_f, the associated loss $\mathcal{L} : \mathbb{R}^{n_v} \to \mathbb{R}$ is given by:

$$\mathcal{L}(v_f) = \mathcal{E} \circ \mathcal{P} \circ \mathcal{F}(v_f).$$

Since distinct functions can admit the same persistence diagram, the global minimizer of the above loss may not be unique. However, given the search space ($\mathbb{R}^{n_v}$), the search for a global minimizer is still not tractable in practice and local minimizers will be searched instead.

If $\mathcal{E}$ is locally Lipschitz and a definable function of persistence, then the composition $\mathcal{E} \circ \mathcal{P} \circ \mathcal{F}$ is also definable and locally Lipschitz [22]. This implies that the generic loss $\mathcal{L}$ is differentiable almost everywhere and admits a well defined sub-differential. Then, a stochastic sub-gradient descent algorithm [57] converges almost surely to a critical point of $\mathcal{L}$ [26]. In practice, this means that the loss can be decreased by displacing each diagram point p_i in the diagram $\mathcal{D}(f)$ according to the sub-gradient. Assuming a constant global lexicographic order, this displacement can be back-propagated into modifications in data values in the vector v_f, by identifying the vertices v_{i_b} and v_{i_d} corresponding to the birth and death (Sec. 2.2) of the i^{th} point in the diagram order:

$$\begin{aligned} v_{i_b} &= \mathcal{F}^+\big(\mathcal{P}^{-1}(2i)\big) \\ v_{i_d} &= \mathcal{F}^+\big(\mathcal{P}^{-1}(2i+1)\big), \end{aligned} \tag{2}$$

and by updating their data values $v_f(v_{i_b})$ and $v_f(v_{i_d})$ accordingly.

3 Approach

This section describes our overall approach for the optimization of the topological simplification of scalar data.

Given the diagram $\mathcal{D}(f)$ of the input field f, we call a *signal* pair a persistence pair of $\mathcal{D}(f)$ which is selected by the user for preservation. Symmetrically, we call a *non-signal* pair a persistence pair of $\mathcal{D}(f)$ which is selected by the user for cancellation. Note that this distinction between *signal* and *non-signal* pairs is application dependent. In practice, the user can be aided by several criteria, such as persistence [28], geometric measures [21], etc. Then, topological simplification can be expressed as an optimization problem, with the following objectives:

1. Penalizing the persistence of the *non-signal* pairs;
2. Enforcing the precise preservation of the *signal* pairs.

In short, we wish to penalize the undesired features (objective *1*), and, at the same time, enforce the precise preservation of the features of the input which are deemed relevant (objective *2*). The latter objective is important in practice to preserve the accuracy of the features of interest. As later discussed in Sec. 4.3 (and illustrated in Fig. 6), *non-signal* and *signal* pairs do interact during the optimization, thereby perturbing *signal* pairs. In our experiments (Sec. 5), at each optimization iteration, 11% of the *signal* pairs are perturbed by *non-signal* pairs (on average, and up to 32%). In certain configurations, this can drastically alter the persistence of *signal* pairs. Hence, to address this issue, the precise preservation of the *signal* pairs should be explicitly constrained.

In the following, we formalize this specific optimization problem based on the generic framework described in Sec. 2.4. Our novel solver (for efficiently solving it) is presented in Sec. 4.

Let $\mathcal{D}_T$ be the *target diagram*. It can be obtained by copying the diagram $\mathcal{D}(f)$ of the input field f, and by removing the *non-signal* pairs. $\mathcal{D}_T$ encodes the two objectives of our problem: it describes the constraints for the cancellation of the noisy features of f (objective *1*) as well as the lock constraints for its features of interest (objective *2*).

In general, a perfect *reconstruction* (i.e., a scalar field g, close to f, such that $\mathcal{D}(g) = \mathcal{D}_T$) may not exist in 3D (deciding on its existence is NP-hard [4]). Thus, a practical strategy consists in optimizing the scalar field f such that its diagram $\mathcal{D}(f)$ gets *as close as possible* to $\mathcal{D}_T$ (and relaxing $||f-g||_\infty$). For this, we consider the following *simplification energy* $\mathcal{E}$ (to be used within the generic loss $\mathcal{L}$, introduced in Sec. 2.4):

$$\mathcal{E}\big(\mathcal{D}(f)\big) = \mathcal{W}_2\big(\mathcal{D}(f), \mathcal{D}_T\big)^2. \tag{3}$$

Algorithm 1 Baseline optimization approach for topological simplification.

Input: Input scalar field $f = f_0 : \mathcal{K} \to \mathbb{R}$.
Input: Target diagram $\mathcal{D}_T$.
Input: Stopping conditions $s \in [0,1]$, $j_{max} \in \mathbb{N}$.
Output: Topologically simplified scalar field $g = f_{final} : \mathcal{K} \to \mathbb{R}$.

$j \leftarrow 0$
$\mathcal{D}(f_j) \leftarrow PersistenceDiagramComputation(v_{f_j})$
$(\mathcal{L}(v_{f_j}), \phi_j^*) \leftarrow WassersteinDistanceComputation(\mathcal{D}(f_j), \mathcal{D}_T)$
do
  $j \leftarrow j+1$
  $v_{f_j} \leftarrow GradientDescentStep(\phi_{j-1}^*, v_{f_{j-1}})$
  $\mathcal{D}(f_j) \leftarrow PersistenceDiagramComputation(v_{f_j})$
  $(\mathcal{L}(v_{f_j}), \phi_j^*) \leftarrow WassersteinDistanceComputation(\mathcal{D}(f_j), \mathcal{D}_T)$
while $\mathcal{L}(v_{f_j}) > s\mathcal{L}(v_{f_0})$ and $j < j_{max}$

Since the Wasserstein distance is locally Lipschitz and a definable function of persistence [22], the optimization framework of Sec. 2.4 can be used to optimize $\mathcal{L} = \mathcal{E} \circ \mathcal{P} \circ \mathcal{F}$ with guaranteed convergence.

Specifically, at each iteration, given the optimal assignment ϕ^* induced by the Wasserstein distance between $\mathcal{D}(f)$ and $\mathcal{D}_T$ (Sec. 2.3), one can displace each point p_i in $\mathcal{D}(f)$ towards its individual target $\phi^*(p_i)$ by adjusting accordingly the corresponding scalar values $v_f(v_{i_b})$ and $v_f(v_{i_d})$ (Eq. 2). In practice, the generic optimization framework reviewed in Sec. 2.4 computes this displacement (given ϕ^*) via automatic differentiation [22] and by using Adam [57] for gradient descent.

However, depending on the employed step size, a step of gradient descent on v_f may change the initial filtration order (Sec. 2.2). Thus, after a step of gradient descent, the persistence diagram of the optimized data needs to be recomputed and, thus, so does its optimal assignment ϕ^* to the target $\mathcal{D}_T$. This procedure is then iterated, until the loss at the current iteration is lower than a user-specified fraction s of the loss at the first iteration (or until a maximum number j_{max} of iterations). We call this overall procedure the *baseline optimization for topological simplification*. It is summarized in Alg. 1 and illustrated in Fig. 4.

As shown in Alg. 1, each iteration j of the optimization involves a step of gradient descent, the computation of the diagram $\mathcal{D}(f_j)$ and the computation of its Wasserstein distance to $\mathcal{D}_T$. While the first of these three steps has linear time complexity, the other two steps are notoriously computationally expensive and both have cubic theoretical worst case time complexity, $\mathcal{O}(n_\sigma^3)$. In practice, practical implementations for persistence diagram computation tend to exhibit a quadratic behavior [8, 9, 38]. Moreover, the exact optimal assignment algorithm [66] can be approximated in practice to improve runtimes, for instance with Auction-based [11, 56, 89] or sliced approximations [23].

However, even when using the above practical implementations for persistence computation and assignment optimization, the baseline optimization approach for topological simplification has impractical runtimes for datasets of standard size. Specifically, for the simplifications considered in our experiments (Sec. 5), this approach can require up to days of computations per dataset. When it completes within 24 hours, the computation spends 20% of the time in persistence computation and 75% in assignment optimization.

4 Algorithm

This section describes our algorithm for topological simplification optimization. It is based on a number of practical accelerations of the baseline optimization (Alg. 1), which are particularly relevant for the problem of topological simplification.

4.1 Direct gradient descent

Instead of relying on automatic differentiation and the Adam optimizer [57] as done in the generic framework reviewed in Sec. 2.4, similar to [76], we can derive the analytic expression of the gradient of our energy on a per-iteration basis (Eq. 3) and perform at each iteration a direct step of gradient descent, in order to improve performance. Specifically, at the iteration j (Alg. 1), given the current persistence diagram $\mathcal{D}(f_j)$, if the assignments between diagonal points are ignored (these have zero cost, Sec. 2.3), Eq. 3 can be re-written as:

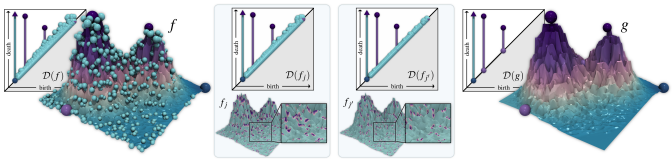

Fig. 5: Updated vertices (dark purple vertices, center insets) along the topological simplification optimization of a noisy terrain (*non-signal* pairs, to simplify, are shown in cyan). In this example, only 20% of the vertices are updated per iteration on average. The discrete gradient (at the core of a recent, fast persistence computation algorithm [38]) only needs to be recomputed for these, yielding a x2 speedup for persistence computation.

$$\mathcal{E}(\mathcal{D}(f_j)) = \min_{\phi \in \Phi} \sum_{p_i \in \mathcal{D}(f_j)} ||p_i - \phi(p_i)||_2^2.$$

As the optimal assignment ϕ_j^* (i.e., minimizing the energy for a fixed $\mathcal{D}(f_j)$) is constant at the iteration j, the energy can be re-written as:

$$\mathcal{E}(\mathcal{D}(f_j)) = \sum_{p_i \in \mathcal{D}(f_j)} \big(p_{i_b} - \phi_j^*(p_i)_b\big)^2 + \big(p_{i_d} - \phi_j^*(p_i)_d\big)^2.$$

Then, given Eq. 2, for the iteration j, the overall optimization loss $\mathcal{L}(v_{f_j})$ can be expressed as a function of the input data vector v_{f_j}:

$$\mathcal{L}(v_{f_j}) = \sum_{p_i \in \mathcal{D}(f_j)} \big(v_{f_j}(v_{i_b}) - \phi_j^*(p_i)_b\big)^2 + \big(v_{f_j}(v_{i_d}) - \phi_j^*(p_i)_d\big)^2.$$

Then, for the iteration j, given the constant assignment ϕ_j^*, this loss is convex with v_{f_j} (in addition to being locally Lipschitz) and gradient descent can be considered. Specifically, let $\nabla v_{i_b} \in \mathbb{R}^{n_v}$ be a vector with zero entries, except for the $i_b{}^{th}$ entry, set to 1. Let $\nabla v_{i_d} \in \mathbb{R}^{n_v}$ be the vector constructed similarly for v_{i_d}. Then, by the chain rule, we have:

$$\begin{aligned}\nabla \mathcal{L}(v_{f_j}) \quad = \quad & \sum_{p_i \in \mathcal{D}(f_j)} \Big(2\big(v_{f_j}(v_{i_b}) - \phi_j^*(p_i)_b\big)\nabla v_{i_b} \\ + \quad & 2\big(v_{f_j}(v_{i_d}) - \phi_j^*(p_i)_d\big)\nabla v_{i_d}\Big).\end{aligned}$$

We now observe that the gradient can be split into two terms, a *birth gradient* (noted $\nabla \mathcal{L}(v_{f_j})_b$) and a *death gradient* (noted $\nabla \mathcal{L}(v_{f_j})_d$):

$$\begin{aligned}\nabla \mathcal{L}(v_{f_j})_b &= \textstyle\sum_{p_i \in \mathcal{D}(f_j)} 2\big(v_{f_j}(v_{i_b}) - \phi_j^*(p_i)_b\big)\nabla v_{i_b} \\ \nabla \mathcal{L}(v_{f_j})_d &= \textstyle\sum_{p_i \in \mathcal{D}(f_j)} 2\big(v_{f_j}(v_{i_d}) - \phi_j^*(p_i)_d\big)\nabla v_{i_d}.\end{aligned} \tag{4}$$

Then, given the above gradient expressions, a step of gradient descent is obtained by:

$$v_{f_{j+1}} = v_{f_j} - \big((\alpha_b \nabla \mathcal{L}(v_{f_j})_b + \alpha_d \nabla \mathcal{L}(v_{f_j})_d\big),$$

where $\alpha_b, \alpha_d \in \mathbb{R}$ are the gradient step sizes for the birth and death gradients respectively. Such individual step sizes enable an explicit control over the evolution of the persistence pairs to cancel (see Sec. 5.3).

4.2 Fast persistence update

As described in Sec. 3, each optimization iteration j involves the computation of the persistence diagram of the data vector v_{f_j}, which is computationally expensive (20% of the computation time on average). Subsequently, for each persistence pair p_i, the data values of its vertices v_{i_b} and v_{i_d} will be updated given the optimal assignment ϕ_j^*.

A key observation can be leveraged to improve the performance of the persistence computation stage. Specifically, the updated data vector $v_{f_{j+1}}$ only contains updated data values for the subset of the vertices of $\mathcal{K}$ which are the birth and death vertices v_{i_b} and v_{i_d} of a persistence pair p_i. Then, only a small fraction of the vertices are updated from one iteration to the next, as shown in Fig. 5. In practice, for the simplifications considered in our experiments (Sec. 5), 90% of the vertices of $\mathcal{K}$ do *not* change their data values between consecutive iterations (on average over our datasets, with the baseline optimization). This indicates that a procedure capable of quickly updating the persistence diagram $\mathcal{D}(f_{j+1})$ based on $\mathcal{D}(f_j)$ has the potential to improve performance in practice.

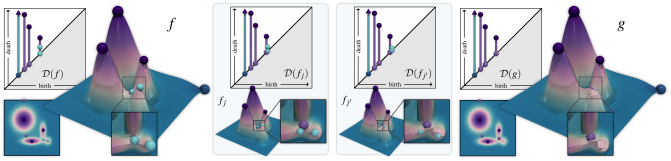

Fig. 6: Interactions between *non-signal* and *signal* pairs during the optimization. A multi-saddle vertex can be involved in both a *non-signal* pair p (cyan bar in $\mathcal{D}(f)$) and a *signal* pair p' (vertically aligned purple bar in $\mathcal{D}(f)$). At iteration j, the update of the *non-signal* pair p unfolds the multi-saddle into multiple simple saddles of distinct values, effectively *perturbing* the birth of the *signal* pair p' and making it *non-still*. In real-life data, especially in 3D, such configurations occur often, and cascade. In our experiments (Sec. 5), at each iteration, 11% of the *signal* pairs are perturbed this way by *non-signal* pairs (on average, and up to 32%). This is addressed by our loss (Sec. 3) which enforces *signal* pair preservation.

Several approaches focus on updating a persistence diagram based on a previous estimation [25, 61], with a time complexity that is linear for each vertex order transposition between the two scalar fields. However, this number of transpositions can be extremely large in practice.

Instead, we derive a simple procedure based on recent work for computing persistent homology with Discrete Morse Theory (DMT) [31, 38, 78], which we briefly review here for completeness. Specifically, the *Discrete Morse Sandwich* (DMS) approach [38] revisits the seminal algorithm *PairSimplices* [28] within the DMT setting, with specific accelerations for volume datasets. This algorithm is based on two main steps. First, a *discrete gradient field* is computed, for the fast identification of zero-persistence pairs. Second, the remaining persistence pairs are computed by restricting the algorithm *PairSimplices* to the critical simplices (with specific accelerations for the persistent homology groups of dimension 0 and $d-1$).

The first key practical insight about this algorithm is that its first step, discrete gradient computation, is documented to represent in practice, in 3D, 66% of the persistence computation time on average [38] (in sequential mode). This indicates that, if one could quickly update the discrete gradient between consecutive iterations, the overall persistence computation step could be accelerated by up to a factor of 3 in practice.

The second key practical insight about this algorithm is that the discrete gradient computation is a completely *local* operation, specifically to the lower star of each vertex v [78] (i.e., the co-faces of v containing no higher vertex than v in the global vertex order).

Thus, we leverage the above two observations to expedite the computation of the diagram $\mathcal{D}(f_j)$, based on the diagram $\mathcal{D}(f_{j-1})$. Specifically, we mark as *updated* all the vertices of $\mathcal{K}$ for which the data value is updated by gradient descent at iteration $j-1$ (Sec. 4.1). Then, the discrete gradient field at step j is copied from that at step $j-1$ and the local discrete gradient computation procedure [78] is only re-executed for these vertices for which the lower star may have changed from step $j-1$ to step j, i.e. the vertices marked as *updated* or which contain *updated* vertices in their star. This localized update guarantees the computation of the correct discrete gradient field at step j, with a very small number of local re-computations. Next, the second step of the DMS algorithm [38] (i.e., the computation of the persistence pairs from the critical simplices) is re-executed as-is.

4.3 Fast assignment update

As described in Sec. 3, each optimization iteration j involves the computation of the optimal assignment ϕ_j^* from the current diagram $\mathcal{D}(f_j)$ to the target $\mathcal{D}_T$, which is computationally expensive.

However, for the problem of simplification, a key practical observation can be leveraged to accelerate this assignment computation. In practice, an important fraction of the pairs of $\mathcal{D}(f_j)$ to optimize (among *signal* and *non-signal* pairs) may only move slightly in the domain from one iteration to the next (as illustrated in Fig. 6), and some do not move at all. For these pairs which do not move at step j, the assignment can be re-used from the step $j-1$, hence reducing the size of the assignment problem (Sec. 2.3), and hence reducing its practical runtime.

Given two persistence diagrams $\mathcal{D}(f_j)$ and $\mathcal{D}(f_{j-1})$, we call a *still* persistence pair a pair of points (p_i, p_i') with $p_i \in \mathcal{D}(f_j)$ and $p_i' \in \mathcal{D}(f_{j-1})$ such that $v_{i_b} = v_{i'_b}$ and $v_{i_d} = v_{i'_d}$. In other words, a *still* persistence pair is a pair which does not change its birth and death vertices from one optimization iteration to the next. In practice, for the simplifications considered in our experiments (Sec. 5), 84% of the persistence pairs of $\mathcal{D}(f_j)$ are still (on average over the iterations and our test datasets, Sec. 5). This indicates that a substantial speedup could be obtained by expediting the assignment computation for still pairs.

Let $\mathcal{S}$ be the set of still pairs between the iteration j and $j-1$. Then, for each pair $(p_i, p_i') \in \mathcal{S}$, we set $\phi_j^*(p_i) \leftarrow \phi_{j-1}^*(p_i')$. Concretely, we re-use at step j the assignment at step $j-1$ for all the still pairs.

Next, let $\overline{\mathcal{D}(f_j)}$ be the *reduced* diagram at step j, i.e., the subset of $\mathcal{D}(f_j)$ which does not contain still pairs: $\overline{\mathcal{D}(f_j)} = \mathcal{D}(f_j) - \{p_i \in \mathcal{D}(f_j), (p_i, p_i') \in \mathcal{S}\}$. Similarly, let $\overline{\mathcal{D}_{T\,j}}$ be the *reduced* target at step j, i.e., the subset of $\mathcal{D}_T$ which has not been assigned to still pairs: $\overline{\mathcal{D}_{T\,j}} = \mathcal{D}_T - \{p_i'' \in \mathcal{D}_T, p_i'' = \phi_j^*(p_i), (p_i, p_i') \in \mathcal{S}\}$. Then, we finally complete the assignment between $\mathcal{D}(f_j)$ and $\mathcal{D}_T$ by computing the Wasserstein distance between $\overline{\mathcal{D}(f_j)}$ and $\overline{\mathcal{D}_{T\,j}}$, as documented in Sec. 2.3.

Note that, in the special case where the reduced target $\overline{\mathcal{D}_{T\,j}}$ is empty (i.e., all *signal* pairs are *still*), the reduced diagram $\overline{\mathcal{D}(f_j)}$ only contains *non-signal* pairs. Then, the optimal assignment can be readily obtained (without any assignment optimization) by simply assigning each point p_i in $\overline{\mathcal{D}(f_j)}$ to its diagonal projection $\Delta(p_i)$. However, from our experience, such a perfect scenario never occurs on real-life data, at the notable exception of the very first iteration (before the data values are actually modified by the solver). For the following iterations, many *signal* pairs are not still in practice. Fig. 6 illustrates this with a simple 2D example involving a multi-saddle vertex. However, in real-life data, such configurations occur very often, and cascade. Also, these configurations get significantly more challenging in 3D. For instance, the birth and death vertices of a given *signal* pair can both be multi-saddles, themselves possibly involved with *non-signal* pairs to update (hence yielding perturbations in the *signal* pair). In certain configurations, this can drastically alter the persistence of the *signal* pairs affected by such perturbations. This is addressed by our loss (Sec. 3) which enforces the preservation of the *signal* pairs via assignment optimization.

5 Results

This section presents experimental results obtained on a computer with two Xeon CPUs (3.0 GHz, 2x8 cores, 64GB of RAM). We implemented our algorithm (Sec. 4) in C++ (with OpenMP) as a module for TTK [14, 85]. We implemented the baseline optimization approach (Sec. 3) by porting the original implementation by Carriere et al. [22] from TensorFlow/Gudhi [1, 64] to PyTorch/TTK [74] and by applying it to the loss described in Sec. 3. We chose this approach as a baseline, since its implementation is simple and publicly available, and since it provides performances comparable to alternatives [70]. In our implementations, we use the DMS algorithm [38] for persistence computation (as it is reported to provide the best practical performance for scalar data) and the Auction algorithm [11, 56] for the core assignment optimization, with a relative precision of 0.01, as recommended in the literature [56]. Persistence computation with DMS [38] is the only step of our approach which leverages parallelism (see [38] for a detailed performance analysis). Experiments were performed on a selection of 10 (simulated and acquired) 2D and 3D datasets extracted from public repositories [58, 87], with an emphasis on 3D datasets containing large filament structures (and thus possibly, many persistent saddle pairs). The 3D datasets were resampled to a common resolution (256^3), to better observe runtime variations based on the input topological complexity. Moreover, for each dataset, the data values were normalized to the interval $[0, 1]$, to facilitate parameter tuning across distinct datasets.

Our algorithm is subject to two meta-parameters: the gradient step sizes α_b and α_d. To adjust them, we selected as default values the ones which minimized the runtime for our test dataset with the largest diagram. This resulted in $\alpha_b = \alpha_d = 0.5$ (which coincides, given a

Table 1: Time performance comparison between the baseline optimization approach (Sec. 3) and our solver (Sec. 4), for a basic simplification (*non-signal* pairs: input pairs less persistent than 1% of the function range). The column N.S.S.P. reports the average percentage of non-still *signal* pairs for our solver. The stopping condition is set to $s = 0.01$.

Dataset	d	$\lvert\mathcal{D}(f)\rvert$	$\lvert\mathcal{D}_T\rvert$	Baseline (Sec. 3)			Our solver (Sec. 4)				
				#It.	Time/It (s.)	Time (s.)	N.S.S.P.	#It.	Time/It (s.)	Time (s.)	Speedup
Cells	2	7,676	2,635	89	0.58	52	0.07%	10	0.20	2	**26**
Ocean Vortices	2	12,069	2,781	87	0.61	53	8.02%	12	0.25	3	**18**
Aneurysm	3	38,490	24,725	80	29.89	2,391	3.70%	8	11.63	93	**26**
Bonsai	3	168,489	55,464	67	56.73	3,801	3.90%	10	13.50	135	**28**
Foot	3	754,965	474,271	60	914.47	54,868	11.28%	4	104.25	417	**132**
Neocortical Layer Axon	3	765,406	483,791	89	735.36	65,447	32.04%	8	263.38	2,107	**31**
Dark Sky	3	1,140,653	774,793	NA	NA	**> 24h**	9.08%	6	122.00	732	**> 118**
Backpack	3	1,331,362	84,402	66	305.58	20,168	21.46%	9	62.22	560	**36**
Head Aneurysm	3	1,345,168	234,672	NA	NA	**> 24h**	6.68%	5	83.80	419	**> 206**
Chameleon	3	3,641,961	32,578	55	210.51	11,578	18.22%	8	74.75	598	**19**

Table 2: Individual gains (in percentage of runtime) for each of our accelerations for the topological simplification parameters used in Tab. 1.

Dataset	d	$\lvert\mathcal{D}(f)\rvert$	$\lvert\mathcal{D}_T\rvert$	Persistence Update (Sec. 4.2)	Assignment Update (Sec. 4.3)
Cell	2	7,676	2,635	5.6	79.2
Ocean Vortices	2	12,069	2,781	−0.4	79.8
Aneurysm	3	38,490	24,725	41.4	24.8
Bonsai	3	168,489	55,464	12.1	80.6
Foot	3	754,965	474,271	0.2	90.9
Neocortical Layer Axon	3	765,406	483,791	0.0	74.6
Dark Sky	3	1,140,653	774,793	3.2	96.7
Backpack	3	1,331,362	84,402	1.1	74.5
Head Aneurysm	3	1,345,168	234,672	1.3	93.4
Chameleon	3	3,641,961	32,578	1.5	61.3

Table 3: Quality comparison between the baseline optimization approach (Sec. 3) and our solver (Sec. 4) for the parameters used in Tab. 1.

Dataset	d	Baseline (Sec. 3)			Our solver (Sec. 4)		
		$\mathcal{L}(v_g)$	$\lvert\lvert f-g\rvert\rvert_2$	$\lvert\lvert f-g\rvert\rvert_\infty$	$\mathcal{L}(v_g)$	$\lvert\lvert f-g\rvert\rvert_2$	$\lvert\lvert f-g\rvert\rvert_\infty$
Cells	2	0.0003	0.2939	0.0054	0.0003	0.2996	0.0070
Ocean Vortices	2	0.0006	0.3710	0.0055	0.0005	0.3769	0.0074
Aneurysm	3	0.0007	0.3481	0.0063	0.0006	0.3174	0.0117
Bonsai	3	0.0064	1.0243	0.0060	0.0053	1.0541	0.0117
Foot	3	0.0326	2.0526	0.0058	0.0297	2.0483	0.0127
Neocortical Layer Axon	3	0.0271	2.0876	0.0085	0.0279	2.1435	0.0154
Dark Sky	3	NA	NA	NA	0.0166	1.8617	0.0148
Backpack	3	0.0339	2.4438	0.0070	0.0312	2.1991	0.0159
Head Aneurysm	3	NA	NA	NA	0.0750	3.7571	0.0130
Chameleon	3	0.1679	5.1264	0.0055	0.1736	4.7773	0.0143

persistence pair to cancel, to a displacement of its birth and death vertices halfway towards the other, in terms of function range). For the baseline optimization approach (Sec. 3), we set the initial learning rate of Adam [57] to the largest value which still enabled practical convergence for all our datasets (specifically, 10^{-4}). For both approaches, we set the maximum number of iterations j_{max} to $1,000$ (however, it has never been reached in our performance experiments).

5.1 Quantitative performance

The time complexity of each iteration of the baseline optimization is cubic in the worst case, but quadratic in practice (Sec. 3). As discussed in Sec. 4, our approach has the same worst case complexity, but behaves more efficiently in practice thanks to our accelerations.

Tab. 1 provides an overall comparison between the baseline optimization (Sec. 3) and our solver (Sec. 4). Specifically, it compares both approaches in terms of runtime, for a basic simplification scenario: *non-signal* pairs are identified as the input pairs with a persistence smaller than 1% of the function range (see Appendix A for an aggressive simplification scenario). For both approaches, we set the stopping criterion s to 0.01, such that both methods reach a similar residual loss at termination (and hence produce results of comparable quality). This table shows that for a basic simplification scenario, our approach produces results within minutes (at most 35). In contrast, the baseline approach does not produce a result after 24 hours of computation for the largest examples. Otherwise, it still exceeds hours of computation for diagrams of modest size. Overall, our approach results in an average $\times 64$ speedup. This acceleration can be explained by several factors. First, the direct gradient descent (Sec. 4.1) requires *fewer* iterations than the baseline approach (we discuss this further in the next paragraph, presenting Tab. 3). Second, our approach also results in *faster* iterations, given the accelerations presented in Sec. 4.

In practice, the overall runtime for our solver is a function of the size of the input and target diagrams (large diagrams lead to large assignment problems). The size of the topological features in the geometric domain also plays a role (larger features will require more iterations). Finally, the number of still *signal* pairs also plays a role given our fast assignment update procedure (Sec. 4.3, a large number of still *signal* pairs leads to faster assignments). For instance, the total number of pairs (input plus target) for the *Neocortical Layer Axon* dataset is about $\times 20$ larger than that of the *Aneursym* dataset, and the ratio between their respective runtime is also about 20. Moreover, the *Foot* and *Neocortical Layer Axon* datasets have comparable overall sizes. However, the latter dataset results in a computation time $\times 5$ larger. This can be partly explained by the fact that the topological features are larger in this dataset, yielding twice more iterations (hence explaining a $\times 2$ slowdown). Moreover, the per-iteration runtime is also $\times 2.5$ slower (explaining the overall $\times 5$ slowdown), due the higher percentage of non-still *signal* pairs, increasing the size of the assignment problem.

The runtime gains provided by our individual accelerations are presented in Tab. 2. Specifically, our procedure for fast Persistence update (Sec. 4.2) can save up to 41.4% of overall computation time, and 6.6% on average. Our procedure for fast assignment update (Sec. 4.3) provides the most substantial gains, saving up to 97% of the overall computation time for the largest target diagram, and 76% on average (see Appendix A for a discussion regarding aggressive simplifications).

Tab. 3 compares the quality of the output obtained with the baseline optimization (Sec. 3) and our algorithm (Sec. 4), for the simplification parameters used in Tab. 1. The quality is estimated based on the value of the loss at termination ($\mathcal{L}(v_g)$), which assesses the quality of the topological simplification. To estimate the proximity of the solution g to the input f, we also evaluate the distances $||f-g||_2$ (giving a global error for the entire dataset) and $||f-g||_\infty$ (giving a pointwise worst case error). We refer the reader to Appendix B for complementary quality statistics. Overall, Tab. 3 shows that our approach provides comparable losses to the baseline approach (sometimes marginally better). In terms of data fitting, our approach also provides comparable global distances $||f-g||_2$ (sometimes marginally better). For the pointwise worst case error ($||f-g||_\infty$), our approach can result in degraded values (by a factor 2). This can be explained by the fact that, when tuning the parameters of our approach, we optimized the gradient step size to minimize running time, hence possibly triggering in practice bigger pointwise shifts in data values. In contrast, the baseline approach uses the Adam [57] algorithm, which optimizes step sizes along the iterations, possibly triggering milder pointwise shifts in data values. In principle, the $||f-g||_\infty$ distance could be improved for our solver by considering smaller step sizes, but at the expense of more iterations.

5.2 Analyzing topologically simplified data

Our approach enables the direct visualization and analysis of topologically simplified data. This is illustrated in Fig. 1, which shows the processing of an acquired dataset (*"Aneurysm"*) representing a network of arteries. As documented in the literature [51, 65], this network exhibits a typical tree-like structure, whose accurate geometric extraction is relevant for medical analysis. The filament structure of the arteries can be simply extracted by considering the *discrete integral lines* [38] (a.k.a. *v-paths* [31]) which connect 2-saddles to maxima and which have a minimum function value above 0.1 (scalar fields are normalized). This value 0.1 generates an isosurface (transparent surfaces, Fig. 1) which accurately captures the geometry of the blood vessels. Hence, selecting the discrete integral lines above that threshold guarantees the extraction of the filament structures within the vessels.

As shown in Fig. 1, the diagram $\mathcal{D}(f)$ contains several saddle pairs, corresponding to persistent 1-dimensional generators [38, 53] (curves colored by persistence in the inset zooms), which yields incorrect loops in the filament structure (which is supposed to have a tree-like structure [51]). To remove loops in networks of discrete integral lines, an established topological technique, relying on standard discrete Morse theory [31], consists in reversing the discrete gradient [40] along *saddle connectors*. We recap this procedure here for completeness. Given the persistence diagram $\mathcal{D}(f)$, we process its *non-signal* saddle pairs in increasing order of persistence. For each saddle pair (σ_b, σ_d), its *saddle connector* is constructed by following the discrete gradient of f from σ_d down to σ_b. Next, the pair of critical simplices (σ_b, σ_d) is cancelled, in the discrete sense, by simply reversing the discrete gradient along its saddle connector [31] (i.e., each discrete vector is reversed to point to the preceding co-face). Such a reversal is marked as *valid* if it does

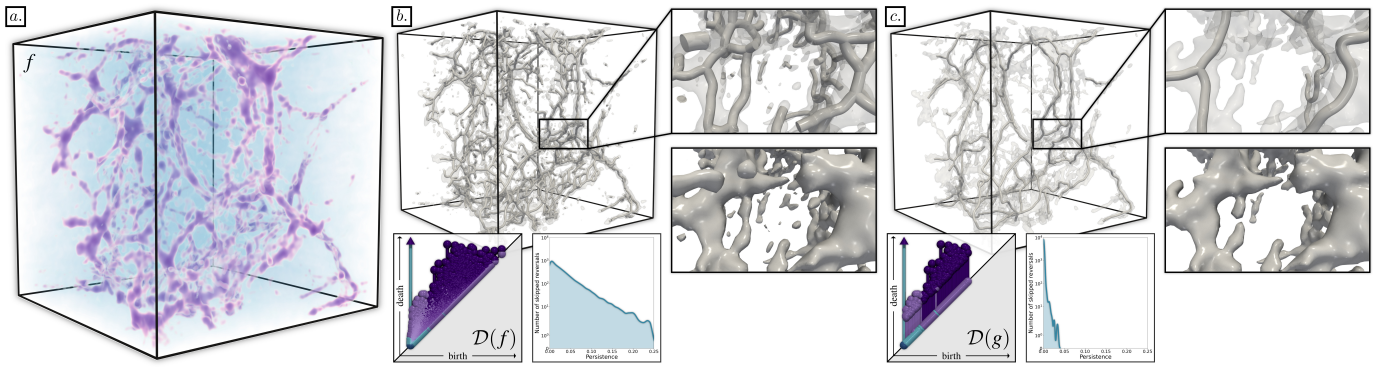

Fig. 7: Topological simplification optimization for a challenging dataset (*"Dark Sky"*: dark matter density in a cosmology simulation, (a), *signal pairs*: pairs with a persistence larger than 0.25). The geometry of the *cosmic web* [80, 84] is captured (b) by an isosurface (at isovalue 0.4) and its core filament structure is extracted by the upward discrete integral lines, started at 2-saddles above 0.4. The latter structure contains many small-scale loops as many, persistent saddle connector reversals could not be performed (bottom left histogram). The local minimum g of the simplification energy (Eq. 3) found by our solver (c) has a number of *non-signal* pairs reduced by 92%. This results in a less cluttered visualization, as the cosmic web has a less complicated topology (noisy connected components are removed and small scale handles are cut, inset zooms). This also induces fewer skips of persistent saddle connector reversals (bottom right histogram), hence simplifying more loops and revealing the main filament structure.

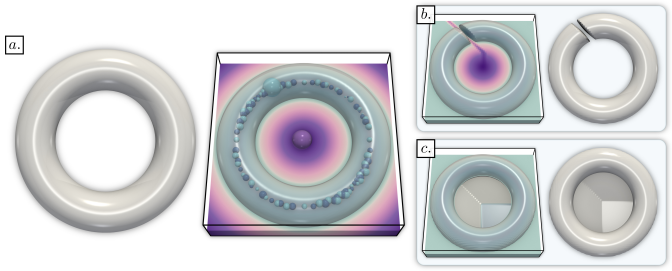

Fig. 8: Handle removal on a torus example. (a) The input surface S (left) is used to compute a 3D signed distance field f (right, color map). f contains a persistent saddle pair (large spheres) encoding the handle of the torus and many low-persistence minimum-saddle pairs (smaller spheres, radius scaled by persistence) which are artifacts (located on the medial axis of S) of the sampling of the distance field (which has discontinuous derivatives). The handle can be removed in the output surface S' (b,c) by considering the zero level set of a simplified field g obtained with our approach (in this example, only the persistent generator of f with infinite persistence has been maintained). The handle can be removed either by *cutting* (b) (by only using the *birth gradient*, Eq. 4) or by *filling* (c) (by only using the *death gradient*, Eq. 4).

not create any cycle in the discrete gradient field. The validity of a reversal is important since invalid reversals result in discrete vector fields which no longer describe valid scalar fields, and from which the subsequent extraction of integral lines can generate further loops (which we precisely aim to remove). The cancellation of a saddle pair (σ_b, σ_d) is *skipped* if the reversal of its saddle connector is not valid, or if its saddle connector does not exist. The latter case occurs for instance for nested saddle pairs, when an invalid reversal of a small persistence pair prevents the subsequent reversal of a larger one. Finally, when all the *non-signal* saddle pairs have been processed, the simplified filament structures are simply obtained from the simplified discrete gradient, by initiating integral lines from 2-saddles up to maxima.

However, in the example of Fig. 1, this saddle connector reversal procedure fails at simplifying the spurious loops in the filament structures, while maintaining a valid discrete gradient (Fig. $1(b)$). As discussed in the literature [40], integral line reversal is indeed not guaranteed to completely simplify saddle pairs (*v-path* co-location [54] as well as specific cancellation orderings [46, 48] can challenge reversals, the latter issue being a manifestation of the NP-hardness of the problem [4]). This is evaluated in the bottom left histogram, which reports the number of *skipped* saddle connector reversals as a function of the persistence of the corresponding pair. Specifically, this histogram shows that the reversal of several high-persistence saddle pairs could not be performed, hence the presence of large loops in the extracted filament structures.

Our approach can be used to efficiently generate a function g which is close to the input f and from which the removal of saddle pairs has been optimized, while maintaining intact the rest of the features (see the resulting diagram $\mathcal{D}(g)$, Fig. 1). Specifically, we set as *non-signal* pairs *all* the saddle pairs of the input, and we set as *signal* pairs all the others (irrespective of their persistence). This enables a direct visualization and analysis of the topologically simplified data, where isosurface handles have been cut (Fig. 1c, bottom-right zoom vs. Fig. 1b, bottom-right zoom) and where most spurious filament loops have been consequently simplified (Fig. 1, top zoom). Note that, as shown in the bottom right histogram, our optimization modifies the input data f into a function g where reversal skips still occur. This is due to the fact that our solver identifies a *local* minimum of the simplification energy (Eq. 3) and that, consequently, a few saddle pairs, with low persistence, may still remain (we recall that sublevel set simplification is NP-hard [4], see Sec. 5.4 for further discussions). However, the skipped reversals which remain after our optimization (Fig. 1, bottom right histogram) only involve very low persistence pairs, hence allowing the cancellation of the largest loops overall.

Fig. 7 illustrates our simplification optimization for a challenging dataset (*"Dark Sky"*: dark matter density in a cosmology simulation). The isosurface capturing the *cosmic web* [80, 84] (inset zooms) has a complicated topology (many noisy connected components and handles), which challenges its visual inspection. Its core filament structure also contains many small-scale loops since many persistent saddle connector reversals could not be performed (Fig. 7, bottom left histogram). Our solver provides a local minimum g to the simplification energy (Eq. 3) with a number of *non-signal* pairs reduced by 92% (see Appendix C for further stress experiments). This results in a less cluttered visualization, as the resulting cosmic web (Fig. $7(c)$) has a less complicated topology (noisy connected components are removed and small scale handles are cut, inset zooms). Moreover, our optimization modifies the data in a way that is more conducive to persistent saddle connector reversals (bottom right histogram), hence simplifying more loops and, thus, better revealing overall the large-scale filament structure of the cosmic web.

5.3 Repairing genus defects in surface processing

Our work can also be used to repair genus defects in surface processing, where surface models, in particular when they are acquired, can include spurious handles due to acquisition artifacts. While several approaches have been proposed to address this issue [24, 91, 92], they typically rely on intensive automatic optimizations, aiming at selecting the *best* sequence of local simplification primitives (i.e. *cutting* or *filling*). In contrast, our approach relies on a simpler and lightweight procedure, which provides control to the user over the primitives to use. Moreover, most existing techniques simplify only one sublevel set, while our

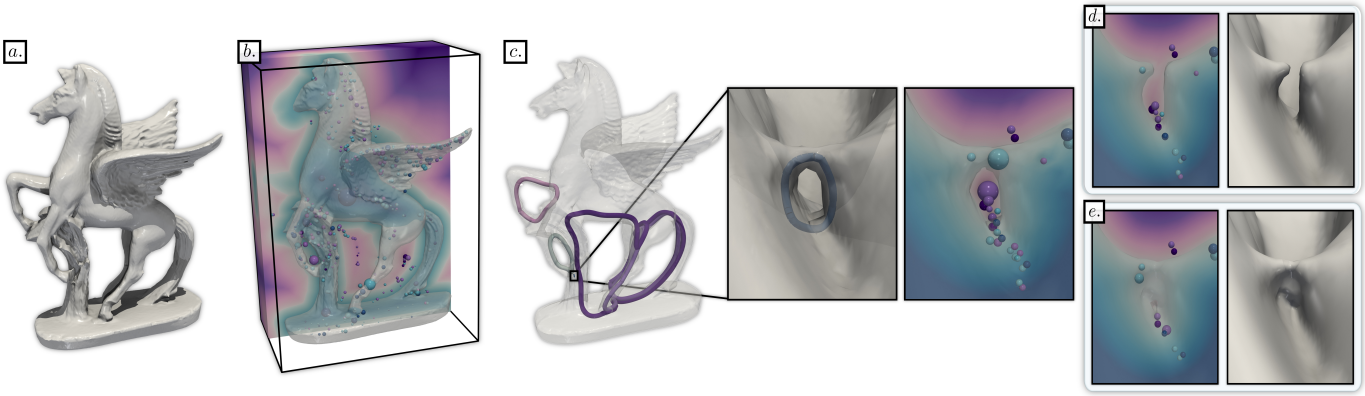

Fig. 9: Removal of a spurious handle from an acquired surface S (a). First, the signed distance field f is computed from S (b). f is shown with a color map on the clipped volume, with its critical simplices colored per dimension, with a sphere with a radius proportional to their persistence. The extraction of the 1-dimensional persistent generators [38, 53] ((c), colored by persistence) reveals the existence of a short generator in f, corresponding to a small handle defect in S (under the Pegasus front left hoof, see inset zooms). Our framework can repair this defect by simplifying the corresponding saddle pair, either by *cutting* ((d), by only using the *birth* gradient, Eq. 4) or by *filling* ((e), by only using the *death* gradient, Eq. 4).

approach processes the whole function range. For this, we consider the three-dimensional signed distance field f to the input surface S, computed on a regular grid (i.e., f encodes for each grid vertex v the distance to the closest point on the surface S, multiplied by -1 if v is located within the volume enclosed by S). For such a field, the zero level set $f^{-1}(0)$ coincides with S. Then, the removal of a handle in S can be performed by creating a simplified signed distance field g, where the corresponding saddle pair has been canceled. Finally, the zero level set $g^{-1}(0)$ provides the simplified surface S'.

This process is illustrated in Fig. 8 where the handle of a torus is removed. Note that, from a topological point of view, this operation can be performed in two ways: either by cutting the handle (Fig. 8(b)), or by filling it (Fig. 8(c)). This can be controlled in our solver by simply adjusting the step sizes for the birth and death gradients (Sec. 4.1). Specifically, given a saddle pair to remove $p_i \in \mathcal{D}(f)$, handle cutting is obtained by setting α_d to zero. Then, the death vertex v_{i_d} will not be modified (above the zero level set), while only the birth vertex v_{i_b} (located in the star of the 1-saddle creating the handle) will increase its value above 0, effectively disconnecting the handle in the output surface S'. Handle filling is obtained symmetrically, by setting α_b to zero (effectively forcing the 2-saddle to decrease its value below 0).

Fig. 9 presents a realistic example of an acquired surface from a public repository [87], which contains a spurious handle, due to acquisition artifacts. First, the signed distance field is computed and its 1-dimensional persistent generators [38, 53] are extracted. The shortest generator corresponds to a small handle, which happens to be a genus defect in this example. Then, the user can choose to repair this defect via cutting or filling, resulting in a repaired surface S' which is close to the input S, and from which the spurious handle has been removed.

5.4 Limitations

Our approach is essentially numerical and, thus, suffers from the same limitations as previous numerical methods for topological simplification (Sec. 1.1). Specifically, the *non-signal* pairs are canceled by our approach by decreasing their persistence to a target value of zero. However, this decrease is ultimately limited by the employed numerical precision (typically, 10^{-6} for single-precision floating point values). From a strictly combinatorial point of view, this can result in *residual pairs* with an arbitrarily small persistence (i.e., in the order of the numerical precision). In principle, this drawback is common to all numerical methods (although sometimes mitigated via smoothing). Then, when computing topological abstractions, these residual pairs need to be removed from the computed abstraction (e.g., with integral line reversal, Sec. 5.2). However, as discussed in the literature [40, 43], post-process mechanisms for simplifying topological abstractions may not guarantee a complete simplification of the abstractions either (this is another concrete implication of the NP-hardness of sublevel set simplification [4]). However, our experiments (Sec. 5.2) showed that our numerical optimization helped such combinatorial mechanisms, by preprocessing the data in a way that resulted eventually in fewer persistent reversal skips (Figs. 1 and 7, right versus left histograms).

Similar to previous persistence optimization frameworks, our approach generates a *local* minimum of the simplification energy (Eq. 3), and thus it is not guaranteed to reach the global minimum. As a reminder, in 3D, an optimal simplification (i.e., $\mathcal{D}(g) = \mathcal{D}_T$) may not exist and finding a sublevel set simplification is NP-hard [4]. However, our experiments (Sec. 5.1) showed that our approach still generated solutions whose quality was on par with the state-of-the-art (comparable losses and distances to the input), while providing substantial accelerations. Moreover, as shown in Sec. 5.2, these solutions enabled the direct visualization of isosurfaces whose topology was indeed simplified (fewer components and handles) and they were also conducive to improved saddle connector reversals.

6 Conclusion

This paper introduced a practical solver for topological simplification optimization. Our solver is based on tailored accelerations, which are specific to the problem of topological simplification. Our accelerations are simple and easy to implement, but result in significant gains in terms of runtime, with $\times 60$ speedups on average on our datasets over state-of-the-art persistence optimization frameworks (with both fewer and faster iterations), for comparable output qualities. This makes topological simplification optimization practical for real-life three-dimensional datasets. We showed that our contributions enabled a direct visualization and analysis of the topologically simplify data, where the topology of the extracted isosurfaces was indeed simplified (fewer connected components and handles). We applied our approach to the extraction of prominent filament structures in 3D data, and showed that our pre-simplification of the data led to practical improvements for the removal of spurious loops in filament structures. We showed that our contributions could be used to repair genus defects in surface processing, where handles due to acquisition artifacts could be easily removed, with an explicit control on the repair primitives (cutting or filling).

While it is tailored to the problem of simplification, our solver is still generic and could in principle be used for other persistence optimization problems, however, with possibly less important performance gains. In the future, we will consider other optimization problems and investigate other acceleration strategies for these specific problems. Since our solver can optimize persistence pairs localized within a neighborhood of the field, we will also investigate divide-and-conquer parallelizations.

Acknowledgments

This work is partially supported by the European Commission grant ERC-2019-COG *"TORI"* (ref. 863464, https://erc-tori.github.io/), by the U.S. Department of Energy, Office of Science, under Award Number(s) DE-SC-0019039, and by a joint graduate research fellowship (ref. 320650) funded by the CNRS and the University of Arizona.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. doi: 10.48550/arXiv.1603.04467 6

[2] P. K. Agarwal, L. Arge, and K. Yi. I/O-efficient batched union-find and its applications to terrain analysis. In *SoCG*, pp. 167–176. ACM, New York, 2006. doi: 10.1145/1137856.1137884 1, 2

[3] K. Anderson, J. Anderson, S. Palande, and B. Wang. Topological data analysis of functional MRI connectivity in time and space domains. In *MICCAI Workshop on Connectomics in NeuroImaging*, pp. 67–77. Springer, Cham, 2018. doi: 10.1007/978-3-030-00755-3_8 1

[4] D. Attali, U. Bauer, O. Devillers, M. Glisse, and A. Lieutier. Homological reconstruction and simplification in R^3. In *SoCG*, pp. 117–126. ACM, New York, 2013. doi: 10.1145/2462356.2462373 2, 4, 8, 9

[5] D. Attali, M. Glisse, F. Lazarus, and D. Morozov. Persistence-Sensitive Simplification of Functions on Surfaces in Linear Time. In *TopoInVis*, 2009. 1, 2

[6] T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly*, 45(1):245–256, 1967. doi: 10.1080/00029890.1970.11992523 1

[7] S. Barannikov. Framed Morse complexes and its invariants. *Adv. Soviet Math.*, 21:93–116, 1994. doi: 10.1090/advsov/021/03 3

[8] U. Bauer, M. Kerber, and J. Reininghaus. Distributed computation of persistent homology. In *Algorithm Engin. and Exp.*, pp. 31–38. SIAM, Philadelphia, 2014. doi: 10.1137/1.9781611973198.4 1, 5

[9] U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. Phat - persistent homology algorithms toolbox. *J. Symb. Comput.*, 78:76–90, 2017. doi: 10.1016/j.jsc.2016.03.008 5

[10] U. Bauer, C. Lange, and M. Wardetzky. Optimal Topological Simplification of Discrete Functions on Surfaces. *Discrete Computational Geometry*, 47(2):347–377, 2012. doi: 10.1007/S00454-011-9350-Z 1, 2

[11] D. P. Bertsekas and D. Castañon. Parallel synchronous and asynchronous implementations of the auction algorithm. *Parallel Computing*, 17(6-7):707–732, 1991. doi: 10.1016/S0167-8191(05)80062-6 5, 6

[12] H. Bhatia, A. G. Gyulassy, V. Lordi, J. E. Pask, V. Pascucci, and P.-T. Bremer. Topoms: Comprehensive topological exploration for molecular and condensed-matter systems. *J. of Computational Chemistry*, 39(16):936–952, 2018. doi: 10.1002/JCC.25181 1

[13] S. Biasotti, D. Giorgio, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *TCS*, 392(1-3):5–22, 2008. doi: 10.1016/J.TCS.2007.10.018 1

[14] T. Bin Masood, J. Budin, M. Falk, G. Favelier, C. Garth, C. Gueunet, P. Guillou, L. Hofmann, P. Hristov, A. Kamakshidasan, C. Kappe, P. Klacansky, P. Laurin, J. Levine, J. Lukasczyk, D. Sakurai, M. Soler, P. Steneteg, J. Tierny, W. Usher, J. Vidal, and M. Wozniak. An Overview of the Topology ToolKit. In *TopoInVis*, pp. 327–342. Springer, Cham, 2019. doi: 10.1007/978-3-030-83500-2_16 1, 6

[15] A. Bock, H. Doraiswamy, A. Summers, and C. T. Silva. TopoAngler: Interactive Topology-Based Extraction of Fishes. *IEEE TVCG*, 24(1):812–821, 2018. doi: 10.1109/TVCG.2017.2743980 1

[16] P. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE TVCG*, 10(4):385–396, 2004. doi: 10.1109/TVCG.2004.3 2

[17] P. Bremer, G. Weber, J. Tierny, V. Pascucci, M. Day, and J. Bell. Interactive exploration and analysis of large scale simulations using topology-based data segmentation. *IEEE TVCG*, 17(9):1307–1324, 2011. doi: 10.1109/TVCG.2010.253 1

[18] H. Carr. *Topological Manipulation of Isosurfaces*. PhD thesis, University of British Columbia, 2004. 2

[19] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Symp. on Dis. Alg.*, 9 pages, pp. 918–926. SIAM, Philadelphia, 2000. doi: 10.1016/S0925-7721(02)00093-7 1

[20] H. Carr, G. Weber, C. Sewell, and J. Ahrens. Parallel peak pruning for scalable SMP contour tree computation. In *IEEE LDAV*, pp. 75–84. IEEE, Baltimore, 2016. doi: 10.1109/LDAV.2016.7874312 1

[21] H. A. Carr, J. Snoeyink, and M. van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *VIS*, pp. 497–504. IEEE, Austin, 2004. doi: 10.1109/VISUAL.2004.96 1, 4

[22] M. Carrière, F. Chazal, M. Glisse, Y. Ike, H. Kannan, and Y. Umeda. Optimizing persistent homology based functions. In *ICML*, pp. 1294–1303. PMLR, 2021. doi: 10.48550/arXiv.2010.08356 2, 4, 5, 6

[23] M. Carrière, M. Cuturi, and S. Oudot. Sliced wasserstein kernel for persistence diagrams. In *ICML*, 10 pages, pp. 664–673. JMLR.org, 2017. doi: 10.48550/arXiv.1706.03358 5

[24] E. W. Chambers, T. Ju, D. Letscher, M. Li, C. N. Topp, and Y. Yan. Some heuristics for the homological simplification problem. pp. 353–359. CCCG, Ontario, 2018. 2, 8

[25] D. Cohen-Steiner, H. Edelsbrunner, and D. Morozov. Vines and vineyards by updating persistence in linear time. In *SoCG*, 8 pages, pp. 119–126. ACM, New York, 2006. doi: 10.1145/1137856.1137877 6

[26] D. Davis, D. Drusvyatskiy, S. M. Kakade, and J. D. Lee. Stochastic Subgradient Method Converges on Tame Functions. *FoCM*, 20(1):119–154, 2020. doi: 10.1007/S10208-018-09409-5 2, 4

[27] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. AMS, 2009. 1, 2, 3

[28] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological Persistence and Simplification. *Discrete Computational Geometry*, 28(4):511–533, 2002. doi: 10.1007/S00454-002-2885-2 1, 3, 4, 6

[29] H. Edelsbrunner, D. Morozov, and V. Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *SoCG*, pp. 127–134. ACM, New York, 2006. doi: 10.1145/1137856.1137878 1, 2

[30] H. Edelsbrunner and E. P. Mucke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. on Graphics*, 9(1):66–104, 1990. doi: 10.1145/77635.77639 3

[31] R. Forman. A User's Guide to Discrete Morse Theory. *AM*, 48:B48c–35, 1998. 2, 6, 7

[32] P. Frosini and C. Landi. Size theory as a topological tool for computer vision. *Pattern Recognition and Image Analysis*, 9(4):596–603, 1999. 3

[33] R. B. Gabrielsson, V. Ganapathi-Subramanian, P. Skraba, and L. J. Guibas. Topology-Aware Surface Reconstruction for Point Clouds. *CGF*, 39(5):197–207, 2020. doi: 10.1111/CGF.14079 2

[34] Y. I. Gingold and D. Zorin. Controlled-topology filtering. *Comput. Aided Des.*, 39(8):676–684, 2007. doi: 10.1016/J.CAD.2007.05.010 2

[35] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based augmented merge trees with fibonacci heaps. In *IEEE LDAV*, pp. 6–15. Los Alamitos, 2017. doi: 10.1109/LDAV.2017.8231846 1

[36] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-Based Augmented Contour Trees with Fibonacci Heaps. *IEEE TPDS*, 30(8):1889–1905, 2019. doi: 10.1109/TPDS.2019.2898436 1

[37] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based Augmented Reeb Graphs with Dynamic ST-Trees. In *EGPGV*, pp. 27–37. EG, Eindhoven, 2019. doi: 10.2312/PGV.20191107 1

[38] P. Guillou, J. Vidal, and J. Tierny. Discrete Morse Sandwich: Fast Computation of Persistence Diagrams for Scalar Data – An Algorithm and A Benchmark. *IEEE TVCG*, 30(4):1897–1915, 2023. doi: 10.1109/TVCG.2023.3238008 1, 3, 5, 6, 7, 9

[39] D. Günther, A. Jacobson, J. Reininghaus, H. Seidel, O. Sorkine-Hornung, and T. Weinkauf. Fast and Memory-Efficienty Topological Denoising of 2D and 3D Scalar Fields. *IEEE TVCG*, 20(12):2585–2594, 2014. doi: 10.1109/TVCG.2014.2346432 2

[40] D. Günther, J. Reininghaus, H. Seidel, and T. Weinkauf. Notes on the simplification of the morse-smale complex. In *TopoInVis*, pp. 135–150. Springer, Berlin, 2013. doi: 10.1007/978-3-319-04099-8_9 2, 7, 8, 9

[41] A. Gyulassy. *Combinatorial construction of Morse-Smale complexes for data analysis and visualization*. PhD thesis, UC Davis, 2008. 1, 2

[42] A. Gyulassy, P. Bremer, R. Grout, H. Kolla, J. Chen, and V. Pascucci. Stability of dissipation elements: A case study in combustion. *CGF*, 33(3):51–60, 2014. doi: 10.1111/CGF.12361 1

[43] A. Gyulassy, P. Bremer, B. Hamann, and V. Pascucci. Practical considerations in morse-smale complex computation. In *TopoInVis*, pp. 67–78. Springer, Berlin, 2009. doi: 10.1007/978-3-642-15014-2_6 2, 9

[44] A. Gyulassy, P. Bremer, and V. Pascucci. Shared-Memory Parallel Computation of Morse-Smale Complexes with Improved Accuracy. *IEEE TVCG*, 25(1):1183–1192, 2019. doi: 10.1109/TVCG.2018.2864848 1

[45] A. Gyulassy, P. T. Bremer, B. Hamann, and V. Pascucci. A practical approach to Morse-Smale complex computation: Scalability and generality. *IEEE TVCG*, 14(6):1619–1626, 2008. doi: 10.1109/TVCG.2008.110 1

[46] A. Gyulassy, M. A. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically Clean Distance Fields.

IEEE TVCG, 13(6):1432–1439, 2007. doi: 10.1109/TVCG.2007.70603 8

[47] A. Gyulassy, A. Knoll, K. Lau, B. Wang, P. Bremer, M. Papka, L. A. Curtiss, and V. Pascucci. Interstitial and Interlayer Ion Diffusion Geometry Extraction in Graphitic Nanosphere Battery Materials. *IEEE TVCG*, 22(1):916–925, 2016. doi: 10.1109/TVCG.2015.2467432 1

[48] A. Gyulassy, V. Natarajan, V. Pascucci, P. Bremer, and B. Hamann. Topology-based simplification for feature extraction from 3d scalar fields. In *VIS*, pp. 535–542. IEEE, 2005. doi: 10.1109/VISUAL.2005.1532839 8

[49] H. Freudenthal. Simplizialzerlegungen von beschrankter Flachheit. *Ann. of Math.*, 43(3):580–583, 1942. doi: 10.48550/arXiv.2302.11922 3

[50] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. *CGF*, 35(3):643–667, 2016. doi: 10.1111/CGF.12933 1

[51] P. Hu, S. Boorboor, J. Marino, and A. E. Kaufman. Geometry-aware planar embedding of treelike structures. *IEEE TVCG*, 29(7):3182–3194, 2022. doi: 10.1109/TVCG.2022.3153871 7

[52] H.W. Kuhn. Some combinatorial lemmas in topology. *IBM JoRD*, 4(5):518–524, 1960. doi: 10.1147/RD.45.0518 3

[53] F. Iuricich. Persistence cycles for visual exploration of persistent homology. *IEEE TVCG*, 28(12):4966–4979, 2021. doi: 10.1109/TVCG.2021.3110663 1, 7, 9

[54] F. Iuricich, U. Fugacci, and L. D. Floriani. Topologically-consistent simplification of discrete morse complex. *Comput. Graph.*, 51:157–166, 2015. doi: 10.1016/J.CAG.2015.05.007 8

[55] J. Kasten, J. Reininghaus, I. Hotz, and H. Hege. Two-dimensional time-dependent vortex regions based on the acceleration magnitude. *IEEE TVCG*, 17(12):2080–2087, 2011. doi: 10.1109/TVCG.2011.249 1

[56] M. Kerber, D. Morozov, and A. Nigmetov. Geometry helps to compare persistence diagrams. *ACM J. of Experimental Algorithmics*, 22:1–20, 2017. doi: 10.1145/3064175 5, 6

[57] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 13 pages. Ithaca, New York, 2015. doi: 10.48550/arXiv.1412.6980 2, 4, 5, 7

[58] P. Klacansky. Open Scientific Visualization Data Sets. https://klacansky.com/open-scivis-datasets/, 2020. 6

[59] J. Lukasczyk, C. Garth, R. Maciejewski, and J. Tierny. Localized topological simplification of scalar data. *IEEE TVCG*, 27(2):572–582, 2020. doi: 10.1109/TVCG.2020.3030353 1, 2

[60] J. Lukasczyk, M. Will, F. Wetzels, G. H. Weber, and C. Garth. ExTreeM: Scalable Augmented Merge Tree Computation via Extremum Graphs. *IEEE TVCG*, 30(1):1085–1094, 2024. doi: 10.1109/TVCG.2023.3326526 1

[61] Y. Luo and B. J. Nelson. Accelerating iterated persistent homology computations with warm starts. *Computational Geometry*, 120:102089, 2024. doi: 10.1016/j.comgeo.2024.102089 6

[62] R. G. C. Maack, J. Lukasczyk, J. Tierny, H. Hagen, R. Maciejewski, and C. Garth. Parallel computation of piecewise linear morse-smale segmentations. *IEEE TVCG*, 30(4):1942–1955, 2023. doi: 10.1109/TVCG.2023.3261981 1

[63] D. Maljovec, B. Wang, P. Rosen, A. Alfonsi, G. Pastore, C. Rabiti, and V. Pascucci. Topology-inspired partition-based sensitivity analysis and visualization of nuclear simulations. In *PacificViz*, pp. 64–71. IEEE, Taipei, 2016. doi: 10.1109/PACIFICVIS.2016.7465252 1

[64] C. Maria, J. Boissonnat, M. Glisse, and M. Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *Mathematical Software*, pp. 167–174. Springer, 2014. doi: 10.1007/978-3-662-44199-2_28 6

[65] J. Marino and A. E. Kaufman. Planar visualization of treelike structures. *IEEE TVCG*, 22(1):906–915, 2016. doi: 10.1109/TVCG.2015.2467413 7

[66] J. Munkres. Algorithms for the assignment and transportation problems. *J. of SIAM*, 5(1):32–38, 1957. doi: 10.1137/0105003 5

[67] F. Nauleau, F. Vivodtzev, T. Bridel-Bertomeu, H. Beaugendre, and J. Tierny. Topological Analysis of Ensembles of Hydrodynamic Turbulent Flows – An Experimental Study. In *IEEE LDAV*, pp. 1–11. Los Alamitos, 2022. doi: 10.1109/LDAV57265.2022.9966403 1

[68] X. Ni, M. Garland, and J. C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics*, 23(3):613–622, 2004. doi: 10.1145/1015706.1015769 2

[69] A. Nigmetov, A. S. Krishnapriyan, N. Sanderson, and D. Morozov. Topological regularization via persistence-sensitive optimization. *Comp. Geom.*, 120:102086, 2024. doi: 10.1016/j.comgeo.2024.102086 2

[70] A. Nigmetov and D. Morozov. Topological optimization with big steps. *Discrete & Computational Geometry*, 72:310–344, 2022. doi: 10.1007/s00454-023-00613-x 2, 6

[71] M. Olejniczak, A. S. P. Gomes, and J. Tierny. A Topological Data Analysis Perspective on Non-Covalent Interactions in Relativistic Calculations. *IJQC*, 120(8):e26133, 2019. doi: 10.1002/qua.26133 1

[72] M. Olejniczak and J. Tierny. Topological Data Analysis of Vortices in the Magnetically-Induced Current Density in LiH Molecule. *PCCP*, 25:5942–5947, 2023. doi: 10.1039/D2CP05893F 1

[73] V. Pascucci, G. Scorzelli, P. T. Bremer, and A. Mascarenhas. Robust online computation of Reeb graphs: simplicity and speed. *ACM Transactions on Graphics*, 26(3):58, 2007. doi: 10.1145/1276377.1276449 1, 2

[74] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 12 pages. Curran Associates Inc., Red Hook, 2019. doi: 10.48550/arXiv.1912.01703 6

[75] G. Patanè and B. Falcidieno. Computing smooth approximations of scalar functions with constraints. *Comput. Graph.*, 33(3):399–413, 2009. doi: 10.1016/J.CAG.2009.03.014 2

[76] A. Poulenard, P. Skraba, and M. Ovsjanikov. Topological function optimization for continuous shape matching. *CGF*, 37(5):13–25, 2018. doi: 10.1111/CGF.13487 2, 5

[77] V. Robins. Toward computing homology from finite approximations. *Topology Proceedings*, 24(1):503–532, 1999. 3

[78] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images. *IEEE Trans. PAMI*, 33(8):1646–1658, 2011. doi: 10.1109/TPAMI.2011.95 1, 3, 6

[79] N. Shivashankar and V. Natarajan. Parallel Computation of 3D Morse-Smale Complexes. *CGF*, 31(3):965–974, 2012. doi: 10.1111/J.1467-8659.2012.03089.X 1

[80] N. Shivashankar, P. Pranav, V. Natarajan, R. van de Weygaert, E. P. Bos, and S. Rieder. Felix: A topology based framework for visual exploration of cosmic filaments. *IEEE TVCG*, 22(6):1745–1759, 2016. doi: 10.1109/TVCG.2015.2452919 1, 8

[81] P. Soille. Optimal Removal of Spurious Pits in Digital Elevation Models. *WRR*, 40(12):W12509, 2004. doi: 10.1029/2004WR003060 1, 2

[82] M. Soler, M. Petitfrere, G. Darche, M. Plainchault, B. Conche, and J. Tierny. Ranking Viscous Finger Simulations to an Acquired Ground Truth with Topology-Aware Matchings. In *IEEE LDAV*, pp. 62–72. Los Alamitos, 2019. doi: 10.1109/LDAV48142.2019.8944365 1

[83] E. Solomon, A. Wagner, and P. Bendich. A fast and robust method for global topological functional optimization. In *AISTATS*, pp. 109–117. PMLR, Cambridge, 2021. doi: 10.48550/arXiv.2009.08496 2

[84] T. Sousbie. The Persistent Cosmic Web and its Filamentary Structure: Theory and Implementations. *Royal Astronomical Society*, 414:384–403, 2011. doi: 10.1111/j.1365-2966.2011.18394.x 1, 8

[85] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. *IEEE TVCG*, 24(1):832–842, 2017. doi: 10.1109/TVCG.2017.2743938 1, 6

[86] J. Tierny and V. Pascucci. Generalized topological simplification of scalar fields on surfaces. *IEEE TVCG*, 18(12):2005–2013, 2012. doi: 10.1109/TVCG.2012.228 1, 2

[87] TTK Contributors. TTK Data. https://github.com/topology-tool-kit/ttk-data/, 2020. 6, 9

[88] TTK Contributors. TTK Online Example Database. https://topology-tool-kit.github.io/examples/, 2022. 1

[89] J. Vidal, J. Budin, and J. Tierny. Progressive Wasserstein Barycenters of Persistence Diagrams. *IEEE TVCG*, 26(1):151–161, 2020. doi: 10.1109/TVCG.2019.2934256 5

[90] T. Weinkauf, Y. I. Gingold, and O. Sorkine. Topology-based Smoothing of 2D Scalar Fields with C^1-Continuity. *CGF*, 29(3):1221–1230, 2010. doi: 10.1111/J.1467-8659.2009.01702.X 2

[91] D. Zeng, E. W. Chambers, D. Letscher, and T. Ju. To cut or to fill: a global optimization approach to topological simplification. *ACM Trans. on Graphics*, 39(6):201, 2020. doi: 10.1145/3414685.3417854 8

[92] D. Zeng, E. W. Chambers, D. Letscher, and T. Ju. Topological simplification of nested shapes. *CGF*, 41(5):161–173, 2022. doi: 10.1111/CGF.14611 8

[93] A. J. Zomorodian. Topology for computing. In M. J. Atallah and M. Blanton, eds., *Algorithms and Theory of Computation Handbook (Second Edition)*, chap. 3, pp. 82–112. CRC Press, Boca Raton, 2010. doi: 10.1017/CBO9780511546945 1, 2