Localized Topological Simplification of Scalar Data

Jonas Lukasczyk, Christoph Garth, Ross Maciejewski, Julien Tierny

Localized Topological Simplification of Scalar Data

Jonas Lukasczyk, Christoph Garth, Ross Maciejewski, and Julien Tierny

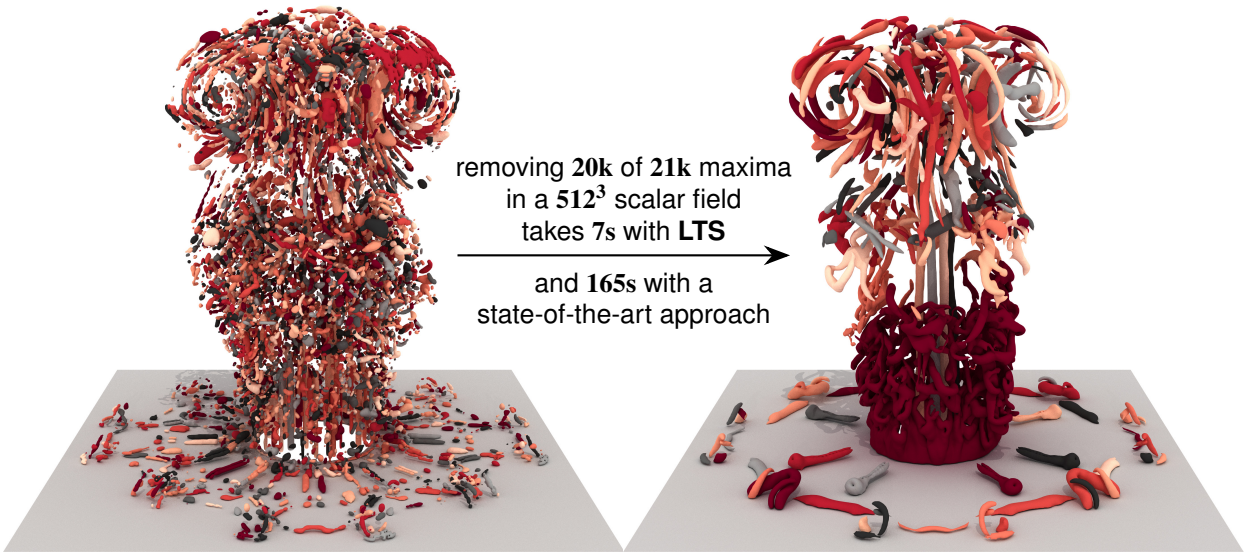


Fig. 1. Illustration of the impact of localized topological simplification (LTS) on the resulting topology-based feature characterization of individual vortices (colored regions) in a computational fluid dynamics simulation [25]. The simulation models the injection of a jet into a medium at rest (512^3 voxels) where individual vortices are characterized as regions around vorticity magnitude maxima. By selecting maxima that need to be removed via different persistence thresholds for different parts of the domain, it is possible to suppress noise, and to combine multiple small vortices into larger regions; allowing users to explore different merging thresholds. In this example, even for the aggressive removal of $20k$ out of $21k$ maxima, LTS only needs to process 1% of the scalar field. Moreover, due to the algorithm's localized nature, LTS can utilize shared-memory parallelism to outperform previous approaches by an order of magnitude. In this example, exploring a different selection of maxima with the simplification algorithm available in the Topology ToolKit [55] takes roughly $165s$, whereas LTS only requires $7s$. This significant speedup enables a higher level of interactivity for data visualization and analysis, which is essential if simplification parameters and their effect are not known a priori.

Abstract— This paper describes a localized algorithm for the topological simplification of scalar data, an essential pre-processing step of topological data analysis (TDA). Given a scalar field f and a selection of extrema to preserve, the proposed localized topological simplification (LTS) derives a function g that is close to f and only exhibits the selected set of extrema. Specifically, sub- and superlevel set components associated with undesired extrema are first locally flattened and then correctly embedded into the global scalar field, such that these regions are guaranteed—from a combinatorial perspective—to no longer contain any undesired extrema. In contrast to previous global approaches, LTS only and independently processes regions of the domain that actually need to be simplified, which already results in a noticeable speedup. Moreover, due to the localized nature of the algorithm, LTS can utilize shared-memory parallelism to simplify regions simultaneously with a high parallel efficiency (70%). Hence, LTS significantly improves interactivity for the exploration of simplification parameters and their effect on subsequent topological analysis. For such exploration tasks, LTS brings the overall execution time of a plethora of TDA pipelines from minutes down to seconds, with an average observed speedup over state-of-the-art techniques of up to $\times 36$. Furthermore, in the special case where preserved extrema are selected based on topological persistence, an adapted version of LTS partially computes the persistence diagram and simultaneously simplifies features below a predefined persistence threshold. The effectiveness of LTS, its parallel efficiency, and its resulting benefits for TDA are demonstrated on several simulated and acquired datasets from different application domains, including physics, chemistry, and biomedical imaging.

Index Terms—Topological data analysis, scalar data, simplification, feature extraction.

1 Introduction

In many applications, datasets produced by acquisition or simulation currently reach unprecedented levels in terms of size and complexity. This motivates the design of advanced analysis tools capable of extracting the relevant information from such datasets, and supporting its interpretation through interactive visualization and analysis. This is the purpose of Topological Data Analysis (TDA) [19], which provides a family of generic, robust, and efficient techniques for the extraction of the inherent structural information in the data. It has been successfully applied over the last two decades in a number of visualization and analysis tasks [37]. Popular applications include as-

- *Jonas Lukasczyk and Ross Maciejewski are with Arizona State University. E-mail: jl@jluk.de and rmacieje@asu.edu.*
- *Christoph Garth is with Technische Universität Kaiserslautern. E-mail: garth@cs.uni-kl.de.*
- *Julien Tierny is with Sorbonne Université and CNRS. E-mail: julien.tierny@sorbonne-universite.fr.*

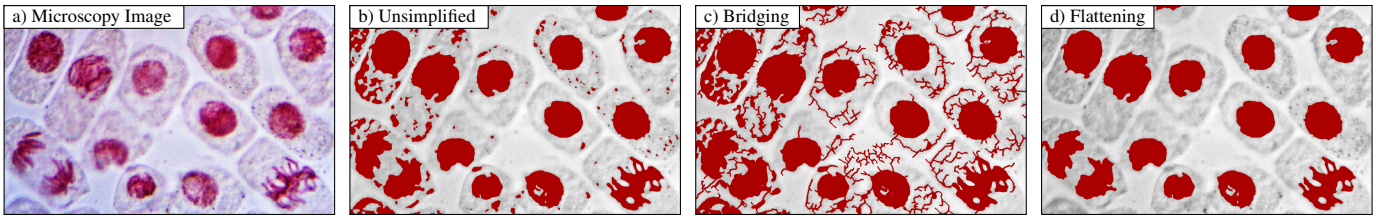


Fig. 2. Extraction of plant cell nuclei depicted in a microscopy image [49] (a) via sublevel set components of the color brightness scalar field that has been simplified based on different strategies (b-d). Without any simplification (b), the extracted cells contain numerous false positives in the form of undesired components that result from non-persistent extrema. Bridging (c) removes undesired extrema only in a topological sense, by creating bridges from undesired extrema towards desired ones, which simply attaches undesired to desired components via noticeable line artifacts. Flattening (d) completely removes components that are associated with undesired extrema while preserving the shape of desired components.

trophysics [50, 54], biological imaging [2, 9, 15], chemistry [7, 27, 44], fluid dynamics [10, 38, 40], material sciences [35, 36, 41, 53], and turbulent combustion [12, 32]. For scalar data, TDA introduces a number of topological abstractions that capture various types of structural features, such as the persistence diagram [20], the contour tree [14, 29], the Reeb graph [8, 45, 47], and the Morse-Smale complex [18, 33, 34, 48].

An important aspect of TDA is its ability to provide multi-scale representations of the aforementioned abstractions, which enables users to distinguish noise from features (Figs. 1 and 2). These representations are a critical component in many applications to support multi-scale analysis and visualization. In particular, several importance measures have been introduced to estimate the relevance of the features of interest represented by the *critical points* of the scalar field (Sec. 3.1). Such measures include topological persistence [20] and geometrical measures on level sets [15]. Moreover, as the notion of a feature of interest greatly depends on the application, users often characterize the importance of extrema with ad-hoc importance measures. For instance, Carr et al. [15] showed that importance measures based on a hyper-volume were often more effective for medical data than topological persistence, and Guenther et al. [27] demonstrated that critical points corresponding to relevant chemical interactions can be isolated by combined thresholding of multiple chemical quantitites. In general, once an importance measure has been established, multi-scale representations in TDA can be obtained either in *(i)* a post-processing step (by iteratively simplifying the computed abstractions), or in *(ii)* a pre-processing step (by simplifying the data *before* computing the topological abstractions). In the first case *(i)*, a tailored algorithm based on iterative cancellations must be derived for each specific topological abstraction. In the second case *(ii)*, the values of the original scalar field have to be altered in a pre-processing step such that the resulting simplified scalar field only exhibits extrema deemed relevant by some importance measure while still being close to the original scalar field. Although the latter strategy requires a perturbation of the original scalar field, it has the advantage of being generic (since any importance measure can be used to classify critical points) and agnostic (since the simplified scalar field can be used like any other scalar field as an input for subsequent TDA). This strategy decouples the simplification from the topological abstractions themselves, which enables software frameworks—such as the Topology ToolKit (TTK) [55]—to mutualize and modularize implementations of different algorithms. Specifically, all topological abstractions computed with TTK on the simplified field—e.g. critical points [5], merge [28] and contour trees [29], Reeb graphs [30], and Morse-Smale complexes [55]—benefit from the same pre-simplification without being aware that the input field was actually simplified. This is particularly useful in advanced analysis scenarios comining multiple abstractions [27, 40]. Finally, the pre-simplification of the scalar field does not prohibit further post-simplification of the computed abstractions. For these reasons, this paper focuses on the pre-simplification of scalar data.

In addition to the approaches introduced for the persistence-driven topological simplification of scalar data [4, 6, 21], Tierny and Pascucci proposed a generic algorithm that supports arbitrary importance measures [56]. These simplification algorithms can again be classified into two strategies: bridging (a.k.a. carving) and flattening (a.k.a. flooding). In short, bridging connects undesired contours to desired ones via small bridges (technically, along separatrices), thus only removing them in a topological sense (Fig. 2c). In contrast, flattening completely removes contours induced by undesired extrema while preserving the shape of desired contours (Fig. 2d). However, both strategies are limited to the removal of extremum-saddle pairs, which notably excludes saddle-saddle pair removal in 3D (Sec. 5.3). Additionally, the two strategies have an impact on feature geometry as they alter the underlying scalar field. In particular, bridging adds line artifacts between contours, and flattening introduces flat-plateaus that also impact contained integral lines. But most importantly, despite the acceptable asymptotic complexity (linearithmic time) of both strategies, existing algorithms suffer from performance issues due to their global and sequential nature. Specifically, simplification can empirically account for up to 90% of overall computation time in many TDA pipelines (Sec. 5.2).

This paper revisits the problem of topological simplification of scalar data to address this performance issue. Given a scalar field f and a selection of extrema to preserve, the proposed algorithm creates, based on localized flattening, a function g that only exhibits the selected set of extrema and has a small distance $||f-g||_\infty$ for data fitting purposes. Moreover, the proposed localized topological simplification (LTS) algorithm (Sec. 4) is output-sensitive as it only visits the regions of the data where simplification is needed; resulting in an immediate speedup over previous approaches. Due to its localized nature, it can be parallelized, and we present—to our knowledge—the first parallel approach to topological simplification of scalar data. We provide performance benchmarks obtained with our OpenMP [17] implementation that achieves a high parallel efficiency (70%). For the case where the considered importance measure is topological persistence, we present a variant of our framework adapted to persistence-driven simplification. Given an input threshold ε, we show how to efficiently combine LTS with a partial computation of the persistence diagram. For applications where a relevant value of ε can be estimated a priori, this further improves performance. Extensive experiments on synthetic and real-world data on a commodity workstation report an observed speedup of $\times 36$ compared to previous approaches. This significant improvement is illustrated in various TDA scenarios (Sec. 5.2), where LTS brings the overall analysis time from minutes down to seconds, hence enabling interactive multi-scale exploration of the features present in the data.

Contributions

1) Localized Topological Simplification (LTS) (Sec. 4) In contrast to state-of-the-art approaches, which visit the entire input domain, LTS only processes the regions of the domain where the data actually needs simplification. This makes LTS output sensitive, which is particularly relevant for the problem of topological simplification, as noise frequently corresponds to a small fraction of the input domain. In the presented experiments (Sec. 5), this already yields an average speedup over previous, global approaches between $\times 3$ and $\times 5$.

2) Parallel Topological Simplification (Sec. 4.4) LTS can be efficiently parallelized due to its localized nature. The presented experiments demonstrate that a shared-memory parallel implementation based on OpenMP yields a high parallel efficiency on real-life datasets (70%). On a commodity workstation, this results in an overall speedup of $\times 36$ over previous, sequential approaches.

3) Parallel Persistence-Driven Simplification (Sec. 4.5) For the special case where extrema are selected based on topological persistence, an adapted version of LTS partially computes the persistence diagram and simultaneously simplifies all features below a given persistence threshold. For applications where persistence is a relevant criterion and a threshold is known a priori, this specialized algorithm further improves performances. This scenario is particularly relevant for *batch* and *in situ* processing, where noise below a conservative threshold can be removed prior to any analysis.

4) Reference Implementation (additional material) We provide a reference C++ implementation of all presented algorithms that can be used to replicate the experiments of Sec. 5 and to perform benchmarks.

2 Related Work

As described before, topological abstractions can be simplified in a post-process *(i)* with a tailored algorithm specific to each abstraction. Such algorithms have been introduced for the contour tree [15], Reeb graph [45], and Morse-Smale complex [34]. Our approach focuses on the *pre*-simplification of data *(ii)*—i.e., prior to the computation of any topological abstraction. Existing methods for data pre-simplification can be classified in the following two categories.

Numerical methods simplify an input scalar field given some constraints on the extrema to preserve, by computing a numerically optimized solution. Such methods usually incorporate the extrema to preserve as hard constraints, while optimizing a geometrical criterion, such as smoothness. Bremer et al. [11] introduced the first technique in this line of work in which a simplified Morse-Smale complex drives a per-cell, iterative simplification of the data via Laplacian smoothing. This method requires computing and simplifying the Morse-Smale complex, which can be computationally expensive. Moreover, the method can require many time-intensive Laplacian iterations to complete. Weinkauf et al. [59] extend the work of Bremer et al. [11] by utilizing bi-Laplacian optimization to additionally enforce the continuity of the gradient across the separatrices of the Morse-Smale complex. In geometry processing, several techniques have been introduced for the computation of smooth scalar fields given a small number of critical points serving as constraints [26, 43]. For instance, Patané et al. [46] introduce a technique combining least-squares approximation and Tikhonov regularization for the topology-driven simplification of scalar fields. These approaches can also be extended to 3D scalar fields [31]and to some extent to vector fields [57, 60].

Although numerical methods can produce smooth outputs that respect given topological constraints, they have two primary limitations. First, the iterative nature of the employed solvers results in long computation times (typically minutes), which prohibits the interactive exploration of different simplification parameters. Second, they are prone to numerical instabilities that can result from an unsuitable triangulation of the input domain or the numerical sensitivity of the geometrical operators used in the optimization. Tierny and Pascucci [56] illustrate that such instabilities frequently occur during Laplacian optimization and result in the presence of additional spurious critical points that prevent the solution from conforming strictly to the input constraints.

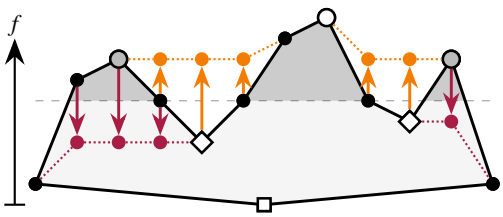


Fig. 3. Differences between removing undesired maxima (gray discs) while preserving desired maxima (white disc) from an input scalar field f via bridging (orange) and flattening (red). Bridging creates a monotone path from an undesired maximum along separatrices towards a desired maximum by *elevating* the value of the corresponding vertices (orange chains). It modifies the global shape of level sets by joining the three super-level set components (dark gray areas above dashed line) with a connected path (see also Fig. 2). In contrast, flattening *lowers* all the vertices of an undesired hill to the closest saddle. Hence, flattening completely removes the contours induced by undesired maxima while preserving the shape and topology of other contours. Here, only the unmodified super-level set component in the middle would remain.

Combinatorial methods, in contrast, are designed to generate outputs that are guaranteed by construction to conform to the input constraints. This category of techniques can be seen as complementary to numerical approaches, as combinatorial methods can be used to fix, in a post-process, the possible artifacts generated by numerical methods. The first approach for the combinatorial simplification of scalar data can be attributed to Edelsbrunner et al. for their work on persistence-driven simplification [21]. Given an input function f and its persistence diagram [20] (Sec. 3.3), the authors first simplify the diagram by removing all features below a persistence threshold ε. In addition to showing that a scalar function g strictly admitting this simplified diagram exists, they also introduce an algorithm to compute such a function g with a bounded error to the input ($||f-g||_\infty \leq \varepsilon$). Their work can be viewed as a generalization of prior works in terrain modeling, where only minimum-saddle persistence pairs were simplified [1, 13, 52]. Attali et al. [4] introduced a simplification algorithm for the case of filtrations of simplicial complexes. In the context of Discrete Morse theory [23], Bauer et al. [6] showed that persistence-driven simplification could be achieved with a combination of *flattening* and *bridging* that results in a minimized error bound $||f-g||_\infty$. Although these methods were initially introduced for two-dimensional domains, they apply readily for domains of higher dimensions. However, these methods—as well as our approach—do not support the removal of saddle-saddle pairs in 3D (see Sec. 6 for further discussion).

Despite the guaranteed correctness of their outputs, these combinatorial methods suffer from several limitations. First, they all use *bridging*, which introduces undesirable visual line artifacts in the output, where the bridged integral lines connect simplified features and therefore often stand out (Fig. 3, orange). This is particularly detrimental if the simplification is used as a pre-process to data segmentation. As shown in Fig. 2c, this can result in the identification of large regions that include the canceled features through a thin connection. In contrast, *flattening*-based methods completely remove undesired extrema by flattening their corresponding regions to the value of their paired saddle (Fig. 3, red). Therefore, flattening is better suited for data segmentation as simplified features are discarded from post-process segmentations (Fig. 2d). Second, methods that manipulate filtrations [4] or Discrete Morse functions [6] require an extra post-processing step to convert their output into a piecewise linear scalar field (which is the traditional representation for scalar data). In particular, this step involves the subdivision of the triangulation (one new vertex per d-simplex, with $d \geq 1$). This can increase the size of the triangulation up to an order of magnitude, which may not be acceptable in certain applications. Third, these approaches focus on the special case where the critical points to cancel are identified according to topological persistence. As discussed previously, many applications come with their own importance measures. To address this, Tierny and Pascucci [56] introduced a flattening-based approach, directly producing a piecewise linear scalar field on its output, and which supports the cancellation of an arbitrary selection of extrema. However, their technique follows a *white-list* approach that removes undesired extrema by globally constraining the sub- and superlevel set components of the desired extrema. This results in an intrinsically sequential algorithm, processing the data globally. In contrast, LTS follows a *black-list* approach by only simplifying the regions that correspond to undesired extrema. Moreover, the LTS approach can be efficiently parallelized, which results in a significant performance gain that enables interactive topological simplification.

3 Technical Background

This section details the technical background of the proposed methodology. We refer the reader to reference text books [19] for a comprehensive introduction to topological data analysis.

3.1 Critical Points of Piecewise Linear Scalar Fields

The input of the proposed methodology is a piecewise-linear (PL) scalar field $f : \mathcal{M} \to \mathbb{R}$, where real-valued data is given at the vertices of a PL d-manifold $\mathcal{M}$, and values inside higher dimensional simplices are linearly interpolated via barycentric coordinates. In this work, we additionally require that every PL scalar field is also injective on the vertices of $\mathcal{M}$, which can be enforced for any input scalar field with a symbolic perturbation inspired by *Simulation of Simplicity* [22]. This can be achieved by sorting the list of vertices $\mathcal{V}$ in increasing f order, and enforcing that two consecutive values are strictly monotonically increasing (by the addition of an arbitrarily small constant, see Sec. 4.1).

The topological features of f can be tracked with the notion of the so-called *sublevel* set, noted $f^{-1}_{-\infty}(w) = \{p \in \mathcal{M} \mid f(p) < w\}$, which is defined as the pre-image of the interval $(-\infty, w)$ by f. Symmetrically, the *superlevel* set is defined as $f^{-1}_{+\infty}(w) = \{p \in \mathcal{M} \mid f(p) > w\}$. In the smooth setting, the topology of these sets (in 3D their connected components, cycles, and voids) can only change at specific locations, named the *critical points* of f [42]. In the PL setting, Banchoff [5] introduced a local characterization of critical points based on their *star* and *link*. The *star* $St(v)$ of a vertex $v \in \mathcal{M}$ is the set of its co-faces: $St(v) = \{\sigma \in \mathcal{M} \mid v < \sigma\}$. The *link* $Lk(v)$ of v consists of the faces τ of the simplices $\sigma \in St(v)$ with an empty intersection with v: $Lk(v) = \{\tau \in \mathcal{M} \mid \tau < \sigma,\ \sigma \in St(v),\ \tau \cap v = \emptyset\}$. The *lower link* $Lk^-(v)$ of v consists of the simplices of $Lk(v)$ lower than $f(v)$: $Lk^-(v) = \{\sigma \in Lk(v) \mid \forall u \in \sigma,\ f(u) < f(v)\}$. The *upper link* is defined symmetrically: $Lk^+(v) = \{\sigma \in Lk(v) \mid \forall u \in \sigma,\ f(u) > f(v)\}$. The lower and upper links of a vertex v are illustrated in red and orange in Fig. 4. A vertex v is *regular* (Fig. 4a) if and only if both $Lk^-(v)$ and $Lk^+(v)$ are simply connected and not empty. For such vertices, the sub- and superlevel sets do not change their topology as they span $St(v)$. Otherwise, v is called a *critical point* of f [5]. These can be classified with regard to their *index* $\mathcal{I}(v)$, which is equal to 0 for local minima ($Lk^-(v) = \emptyset$, Fig. 4b), to d for local maxima ($Lk^+(v) = \emptyset$, Fig. 4c), and otherwise to i for i-saddles ($0 < i < d$, Fig. 4d). Saddles for which the number of connected component of $Lk^-(v)$ or $Lk^+(v)$ is greater than 2 are called *degenerate*. All other saddles are *simple*. As discussed in Sec. 4, the proposed algorithm handles degenerate saddles implicitly.

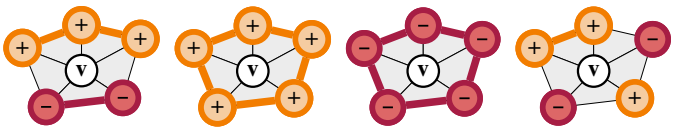
3.2 General Topological Simplification

Let $\mathcal{C}_f$ be the set of critical points of f. Tierny and Pascucci [56] introduce the notion of *general topological simplification* as follows:

Definition 1 (General Topological Simplification)**.** *Given a PL scalar field $f : \mathcal{M} \to \mathbb{R}$, a* general topological simplification *of f is a PL scalar field $g : \mathcal{M} \to \mathbb{R}$ such that the critical points of g form a sub-set of $\mathcal{C}_f$, i.e., $\mathcal{C}_g \subseteq \mathcal{C}_f$ with identical indices $\mathcal{I}$ and locations.*

In other words, a general topological simplification g is a variant of a scalar field f, from which critical points have been removed. In most applications, it is also desirable that g remains close to the input f (with a small distance $||f - g||_\infty$) for data fitting purposes.

Tierny and Pascucci [56] discuss the possible critical point removals. Based on the analysis of the connectivity of the sublevel sets of f, they show that the removal of a minimum m is necessarily accompanied with the removal of a simple saddle s, where $f^{-1}_{-\infty}$ changes its number of connected components. Symmetrically, the removal of a maximum induces the removal of a simple saddle. Thus, they introduce an approach for topological simplification which is based on the constrained construction of the sublevel sets of g, where the connectivity of $g^{-1}_{-\infty}$ is globally controlled to enforce the preservation of minimum and maximum constraints, respectively noted $\mathcal{C}^0_g$ and $\mathcal{C}^d_g$. While this discussion is mostly carried out for 2-dimensional domains, this approach to extremum removal readily applies to domains of higher dimensions and has been implemented as a default simplification mechanism in the *"Topology ToolKit"* open-source library [55]. Our methodology follows a compatible workflow and drives the simplification by user specified sets of extrema to maintain or remove.



a) Regular Point b) Minimum c) Maximum d) Saddle Point

Fig. 4. Vertex classification: if the lower and upper link of a vertex v (red and orange) are both connected, then v is regular (a). It is a minimum or maximum if it has no lower or upper link, respectively (b-c). If its lower or upper link consists of multiple components, then v is called a saddle (d).

3.3 Topological Persistence

General topological simplifications are defined for arbitrary selections of extrema. In practice, these are often selected based on application dependent importance measures. Alternatively, *topological persistence* [20] is often used as an established, general-purpose importance measure. The intuition behind persistence consists of assessing the importance of a critical point, based on the lifetime of its induced feature in $f^{-1}_{+\infty}(w)$ as one continuously decreases the isovalue w. As w decreases, connected components of $f^{-1}_{+\infty}(w)$ appear and merge at the maxima and 2-saddles of f, respectively (Fig. 5a). The Elder rule [19] stipulates that if two connected components—created at the maxima m_0 and m_1 with $f(m_0) < f(m_1)$—meet at a given 2-saddle s, then the *youngest* of the two components (the one created last, at m_0) *dies* in favor of the *oldest* one (created at m_1). In this case, a *persistence pair* $\langle s, m_0 \rangle$ is created and its *topological persistence* p is given by $p(\langle s, m_0 \rangle) = f(m_0) - f(s)$. Each maximum m—with the exception of the global maximum—can be unambiguously paired following this strategy and can consequently be assigned a persistence value, noted $p(m)$. By convention, the global maximum is paired with the global minimum and therefore assigned a persistence equal to the function range.

Persistence pairs are usually visualized with the persistence diagram [19] (Fig. 5a, right), which embeds each pair $\langle a, b \rangle$ as a point in the 2D plane at location $\big(f(a), f(b)\big)$. There, the persistence of the pair can be readily visualized as the height of the point to the diagonal. In particular, features with a high persistence stand out by being far away from the diagonal (dark colored pairs), while noisy features are typically located in the vicinity of the diagonal (light colored pairs). The population of persistence pairs is also often visualized with the *persistence curve* (Sec. 5), which plots, for an increasing threshold ε, the number of pairs more persistent than ε. As described above, 2-saddle/maximum pairs characterize the lifetime of the connected components of $f^{-1}_{+\infty}(w)$. The symmetric reasoning can be applied in 3D to characterize, with minimum/1-saddle pairs, the life time of its voids, while the 1-saddle/2-saddle pairs characterize its independent cycles. Then, given a threshold ε, the notion of persistence-driven simplification is a special case of general simplification, such that $\forall m \in \mathcal{C}^0_g\ :\ p(m) \geq \varepsilon$ and $\forall m \in \mathcal{C}^d_g\ :\ p(m) \geq \varepsilon$.

4 Localized Topological Simplification (LTS)

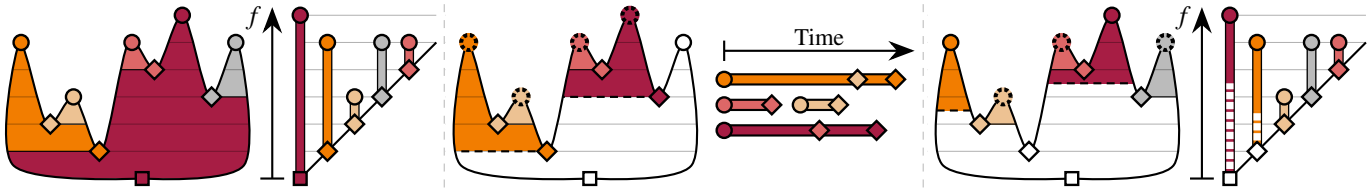
Given a PL scalar field f and a subset of its minima $\overline{\mathcal{C}^0_f}$ and maxima $\overline{\mathcal{C}^d_f}$, the proposed algorithm derives via localized flattening a function g that has a small deviation $||f - g||_\infty$ and only exhibits the specified extrema:

$$\mathcal{C}^0_g = \overline{\mathcal{C}^0_f} \subseteq \mathcal{C}^0_f \quad \text{and} \quad \mathcal{C}^d_g = \overline{\mathcal{C}^d_f} \subseteq \mathcal{C}^d_f. \tag{1}$$

The following description focuses on the removal of maxima, which can be imagined as the process of *"flattening hills"* in a terrain (Fig. 3, red). Minima are processed symmetrically.

4.1 Order-Based Representation

The critical point characterization presented in Sec. 3.1 classifies the vertices of $\mathcal{M}$ as regular or critical without any ambiguity as long as every vertex has a distinct value from all its neighbors, i.e., if there exists a non-ambiguous global vertex order. In this case, every link can be binarily partitioned into *lower* and *upper* link components (Fig. 4). Since this property does not hold for every PL scalar field $f : \mathcal{M} \to \mathbb{R}$, the proposed algorithm first derives based on f an



(a) Persistence Pair Segmentation (b) Parallel Superlevel Set Propagations (c) Persistence-Driven Propagations

Fig. 5. Critical point pairs and their corresponding domain segments (a) computed with superlevel set propagations (b) and persistence-driven propagations (c). Each maximum (disc), its paired saddle (diamond), and its corresponding domain segment (background) are shown with the same color. In the first step of LTS (b), superlevel set propagations determine—optionally in parallel—for each undesired maximum (dashed discs) its corresponding saddle and domain segment (Secs. 4.2-4.4). In the special case of persistence-driven simplification (c), non-persistent maxima are identified by initiating for all maxima superlevel set propagations that terminate as soon as they exceed a given persistence threshold (dashed lines). Thus, maxima paired with a saddle (dashed discs) do not exceed the persistence threshold and can subsequently be simplified (Sec. 4.5).

intermediate scalar field $\hat{f}$ that satisfies this condition via a specific variant of *Simulation of Simplicity* [22]. Next, the algorithm simplifies $\hat{f}$ to $\hat{g}$ based on the same extrema conditions as defined for f, and finally simplifies f to g in a post-processing step based on $\hat{g}$. The overall process is summarized in the inset commutative diagram.

$$\begin{array}{ccc} f:\mathcal{M}\to\mathbb{R} & \dashrightarrow & g:\mathcal{M}\to\mathbb{R} \\ \downarrow & & \uparrow \\ \hat{f}:\mathcal{M}\to\mathbb{N} & \longrightarrow & \hat{g}:\mathcal{M}\to\mathbb{N} \end{array}$$

Let $\mathcal{V}$ be the list of vertices of $\mathcal{M}$, sorted by increasing f values, where vertices with the same scalar value are disambiguated based on their original position in memory. Then, $\hat{f}:\mathcal{M}\to\mathbb{N}$ is the so-called *order field* of f that maps each vertex $v\in\mathcal{M}$ to its position $\hat{f}(v)$ in $\mathcal{V}$. By construction, *(i)* $\hat{f}$ is injective on the vertices of $\mathcal{M}$, and *(ii)* the critical points of $\hat{f}$ are a *superset* of the critical points of f. The first property *(i)* guarantees that now every vertex has a distinct value from its neighbors, which can thus be classified without any ambiguity. The second property *(ii)* is essential for topological simplification, which basically boils down to reordering vertices such that the resulting new order field $\hat{g}$ respects Eq. 1. To this end, note that every critical point of f is also a critical point of $\hat{f}$, but $\hat{f}$ might also have additional critical points that result form the disambiguation. These points, however, will be implicitly removed by LTS (as they do not belong to the set of extrema constraints $\overline{\mathcal{C}^0_{\hat{f}}}=\overline{\mathcal{C}^0_f}$ and $\overline{\mathcal{C}^d_{\hat{f}}}=\overline{\mathcal{C}^d_f}$), which will yield a new order field $\hat{g}$ whose critical points respect Eq. 1.

Finally, f can be turned into g in a post-process by enforcing that the numerical values of g are monotonically (and injectively) increasing with $\hat{g}$. This post-process monotonicity enforcement results in the characteristic *hill flattening* observed with flattening-based simplification.

4.2 Removal of an Individual Maximum

This section first details the proposed approach in a simple configuration where only one maximum m must be removed, i.e.,

$$\mathcal{C}^0_{\hat{g}}=\overline{\mathcal{C}^0_{\hat{f}}}=\mathcal{C}^0_{\hat{f}} \quad \text{and} \quad \mathcal{C}^d_{\hat{g}}=\overline{\mathcal{C}^d_{\hat{f}}}=\mathcal{C}^d_{\hat{f}}\setminus\{m\}. \tag{2}$$

To this end, the following procedure will iteratively transform the order field $\hat{f}$ to $\hat{g}$ in four steps. Fig. 6 provides a running example, where the task is to remove the maximum m, with initial value $\hat{f}(m)=22$.

1) Superlevel Set Component Computation As discussed in Sec. 3, when continuously decreasing an isovalue w, a connected component of $\hat{f}^{-1}_{+\infty}(w)$ is created at the local maximum m of $\hat{f}$. As w further decreases, this component eventually merges with another component at a saddle s. Let $\mathcal{M}^m_s$ denote the connected component of $\hat{f}^{-1}_{+\infty}\big(\hat{f}(s)\big)$ containing m. The vertex set $\mathcal{M}^m_s$ can be computed via a so-called *superlevel set propagation* that initializes $\mathcal{M}^m_s$ with m, and then iteratively adds the largest neighbor of $\mathcal{M}^m_s$ to $\mathcal{M}^m_s$ until a vertex v with a larger, unvisited neighbor n is added, i.e., $\hat{f}(n)>\hat{f}(v)$ and $n\notin\mathcal{M}^m_s$. This implies that n belongs to a distinct superlevel set component, and therefore v has to be a saddle (noted s above). For convenience s is considered to be an element of $\mathcal{M}^m_s$ in the remaining. *Sublevel set propagations* are defined symmetrically. In Fig. 6a, vertices are added to $\mathcal{M}^{22}_{\emptyset}$ (arrows, orange triangles) until a vertex (10) with a higher unvisited neighbor (23) is found. This vertex corresponds to the paired saddle s of m.

2) Localized Simplification To remove the maximum m while preserving all other maxima, the vertices of $\mathcal{M}^m_s$ need to be rearranged in a new global order $\hat{g}$ such that vertices outside $\mathcal{M}^m_s$ preserve their old order (Eq. 3), s is the only maximum of $\hat{g}$ restricted to $\mathcal{M}^m_s$ (Eq. 4), and all minima of $\hat{g}|\mathcal{M}^m_s$ are vertices with neighbors outside $\mathcal{M}^m_s$ (Eq. 5):

$$\forall u\in\mathcal{M} \text{ and } \forall v\in\mathcal{M}\setminus\mathcal{M}^m_s \;:\; \hat{f}(u)<\hat{f}(v)\to\hat{g}(u)<\hat{g}(v) \tag{3}$$

$$\mathcal{C}^d_{\hat{g}|\mathcal{M}^m_s} = \{s\} \tag{4}$$

$$\forall v\in\mathcal{C}^0_{\hat{g}|\mathcal{M}^m_s} \;:\; \exists n\in Lk(v),\, n\notin\mathcal{M}^m_s \tag{5}$$

As detailed in Step 3, if $\hat{g}$ satisfies Eqs. 3-5, then $\hat{g}$ also satisfies Eq. 2. A key insight is that enforcing these conditions on $\hat{g}|\mathcal{M}^m_s$ is itself a special case of topological simplification, localized to vertex values of $\mathcal{M}^m_s$. To simplify notations, let $\hat{l}_0:\mathcal{M}^m_s\to\mathbb{N}$ denote the local order of vertices of $\mathcal{M}^m_s$ (values of opaque vertices in Fig. 6b) induced by the initial global order field $\hat{f}$ (vertex values in Fig. 6a).

This localized simplification problem can be solved by adapting the iterative simplification algorithm of Tierny and Pasucci [56] to this special setting. To summarize, their algorithm takes as input constraints an explicit list of *authorized* extrema to preserve. Then, the algorithm alternates passes to remove *unauthorized* maxima and minima by respectively constraining the connectivity of the super- and sublevel set components of the authorized extrema. However, a maxima pass might introduce additional unauthorized minima that need to be removed with an additional minima pass, and vice versa. The authors show that iteratively alternating between minimum and maximum passes is guaranteed to converge to a global output order that only exhibits the authorized extrema.

However, in the localized setting, the notion of authorized extremum does not readily apply as l_0 does not necessarily exhibit other extrema than m. Thus, LTS first introduces an authorized maximum at s by setting its corresponding value to infinity, and then introduces an authorized minimum at some vertex $v\in\mathcal{M}^m_s$ different from s that admits a neighbor outside $\mathcal{M}^m_s$ by setting its corresponding value to negative infinity, i.e., $\hat{l}_0(s)\hookleftarrow+\infty$ and $\hat{l}_0(v)\hookleftarrow-\infty$ (Fig. 6b). During all iterations, LTS always preserves s as the only authorized maximum, and dynamically updates the set of authorized minima with the vertices which are minima on $\mathcal{M}^m_s$ *and* admit neighbors outside $\mathcal{M}^m_s$.

In the first iteration, the algorithm performs on $\mathcal{M}^m_s$ a superlevel set propagation initiated at the only authorized maximum (arrows of Fig. 6b). The inverse order in which vertices are added during this propagation is guaranteed to yield only one maximum: s. Thus, this inverse order becomes the new local order l_1 on $\mathcal{M}^m_s$ (Fig. 6c).

Next, if unauthorized minima are present—i.e., minima that do not admit neighbors outside $\mathcal{M}^m_s$—the algorithm initiates symmetrically a sublevel set propagation from all current authorized minima. In Fig. 6c, such a propagation is initiated at the authorized minimum 00 to remove the unauthorized minima 01 and 02. The order in which vertices are added during this propagation is guaranteed to omit the unauthorized minima, and therefore becomes the new local order l_2 (Fig. 6d). Then, LTS alternates between maxima and minima iterations until the last computed local order $\hat{l}$ only exhibits authorized extrema (Fig. 6e).

3) Local to Global Order At the end of the previous step, the last computed local order $\hat{l}$ is guaranteed to have only one maximum which is located at s, and all its minima are located next to a vertex outside $\mathcal{M}_s^m$. At this point, it is necessary to update the global order $\hat{f}$ to $\hat{g}$ by reordering the vertices of $\mathcal{M}_s^m$ in the global order such that s preserves its old position, and all other vertices of $\mathcal{M}_s^m$ form a contiguous segment, located immediately before s, and sorted by $\hat{l}$. The position of each vertex of $\mathcal{M}$ in the resulting new global order yields the new order field $\hat{g}$ that satisfies Eqs. 3-5, as detailed next.

Since s was originally a saddle there has to exist a neighbor $n \in \mathcal{M} \setminus \mathcal{M}_s^m$ of s with $\hat{f}(s) < \hat{f}(n)$, so from Eq. 3 follows that $\hat{g}(s) < \hat{g}(n)$, which makes s no longer a maximum on $\mathcal{M}$. Similarly, every minimum $v \in \mathcal{C}^0_{\hat{g}|\mathcal{M}_s^m}$ has a neighbor $n \in \mathcal{M} \setminus \mathcal{M}_s^m$ where $\hat{g}(n) < \hat{g}(v)$—this follows from the fact that $\mathcal{M}_s^m$ was computed by a superlevel set propagation—which makes v no longer a minimum on entire $\mathcal{M}$. Hence, all extrema of $\hat{g}|\mathcal{M}_s^m$ are not extrema of $\hat{g}|\mathcal{M}$, and all extrema outside $\mathcal{M}_s^m$ are preserved. Note that $\mathcal{M}_s^m$ may have contained minima that also got removed during the local simplification (as illustrated in Fig. 6); a known artifact of flattening-based simplification. If these minima are required to be preserved, their existence can be enforced in an optional post-process that makes them smaller than all their neighbors. It follows that $\hat{g}$ satisfies Eq. 2 as required, so the maximum m has truly been removed combinatorially.

4) Symbolic to Numerical Perturbation The previous process only changes the order of vertices such that the maximum m of $\hat{f}$ is no longer classified as an extremum of $\hat{g}$, while preserving all other extrema. This symbolic perturbation is sufficient for most TDA pipelines since the criticality of vertices only depends on their order (Sec. 4.1). In certain cases, however, it may be useful to reflect the new order in the original numerical values by deriving a new scalar field g based on $\hat{g}$. To this end, g is first initialized to f, and then every vertex of $\mathcal{M}$ is visited in decreasing value of $\hat{g}$. If during this process a visited vertex v has a higher g value than its predecessor v' (i.e., $g(v) > g(v')$), then its g value is updated to be smaller than its predecessor (i.e., $g(v) \leftarrow g(v') - \zeta$, with ζ arbitrarily small) which effectively flattens the hill $\mathcal{M}_s^m$. This also guarantees that g only deviates from f on $\mathcal{M}_s^m$ with at most the height of the hill, i.e., $||f - g||_\infty \leq f(m) - f(s)$.

4.3 Removal of Multiple Maxima

In principle, the strategy described in Sec. 4.2 can be used iteratively to remove multiple maxima one by one. However, regions corresponding to discarded maxima can form localized clusters, i.e., *hill chains*. Since hills in these chains are nested, flattening each hill one after the other with this strategy results in multiple passes over the same portions of the data. To address this issue, it is necessary to slightly modify the first three steps of the localized simplification process: Step 1 needs to compute the combined region of every hill chain, Step 2 needs to locally simplify each of these combined regions, and Step 3 needs to integrate all local orders into the global order.

To this end, Step 1 has to initiate from each undesired maximum $m_i \in \mathcal{C}^d_{\hat{f}} \setminus \overline{\mathcal{C}^d_{\hat{f}}}$ a superlevel set propagation $\mathcal{M}_{s_i}^{m_i}$ (Sec. 4.2). When a propagation reaches a saddle s_i, then the propagation stops if at least one higher saddle neighbor has not been visited so far by any propagation (Fig. 5b: light propagations), or otherwise—i.e., if now all higher saddle neighbors have been visited by some propagation—the current propagation merges with the other propagations that reached the same saddle and then continues towards the next saddle (Fig. 5b: dark propagations). In the latter case, this means at an algorithmic level that the priority queues used for the propagations which stopped at s_i need to be merged with that of the current propagation to guarantee the valid extraction of the superlevel set component. As suggested by Gueunet et al. [29] in the context of computing augmented contour trees, LTS uses Fibonacci heaps [16, 24] to model the propagation priority queues since they support constant time merge operations; guaranteeing an overall linearithmic time complexity. At the end of Step 1, LTS extracts one superlevel set component per hill chain (Fig. 5b), which can be flattened at once in Step 2. The list of vertices of each hill chain can be efficiently composed by maintaining a Union-Find data structure [16].

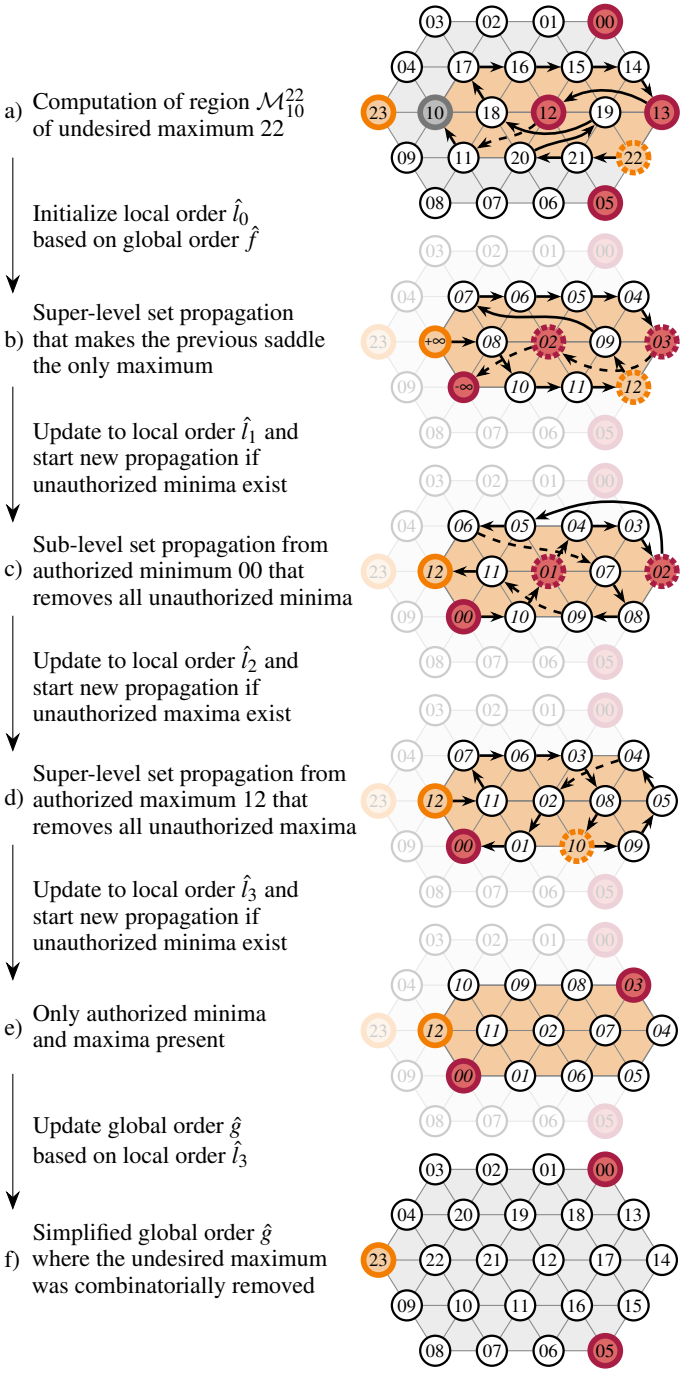


Fig. 6. Illustration of the localized simplification process, where authorized (solid) and unauthorized (dotted) maxima, minima, and saddles are shown with orange, red, and gray nodes, respectively. Here, the algorithm is tasked to update the global order $\hat{f}$ by removing the maximum m with index $\hat{f}(m) = 22$ (a). The core concept of LTS is that removing the maximum m corresponds to reordering the vertices of its superlevel set component $\mathcal{M}_s^m$ (orange triangles, the order is denoted by the arrows between the vertices) in a new local order $\hat{l} : \mathcal{M}_s^m \to \mathbb{N}$ such that s is the only maximum in $\mathcal{M}_s^m$, and all minima in $\mathcal{M}_s^m$ have a neighbor outside $\mathcal{M}_s^m$. This can be enforced by iteratively removing unauthorized extrema locally, via alternating superlevel and sublevel set propagations (b-d) until only authorized extrema remain in $\mathcal{M}_s^m$ (e). Finally, the new local order $\hat{l}$ is used to derive a new global order $\hat{g}$ such that $\mathcal{M}_s^m$ no longer contains any extrema (f). Note, removing a maximum m also removes minima inside its corresponding superlevel set component $\mathcal{M}_s^m$ (e.g., vertices 12 and 13 (a)); a known artifact of flattening-based simplification. If desired, their existence can be enforced in an optional post-process that lowers their order until they are identified as minima again.

Next, Step 2 independently computes a local order for every hill chain as described in Sec. 4.2, and then Step 3 iteratively integrates every computed local order into the global vertex order as described before. Finally, the global procedure that computes an injective numerical perturbation remains identical (Sec. 4.2).

4.4 Parallel Removal of Multiple Maxima

This subsection describes a shared-memory parallelization of LTS. Step 1 can be trivially parallelized on a per discarded maximum basis. However, propagations need to be synchronized at saddles, to discover which propagation is the last one to reach a saddle s. This can be implemented with an atomic counter that records the number of remaining unvisited higher neighbors, which only reaches zero for the last propagation visiting s. Step 2 can be trivially parallelized on a per region basis. Step 3 can be parallelized by computing an order $\hat{i}: \mathcal{M} \to \mathbb{N} \times \mathbb{N}$ operating along the initial order $\hat{f}$ (disambiguating distinct regions) and the local orders $\hat{l}$:

$$\hat{i}(v) = \begin{cases} (\,\hat{f}(s)\,, +\infty\,) & \text{if } v = s \text{ for some } \mathcal{M}_s^m, \\ (\,\hat{f}(s)\,, \hat{l}(v)\,) & \text{if } v \neq s \wedge v \in \mathcal{M}_s^m \text{ for some } \mathcal{M}_s^m, \text{ and} \\ (\,\hat{f}(v)\,, \;\; 0\;\;) & \text{otherwise.} \end{cases} \quad (6)$$

Sorting all vertices based on $\hat{i}$ with a parallel sorting algorithm—such as GNU parallel sort [51]—yields the new global order $\hat{g}$. Finally, Step 4 cannot be parallelized in its current form as it requires a sequential pass over all vertices. However, this step induces negligible computation times in practice, which does not impair parallel performances.

4.5 Persistence-Driven Simplification

In the special case where all maxima below a given persistence threshold ε need to be removed, an adapted version of LTS first detects all non-persistent maxima during the superlevel set propagations of Step 1, and then locally simplifies their corresponding regions as described before. Specifically, Step 1 now has to initiate, from *each* maximum m, a superlevel set propagation that now also tracks the scalar difference between m and its last visited vertex. If this difference exceeds ε, then the current propagation immediately terminates, since it must correspond to a *persistent maximum* (Fig. 5c, dark orange and dark red). All other propagations—i.e., propagations that merged with persistent propagations (Fig. 5c, light red) or propagations that terminated at a saddle with unvisited larger neighbors (Fig. 5c, light orange and light gray)—must correspond to *non-persistent maxima*. As illustrated in Fig. 5c, this procedure partially computes the persistence diagram by only constructing critical point pairs with a persistence smaller than ε.

This process can be parallelized as described in Sec. 4.4, and the remaining steps are identical to previous descriptions. In particular, only the regions corresponding to non-persistent maxima are processed by the local flattening procedure. At the end of this process, since each region is flattened by a height equal to the function difference between its highest maximum and its lowest saddle (Sec. 4.2), the output function g guarantees that $||f-g||_\infty \leq \varepsilon$ and that all maxima less persistent than ε have indeed been removed.

5 Results

This section presents experimental results of a C++ implementation of LTS—in the form of a Topology ToolKit (TTK) [55] module—obtained on a desktop computer with a Xeon CPU (2.6 GHz, 2x6 cores) and with 64 GB of RAM. The presented datasets have been downloaded from public repositories [39, 58], and the used TDA pipelines consist of modules readily available in TTK.

5.1 Time Performance

As discussed in Sec. 4.2, LTS extends the baseline approach by Tierny and Pascucci [56] to the local simplification of sub- and superlevel set components. Thus, LTS admits the same asymptotic time complexity: $\mathcal{O}\big(N_I \times |\mathcal{V}| log(|\mathcal{V}|)\big)$, where $|\mathcal{V}|$ is the number of vertices in $\mathcal{M}$, and N_I represents the number of iterations of the algorithm (Sec. 4.2) with $N_I = |\mathcal{V}|$ in the worst case [56]. However, in comparison to the baseline approach, LTS improves run times in three ways. First, since LTS is localized, computational intensive procedures are constraint onto (often small) subsets of the domain. In the presented experiments, only up to 2% of the domain is processed for realistic levels of simplification (white lines in Tab. 1), and up to 70% for the most aggressive levels (gray lines). Second, LTS relaxes the constraints on the extrema located on the boundary of sub- and superlevel set components (Sec. 4.2), which has the positive effect that the iterative process converges faster to valid local orders. For a given dataset, the maximum number of iterations, over all components to simplify, was rarely above 1 for realistic simplification levels. For the most aggressive simplification levels (gray lines in Tab. 1), only a few components require at most 6 iterations, while the vast majority of components require only 1 iteration; resulting in an average number of 1 iteration for all datasets across all simplification levels. The final speedup results from the parallel processing of individual regions. These three effects combined—i.e., the parallel simplification of local regions with faster convergence—significantly improve run time performances. In particular, they make the main parts of LTS output-sensitive where run time is now a function over the number of extrema to remove.

Tab. 1 provides a detailed run time comparison between the baseline approach [56] and LTS for various datasets. All timings are based on the implementations available in TTK [55]. First, for realistic extremum selections (with a persistence threshold of 1% of the function range), LTS provides an average speedup in sequential of ×4 over the baseline approach. As discussed above, this speedup can be explained by the localized nature of LTS and its faster convergence. The most important performance gains can be observed when running LTS in parallel, with

Dataset	$\lvert\mathcal{M}\rvert$	BL	LTS (1 core)		LTS (12 cores)		
		Time	Time	**S.U.**	Time	**S.U.**	*P.E.*
Silicium	0.1×10^6	0.100	0.034	**2.9**	0.004	**25.0**	*71 %*
Cells	0.8×10^6	1.410	0.303	**4.7**	0.031	**45.5**	*81 %*
OceanVortices	1×10^6	1.321	0.437	**3.0**	0.049	**27.0**	*74 %*
Foot	17×10^6	20.505	6.673	**3.1**	0.785	**26.1**	*71 %*
Random	17×10^6	65.201	8.178	**8.0**	0.927	**70.3**	*74 %*
Turbulence	17×10^6	53.444	12.015	**4.4**	1.255	**42.6**	*80 %*
Backpack	98×10^6	174.599	67.572	**2.6**	7.454	**23.4**	*76 %*
Jet	134×10^6	167.879	52.121	**3.2**	6.366	**26.4**	*68 %*
Silicium	0.1×10^6	0.111	0.051	**2.2**	0.011	**10.1**	*39 %*
Cells	0.8×10^6	1.767	1.339	**1.3**	0.410	**4.3**	*27 %*
OceanVortices	1×10^6	1.118	1.562	**0.7**	1.193	**0.9**	*11 %*
Foot	17×10^6	19.498	11.171	**1.7**	2.037	**9.6**	*46 %*
Random	17×10^6	51.576	28.363	**1.8**	6.684	**7.7**	*35 %*
Turbulence	17×10^6	49.075	18.435	**2.7**	4.232	**11.6**	*36 %*
Backpack	98×10^6	169.415	94.319	**1.8**	28.655	**5.9**	*27 %*
Jet	134×10^6	165.160	57.871	**2.9**	7.748	**21.3**	*62 %*

Table 1. Performance comparison of the baseline approach [55, 56] (BL) versus the sequential and parallel execution of LTS for realistic extremum selections (white lines: 1% of the function range) and a stress case (gray lines: only global extrema are preserved). Timings are in seconds, speedup (*S.U.*) is relative to the baseline approach, and parallel efficiency (*P.E.*) is defined as parallel speedup divided by the number of cores.

Dataset	$\lvert\mathcal{M}\rvert$	Diagram	Dia. + BL	Dia. + LTS		Pers.-LTS	
		Time	Time	Time	**S.U.**	Time	**S.U.**
Silicium	0.1×10^6	*0.055*	0.156	0.060	**2.6**	0.010	**15.6**
Cells	0.8×10^6	*0.113*	1.505	0.145	**10.4**	0.049	**30.7**
OceanVortices	1×10^6	*0.208*	1.530	0.258	**5.9**	0.132	**11.6**
Foot	17×10^6	*2.355*	23.391	3.207	**7.3**	1.080	**21.7**
Random	17×10^6	*14.808*	81.039	16.440	**4.9**	2.853	**28.4**
Turbulence	17×10^6	*2.703*	67.232	4.024	**16.7**	2.313	**29.1**
Backpack	98×10^6	*24.397*	253.386	32.061	**7.9**	9.364	**27.1**
Jet	134×10^6	*149.327*	316.305	155.827	**2.0**	7.558	**41.9**
Silicium	0.1×10^6	*0.055*	0.155	0.066	**2.3**	0.016	**9.7**
Cells	0.8×10^6	*0.111*	1.874	0.520	**3.6**	0.456	**4.1**
OceanVortices	1×10^6	*0.219*	1.337	1.472	**0.9**	1.224	**1.1**
Foot	17×10^6	*2.363*	21.576	4.433	**4.9**	2.596	**8.3**
Random	17×10^6	*14.891*	65.486	21.903	**3.0**	9.849	**6.6**
Turbulence	17×10^6	*2.725*	51.907	6.853	**7.6**	6.032	**8.6**
Backpack	98×10^6	*24.018*	228.711	52.906	**4.3**	19.511	**11.7**
Jet	134×10^6	*148.811*	322.559	156.641	**2.1**	11.083	**29.1**

Table 2. Performance comparison of persistence-driven simplification (white lines: 1% of the function range, gray lines: only global extrema are preserved) by computing the persistence diagram [29, 55] (Dia, 12 cores) followed either by the baseline (BL) approach [55, 56] or LTS (12 cores), or by alternatively computing the persistence-driven specialization of LTS (Pers.-LTS, 12 cores). Speedup (*S.U.*) is relative to *"Dia + BL"*.

an overall average speedup of $\times 36$ over the baseline approach (white lines, Tab. 1). The average parallel efficiency for realistic simplifications (white lines, Tab. 1) is slightly over 74%, which illustrates the good scaling of LTS in this setup. Second, when stressing LTS with aggressive simplification levels (by only keeping the global minimum and the global maximum, gray lines), performances decrease, further illustrating the output sensitive aspect of LTS. The average speedup in parallel over the baseline drops down to $\times 9$, while the parallel efficiency of LTS drops down to 35%. This can be explained by a work load imbalance between the threads, since for aggressive thresholds, some sub- and superlevel set components can become significantly larger than others, and require more iterations than others.

Table 2 provides a detailed run time comparison about persistence-driven simplification via the computation of the persistence diagram [29] followed either by the baseline approach [56] or LTS (12 cores), or alternatively via the persistence-driven specialization of LTS (12 cores, Sec. 4.5). When used in conjunction with a standard persistence diagram computation (*"Dia. + LTS"*), LTS provides an average speedup of $\times 7$ overall for realistic levels (1% of the function range, white lines). Since LTS only partially computes the persistence diagram, the persistence-driven specialization of LTS (right column) achieves significant performance gains, with an average speedup of $\times 26$ over the baseline approach. Again, for more aggressive simplification levels (gray lines), performances start to deteriorate as regions which may not need to be later simplified still need to be visited by the algorithm in order to compute the partial diagram. Despite this, LTS still provides an order of magnitude speedup on average.

The output-sensitive behavior of LTS is further illustrated in Fig. 7, which plots the simplification time of the random dataset as a function over the percentage of randomly selected extrema to remove. The run time of the baseline approach [56] (gray curve) continuously decreases from 60 to 50 seconds, which indicates that the increasing size of the simplified regions is beneficial to this global approach. Conversely, the run time of the sequential (orange curve) and parallel (red curve) execution of LTS is progressively increasing, which indicates that small regions are beneficial to the local approach.

5.2 Application to Interactive Exploration

The significant speedup of LTS enables the interactive exploration of simplification parameters and their effect on TDA. Figures 8–11 present different use cases of interactive exploration scenarios, where features of acquired and simulated datasets are characterized based on level sets, merge trees, and Morse-Smale complexes. The context of the analysis and the details of the topological pipeline used after simplification is reported in the caption of each figure.

Fig. 8 illustrates the interest of persistence-driven simplification, which can be employed in batch mode to remove, in a pre-processing, features below a conservative persistence level. While topological simplification is nearly always needed to cope with noise, the relevant amount of simplification is often not known a priori and therefore needs to be adjusted interactively on a per dataset basis. Moreover, there may be more than one relevant simplification level, and visualizing the entire hierarchy is often of interest for analysts. This motivates efficient algorithms capable of supporting interactive exploration sessions. In the use cases presented in Figs. 10–11, the persistence diagram and its persistence curve are computed in a pre-process. Next, the user interactively explores different potentially interesting simplification levels, typically reported by the persistence curve. After each modification of the simplification level, the TDA pipeline under consideration is recomputed on the simplified data. For all of these scenarios, pre-simplifying the data with the baseline approach [56] took longer than running the rest of the TDA pipeline. For the larger datasets, this pre-simplification is the main bottleneck, representing up to 90% of the computation. In contrast, LTS provides for the simplification step alone speedups of an order of magnitude, making it possible to update the analysis within seconds while the state-of-the-art needs several minutes.

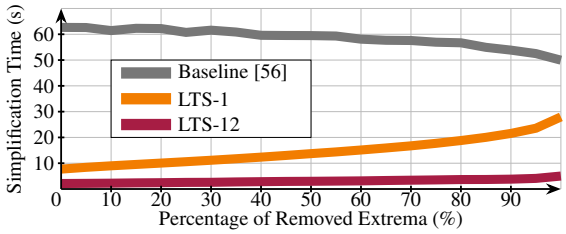


Fig. 7. Computation time as a function of the percentage of extrema to simplify, with a random selection of extrema on the random dataset (gray: baseline [56], orange: sequential LTS, and red: LTS with 12 cores).

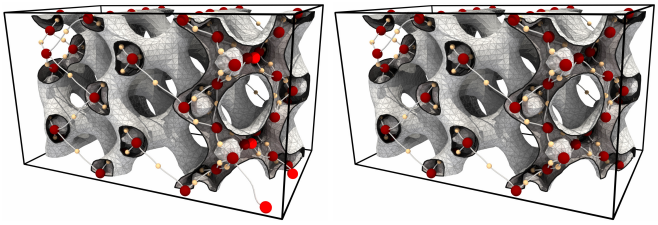


Fig. 8. Persistence-driven simplification of maxima not corresponding to silicium atoms [39] (left, bright red spheres), prior to the extraction of the correct lattice structure via the Morse-Smale complex (dark red: maxima, orange: 2-saddles, white curves: separatrices). Computing the persistence diagram [29] and pre-simplifying the data with the baseline approach [56] takes $0.15s$, while LTS requires $0.02s$.

Fig. 9. Extractions of bones in a CT scan of a human foot [39] (volume rendering, left), where bones are identified as the regions that correspond to the leaf-arcs of the merge tree. For a persistence simplification threshold of 180, the bones of the 5 toes are precisely extracted (center), and are further subdivided along the joints at smaller thresholds (150, right). Pre-simplifying the data with the baseline approach [56] takes $19.6s$, and computing the merge tree segmentation [29] requires $1.8s$. LTS requires $1.3s$ to pre-simplify the data, resulting in a *pipeline-speedup* of $\times 7$.

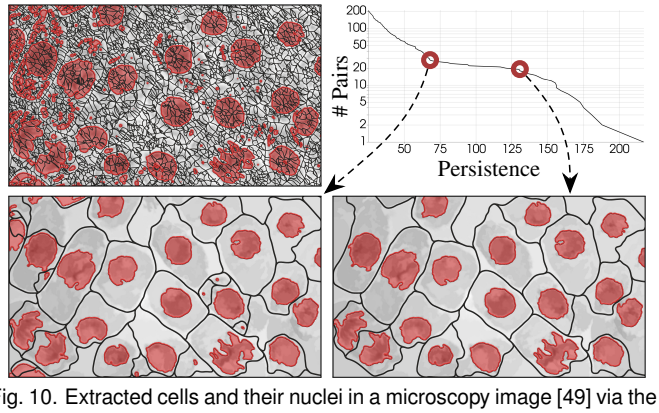


Fig. 10. Extracted cells and their nuclei in a microscopy image [49] via the Morse-Smale complex separatrices (black) and sublevel sets (red). Without simplification (top left), the analysis suffers from over-segmentation with numerous false positives. The persistence curve (top right) exhibits a clear plateau, indicating a stable simplification range separating noise from features, where the appropriate persistence threshold still needs to be adjusted interactively. False positives are still identified for the left extremity of the plateau (bottom left), while a correct extraction is obtained at the right extremity (bottom right). Pre-simplifying the data with the baseline approach [56] takes $1.58s$, whereas LTS only requires $0.18s$. Computing the Morse-Smale complex and the sublevel sets takes $0.94s$.

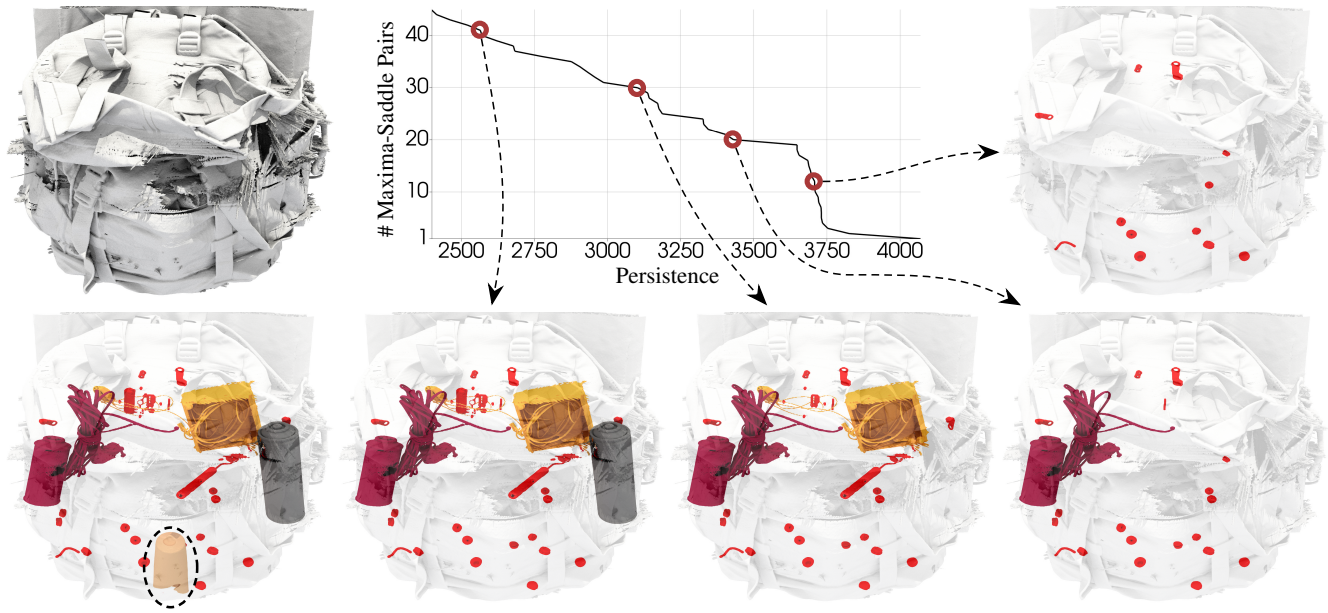


Fig. 11. Level set based feature extraction in the CT-scan of a backpack (top left). The initial level set counts hundreds of connected components (640). The persistence curve exhibits several kinks, which challenges the identification of a clear simplification threshold, hence motivating interactive exploration. For aggressive levels, only the denser objects (metal elements) are maintained in the level set (red objects, top right). For less aggressive values (bottom, from right to left), less dense objects (bottles, cords, tools, boxes, etc.) progressively appear in the segmentation, in decreasing order of the persistence of the corresponding topological feature. For the leftmost image, the front bottle (dashed black line) corresponds to a low-persistence feature whose maximum has been selected geometrically. For each simplification, LTS computes within seconds ($13.567s$, leftmost image) while state-of-the-art techniques [56] take minutes to simplify ($179.175s$), resulting in an overall *pipeline-speedup* of $\times 12$.

5.3 Limitations

In contrast to the baseline approach by Tierny and Pascucci [56], LTS uses a black-list strategy and processes only the regions which need simplification. To keep track of these regions, heavier data structures need to be maintained through the propagations (Fibonacci heaps/Union-Find), but as demonstrated in Sec. 5.1, LTS still provides superior performances, even in sequential. The parallel performance of LTS depends significantly on the work load balance among the threads. Specifically, the number of tasks—i.e., propagations and local simplifications—decreases over time, and at some point the algorithm processes less tasks than available cores. Thus, parallel efficiency starts to degrade when this time interval takes up larger fractions of the total computation time. However, for the realistic simplification scenarios presented in Sec. 5.1, the work load among threads is well balanced.

Since LTS is based on flattening (and as such modifies data values), it introduces visual flat-plateau artifacts (Fig. 3). This can be problematic if one wants to extract geometrical features within the simplified regions, such as the intra-cellular features of Fig. 10. Such geometries should be extracted based on the original data, where the simplified data can be used as a mask for segmentation. Yet, these plateaus are needed to guarantee the removal of undesired topological features in level-set based segmentations (Fig. 2), and LTS can be combined with numerical techniques to provide smoother results if needed. Outside of the simplified areas, the integral lines are left unchanged, since simplification is only applied locally. Thus, the separatrices of the Morse-Smale complex outside of simplified regions are also not impacted (in Fig. 10, the separatrices of the simplified complexes, bottom, are subsets of separatrices of the unsimplified one, top left). Within the simplified regions, however, the geometry of integral lines can change. Although the Morse-Smale complex will indeed only detect the authorized critical points [55], the simplification may consequently have an impact on how separatrices connect them, if they traverse simplified regions, which requires more detailed investigation in future work.

LTS focuses on extremum-saddle pairs and does not support saddle-saddle pair removal. However, from our experience, the extrema of a scalar field are the topological objects that users investigate in priority for feature extraction. Moreover, if extrema are not selected based on persistence, the value of the remaining critical points may change after simplification. This happens for instance if the only minimum to preserve is initially located higher than the only maximum to preserve, in which case the algorithm will change their values to satisfy the input constraints. Finally, in our experiments we observed that computing abstractions on the pre-simplified data seems to be equivalent to post-simplifying the topological abstractions themselves—e.g., the merge trees and Morse-Smale complexes of the simplified fields are identical to the pruned abstractions of the original field—but this observation needs to be further evaluated in future work.

6 Conclusion

This paper described a combinatorial approach for the localized topological simplification (LTS) of scalar data. Given a PL scalar field f and a selection of extrema to preserve, LTS transforms f to a new PL scalar field g via localized iterative flattening, such that g is close to f and only exhibits the selected set of extrema. LTS significantly accelerates an essential part of topological data analysis by reducing the time spent on data pre-simplification by up to an order of magnitude in our experiments ($\times 36$). In many instances, this brings the execution time of TDA pipelines from minutes down to a few seconds, which enables the interactive exploration of simplification parameters. Although LTS scales well in the presented experiments, the tested implementation is optimized for workstations. Next, the code needs to be optimized for larger, shared-memory, high-performance machines. Extensions to GPU computation will also be considered. For larger problems, one can investigate distributed parallelism, for which the localized nature of LTS is also expected to be beneficial. Another line of research is the removal of saddle-saddle pairs, which has been reported to be NP-hard in general [3], and thus heuristic approaches need to be investigated.

Acknowledgments

This work was supported by the U.S. Department of Homeland Security under Grant Award 2017-ST-061-QA0001 and 17STQAC00001-03-03, and the National Science Foundation Program under Award No. 1350573. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security. This work was also partially supported by the European Commission grant ERC-2019-COG "*TORI*" (ref. 863464), and the German research foundation (DFG) through the IRTG 2057. Julien Tierny would like to dedicate this paper to his son Marvin.

REFERENCES

[1] P. K. Agarwal, L. Arge, and K. Yi. I/O-Efficient Batched Union-Find and its Applications to Terrain Analysis. In N. Amenta and O. Cheong, eds., *Symp. on Comp. Geom.*, 2006.

[2] 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*, 2018.

[3] D. Attali, U. Bauer, O. Devillers, M. Glisse, and A. Lieutier. Homological reconstruction and simplification in R^3. In *Symp. on Comp. Geom.*, 2013.

[4] D. Attali, M. Glisse, S. Hornus, F. Lazarus, and D. Morozov. Persistence-Sensitive Simplification of Functions on Surfaces in Linear Time. 2009.

[5] T. F. Banchoff. Critical Points and Curvature for Embedded Polyhedral Surfaces. *The American Mathematical Monthly*, 77(5):475–485, 1970.

[6] U. Bauer, C. Lange, and M. Wardetzky. Optimal Topological Simplification of Discrete Functions on Surfaces. *Disc. Compu. Geom.*, 2012.

[7] 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*, 2018.

[8] S. Biasotti, D. Giorgio, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 2008.

[9] A. Bock, H. Doraiswamy, A. Summers, and C. T. Silva. TopoAngler: Interactive Topology-Based Extraction of Fishes. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2018.

[10] P.-T. Bremer, A. Gruber, J. Bennett, A. Gyulassy, H. Kolla, J. Chen, , and R. Grout. Identifying turbulent structures through topological segmentation. *Communications in Applied Mathematics and Computational Science*, 11:37–53, 2016.

[11] P.-T. Bremer, B. Hamann, H. Edelsbrunner, and V. Pascucci. A Topological Hierarchy for Functions on Triangulated Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 2004.

[12] P.-T. 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 Transactions on Visualization and Computer Graphics*, 2011.

[13] H. Carr. *Topological Manipulation of Isosurfaces*. Ph. D. Thesis, University of British Columbia, 2004.

[14] H. Carr, J. Snoeyink, and U. Axen. Computing Contour Trees in All Dimensions. *Computational Geometry*, 24(2):75 – 94, 2003.

[15] H. A. Carr, J. Snoeyink, and M. van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *IEEE VIS*, 2004.

[16] T. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2009.

[17] L. Dagum and R. Menon. OpenMP: an industry standard API for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

[18] L. De Floriani, U. Fugacci, F. Iuricich, and P. Magillo. Morse complexes for shape segmentation and homological analysis: discrete models and algorithms. *Comp. Grap. For.*, 2015.

[19] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Soc., 2010.

[20] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological Persistence and Simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[21] H. Edelsbrunner, D. Morozov, and V. Pascucci. Persistence-Sensitive Simplification Functions on 2-Manifolds. In *Symp. on Comp. Geom.*, 2006.

[22] H. Edelsbrunner and E. P. Mücke. Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms. *ACM Transactions on Graphics (tog)*, 9(1):66–104, 1990.

[23] R. Forman. A User's Guide to Discrete Morse Theory. *Sém. Lothar. Combin*, 48:35pp, 2002.

[24] M. L. Fredman and R. E. Tarjan. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

[25] C. Garth. Simulation of a Jet Flow. *IEEE Dataport*, 2020. `http://dx.doi.org/10.21227/qjxp-kc31`.

[26] Y. I. Gingold and D. Zorin. Controlled-topology filtering. In R. R. Martin and S. Hu, eds., *Proc. of ACM SPM*, 2006.

[27] D. Guenther, R. Alvarez-Boto, J. Contreras-Garcia, J.-P. Piquemal, and J. Tierny. Characterizing Molecular Interactions in Chemical Systems. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2014.

[28] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-Based Augmented Merge Trees with Fibonacci Heaps. In *IEEE Symposium on Large Data Analysis and Visualization 2017*, 2017.

[29] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-Based Augmented Contour Trees with Fibonacci Heaps. *IEEE Trans. Parallel Distrib. Syst.*, 2019.

[30] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based Augmented Reeb Graphs with Dynamic ST-Trees. In *Eurographics Symposium on Parallel Graphics and Visualization*, 2019.

[31] D. Günther, A. Jacobson, J. Reininghaus, H. Seidel, O. Sorkine-Hornung, and T. Weinkauf. Fast and Memory-Efficient Topological Denoising of 2D and 3D Scalar Fields. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2014.

[32] A. Gyulassy, P. Bremer, R. Grout, H. Kolla, J. Chen, and V. Pascucci. Stability of Dissipation Elements: A case study in combustion. *Computer Graphics Forum*, 2014.

[33] A. Gyulassy, P. Bremer, and V. Pascucci. Shared-Memory Parallel Computation of Morse-Smale Complexes with Improved Accuracy. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2018.

[34] A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci. A Practical Approach to Morse-Smale Complex Computation: Scalability and Generality. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2008.

[35] A. Gyulassy, M. A. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically Clean Distance Fields. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2007.

[36] 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 Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2015.

[37] 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. In *Computer Graphics Forum*, vol. 35, pp. 643–667. Wiley Online Library, 2016.

[38] J. Kasten, J. Reininghaus, I. Hotz, and H. Hege. Two-Dimensional Time-Dependent Vortex Regions based on the Acceleration Magnitude. *IEEE Transactions on Visualization and Computer Graphics*, 2011.

[39] P. Klacansky. Open Scientific Visualization Datasets. `https://klacansky.com/open-scivis-datasets/`, 2020.

[40] D. Laney, A. Mascarenhas, P. Miller, V. Pascucci, et al. Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2006.

[41] J. Lukasczyk, G. Aldrich, M. Steptoe, G. Favelier, C. Gueunet, J. Tierny, R. Maciejewski, B. Hamann, and H. Leitte. Viscous Fingering: A Topological Visual Analytic Approach. *Applied Mechanics and Materials*, 2017.

[42] J. W. Milnor, M. Spivak, R. Wells, and R. Wells. *Morse Theory*. Princeton university press, 1963.

[43] X. Ni, M. Garland, and J. C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Trans. on Graph.*, 2004.

[44] M. Olejniczak, A. S. P. Gomes, and J. Tierny. A Topological Data Analysis Perspective on Non-Covalent Interactions in Relativistic Calculations. *International Journal of Quantum Chemistry*, 2019.

[45] V. Pascucci, G. Scorzelli, P. T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Trans. on Graph.*, 2007.

[46] G. Patanè and B. Falcidieno. Computing smooth approximations of scalar functions with constraints. *Comput. Graph.*, 2009.

[47] G. Reeb. Sur les points singuliers dune forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus des séances de l'Académie des sciences*, 222(847-849):76, 1946.

[48] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2011.

[49] S. Shanti. Microscope Image of Allium Cepa. *Cell Image Library (CIL)*, 2013. `https://doi.org/doi:10.7295/W9CIL43552`.

[50] 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 Transactions on Visualization and Computer Graphics*, 2016.

[51] J. Singler and B. Konsik. The GNU libstdc++ Parallel Mode: Software Engineering Considerations. In *Proceedings of the 1st international workshop on Multicore software engineering*, pp. 15–22, 2008.

[52] P. Soille. Optimal Removal of Spurious Pits in Digital Elevation Models. *Water Resources Research*, 2004.

[53] 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 Symposium on Large Data Analysis and Visualization*, 2019.

[54] T. Sousbie. The Persistent Cosmic Web and its Filamentary Structure: Theory and Implementations. *Royal Astronomical Society*, 2011.

[55] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2017. `https://topology-tool-kit.github.io/`.

[56] J. Tierny and V. Pascucci. Generalized Topological Simplification of Scalar Fields on Surfaces. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2012.

[57] X. Tricoche, G. Scheuermann, and H. Hagen. Continuous Topology Simplification of Planar Vector Fields. In *Proceedings Visualization, 2001. VIS'01.*, pp. 159–166. IEEE, 2001.

[58] TTK Contributers. *TTK Data Repository*, 2020. `https://github.com/topology-tool-kit/ttk-data/tree/dev`.

[59] T. Weinkauf, Y. I. Gingold, and O. Sorkine. Topology-based Smoothing of 2D Scalar Fields with C^1-Continuity. *Computer Graphics Forum*, 2010.

[60] E. Zhang, K. Mischaikow, and G. Turk. Vector Field Design on Surfaces. *ACM Transactions on Graphics (ToG)*, 25(4):1294–1326, 2006.