



HAL
open science

Fast Approximation of Persistence Diagrams with Guarantees

Jules Vidal, Julien Tierny

► **To cite this version:**

Jules Vidal, Julien Tierny. Fast Approximation of Persistence Diagrams with Guarantees. IEEE Symposium on Large Data Analysis and Visualization, Oct 2021, New Orleans, United States. hal-03331235

HAL Id: hal-03331235

<https://hal.science/hal-03331235>

Submitted on 1 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fast Approximation of Persistence Diagrams with Guarantees

Jules Vidal and Julien Tierny *
CNRS – Sorbonne Université

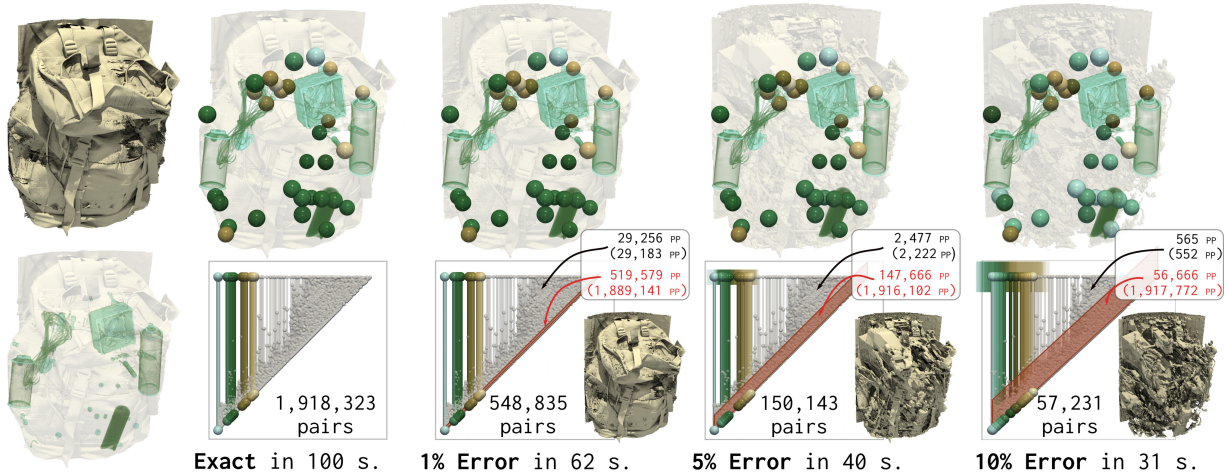


Figure 1: Approximations of persistence diagrams for a CT scan of a backpack, with different Bottleneck approximation errors. High persistence features correspond to high density objects present in the bag (leftmost, bottom). In this example, our controlled approximation reduces computation time by 70% for an error tolerance of 10%. Our algorithm also provides an approximation of the scalar field that is precise around persistent features (top, volume rendering of the approximated fields) and deteriorated elsewhere (bottom, isocontours capturing the cloth of the bag). For each approximation, the thirty most persistent maxima are represented by spheres (top views) which correctly capture the features of the data. The uncertainty resulting from our approximation is visualized in the diagram with (i) colored squares, which bound the correct location of *certain* features (which are guaranteed to be present in the exact result). These are located outside a (ii) red band (in the vicinity of the diagonal), which denotes features which *might* not exist in the exact diagram. Our approximations precisely capture the high persistence features of the data (out of the red band, black numbers) and collect significantly less noisy features (red numbers, red band) than the exact result (numbers in parentheses).

ABSTRACT

This paper presents an algorithm for the efficient approximation of the saddle-extremum persistence diagram of a scalar field. Vidal et al. introduced recently a fast algorithm for such an approximation (by interrupting a progressive computation framework [78]). However, no theoretical guarantee was provided regarding its approximation quality. In this work, we revisit the progressive framework of Vidal et al. [78] and we introduce in contrast a novel approximation algorithm, with a user controlled approximation error, specifically, on the Bottleneck distance to the exact solution. Our approach is based on a hierarchical representation of the input data, and relies on local simplifications of the scalar field to accelerate the computation, while maintaining a controlled bound on the output error. The locality of our approach enables further speedups thanks to shared memory parallelism. Experiments conducted on real life datasets show that for a mild error tolerance (5% relative Bottleneck distance), our approach improves runtime performance by 18% on average (and up to 48% on large, noisy datasets) in comparison to standard, exact, publicly available implementations. In addition to the strong guarantees on its approximation error, we show that our algorithm also provides in practice outputs which are on average 5 times more accurate (in terms of the L_2 -Wasserstein distance) than a naive approximation baseline. We illustrate the utility of our approach for interactive data exploration and we document visualization strategies for conveying the uncertainty related to our approximations.

*E-mails: {jules.vidal, julien.tierny}@sorbonne-universite.fr

1 INTRODUCTION

Modern datasets, acquired or simulated, are continuously gaining in geometrical complexity, thanks to the ever-increasing accuracy of acquisition devices or computing power of high performance systems. This geometrical complexity makes interactive exploration and analysis difficult, which challenges the interpretation of large datasets by end users. This motivates the definition of expressive data abstractions, capable of capturing the main features present in large datasets into concise representations, which visually convey the most important information to the users.

In that context, Topological Data Analysis (TDA) [21] forms a family of generic, robust, and efficient techniques whose utility has been demonstrated in a number of visualization tasks [42] for revealing the implicit structural patterns present in complex datasets. Examples of popular application fields include turbulent combustion [10, 36, 49], material sciences [26, 39, 40], nuclear energy [53], fluid dynamics [47], bioimaging [7, 13], quantum chemistry [3, 30, 56] or astrophysics [68, 71]. These applications rely on established topological data abstractions, such as contour trees [8, 11, 34, 70], Reeb graphs [5, 61, 64] or Morse-Smale complexes [19, 37, 38, 65]. In particular, the Persistence diagram [24] is a concise data representation, which visually summarizes the population of features of interest present in large datasets, as a function of a measure of importance called *topological persistence*. Its conciseness made it increasingly popular in data visualization and analysis, where it quickly provides visual hints regarding the number and salience of the structural features present in large datasets.

While topological methods usually have an acceptable time complexity, the construction of the above data representations can still be time consuming in practice for large, real-life datasets. Thus, when they are integrated into larger analysis pipelines, TDA algorithms

can become a serious time bottleneck. Several research directions are possible to improve the time performance of TDA algorithms. A natural avenue for performance improvement consists in revisiting these algorithms for a parallel computation [14, 32, 34, 35, 37, 52, 67], which is particularly relevant for high-performance hardware. Another direction consists in considering *degraded* computations, which trade accuracy for speed. This direction is particularly appealing for persistence diagrams, as they can contain, for noisy datasets, many features with low persistence which would gain in being only *approximated*, as they often have in practice only little relevance (important features typically have a high persistence, see Fig. 1).

In this work, we introduce an algorithm for the fast approximation of extremum persistence diagrams of large scalar datasets, with a user-controlled error bound. In particular, we re-visit the progressive computation framework of Vidal et al. [78] and derive a multiresolution strategy for accelerating the computation, based on a controlled degradation of the input data (hence resulting in a controlled approximation of the result). Extensive experiments demonstrate the time performance gain provided by our algorithm (up to 48% on large, noisy datasets) for mild error tolerance (e.g. 5%), as well as its superior accuracy in comparison to a naive approximation baseline. We also document several visualization strategies for our approximated topological features, to easily enable the visual identification of *certain* features (which will be present in the exact diagram) along with visual clues of their positional uncertainty in the diagram.

1.1 Related Work

The literature related to our work can be classified into two main groups: (i) topological methods, and (ii) multiresolution methods.

(i) **Topological methods** have gained a growing interest from the visualization community for the last two decades [42]. Specifically, our work is closely related to the construction of topological data abstractions on scalar data, that provide a generic and robust description of the features in the data. When considering discretized data as piecewise-linear (PL) scalar fields defined on PL-manifolds, many concepts from the original Morse theory [54] can be adapted to the computational setting. As such, combinatorial algorithms have been developed to efficiently compute topological abstractions in the discrete setting. A local, combinatorial characterization of the critical points of a PL scalar field was introduced by Banchoff [1]. Critical points often correspond to features of interest in the data, but can be numerous in the presence of noise. Topological persistence [21, 24] addresses this issue by introducing an importance measure on critical points. It can be visualized in the Persistence diagram (Sec. 2.3), a visual barcode that can be generally computed using matrix reduction operations [21, 24]. The Persistence diagram has been shown to be stable [15, 16] under well-established metrics such as the Bottleneck [15] and Wasserstein [76] distances. However, some applications call for more discriminative topological abstractions. Merge trees and contour trees, for instance, track the merge and split events of connected components of sub-level sets and level sets that happen at the critical points. Combinatorial algorithms were first designed to compute these trees in low dimension [8]. An efficient algorithm was later introduced by Carr et al. [11] for the computation of the contour tree in all dimensions, with optimal time-complexity. A lot of work also focused on the parallel computation of these structures [14, 33, 34, 52, 55, 59, 70]. The Reeb graph [64] is the generalization of contour trees to domains that are not simply connected, which motivates its use for shape analysis applications [5]. In order to track the connected components of level sets on such domains, original algorithms resorted to slicing strategies [4]. Later work optimized the slicing approach to restrict it to iso-contours at critical points [20, 62]. Algorithms with optimal time complexity were introduced for the 2D case [17] and in arbitrary dimension [57]. Other work focused on its parallel computation [35] or presented efficient algorithms for its computa-

tion in specific settings [61, 74]. An other widely used topological abstraction is the Morse-Smale complex [19], which encodes the relations between critical points in terms of unique integral lines of the gradient field. These integral lines segment the domain into cells in which the gradient integrates to identical extremities. Algorithms for its computation were designed for PL manifolds [22], or later [38, 65] for the Discrete Morse Theory [27]. Parallel methods have been documented [37, 67]. For the case of multivariate scalar data, recent work [12, 72] investigated efficient algorithms for the Reeb space [23] computation, a generalization of the Reeb graph.

(ii) **Multiresolution hierarchical representations** have been largely used by the visualization community to compute and store data representations at different levels of details [79, 80]. They have been notably applied to the efficient and robust extraction of isosurfaces [28, 29, 58] and isocontours [50]. Such representations have been also documented for the contour tree [60], the Reeb graph [43] and the Morse-Smale complex [9, 31, 45]. These representations allow to adaptively simplify these topological abstractions in a *fine-to-coarse* manner. As such, only the output data structures are processed hierarchically, while the input data are processed with classical algorithms on the finest level of the hierarchy.

More closely related to our work, Vidal et al. [78] recently introduced novel ideas for the *progressive* computation of critical points and saddle-extremum persistence diagrams. In their approach, the input data is processed in a *coarse-to-fine* manner with the help of a multiresolution hierarchy on the model of the classical *red* subdivision [41, 46, 51, 63, 66] of triangulated regular grids [2, 44]. The output of the approach is progressively and efficiently updated from one level of the hierarchy to the next. In practice, their algorithms are *interruptible*, which means that the processing of the hierarchy can be stopped before reaching the finest level and still deliver an exploitable result in a lower amount of computation time. Although the authors demonstrated the quality of the interrupted outputs experimentally, there are no theoretical guarantees on their approximation quality (specifically, no error bound). In contrast, we present in this work a method for the approximate computation of a persistence diagram, with strong guarantees on the approximation error.

1.2 Contributions

This paper makes the following new contributions:

1. *A fast approximation algorithm for extremum persistence diagrams, with controlled error:* We introduce a fast algorithm for the approximated computation of persistence diagrams of extremum/saddle pairs. Our approach provides strong, user-controlled guarantees on the Bottleneck distance between the approximated diagram and the exact result. By construction, our approximation method processes a smaller number of topological events than exact methods, which reduces its run time. We observe an 18% reduction of the run time on average, for a mild tolerance of 5% on the relative Bottleneck error. Moreover, our approach provides much more accurate approximations (in terms of the L_2 Wasserstein distance) than a naive approximation baseline.
2. *An approximated visualization of topological features with uncertainty assessment:* We describe several strategies for the visualization of our topological approximations. As a by-product of our algorithm, we provide an approximated scalar field, for visualization purposes, which exactly matches our approximated diagrams and in which the approximated critical points can be reliably embedded. We augment the approximated diagrams with visual glyphs assessing the uncertainty of the approximation. These glyphs enable the visual identification of *certain* features (which will be present in the exact diagram) along with visual clues on their positional uncertainty in the diagram.
3. *A reference implementation (additional material):* Finally, we provide a reference C++ implementation of our algorithms that can be used to replicate our results as well as for future benchmarks.

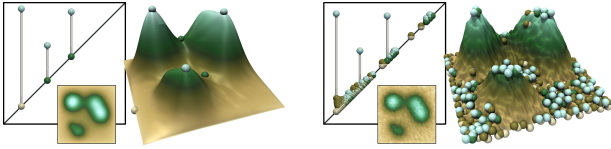


Figure 2: Persistence diagrams of a clean (left) and noisy (right) 2D scalar field. Critical points are represented by spheres (light brown: minima, cyan: maxima, others: saddles). The persistence diagram highlights the three main features of the data (high persistence pairs). On the right, the small pairs near the diagonal indicate the presence of noisy features with low persistence.

2 PRELIMINARIES

This section presents the theoretical background of our work. It contains definitions adapted from the Topology ToolKit [73]. We refer the reader to textbooks [21] for an introduction to topology.

2.1 Input Data

The input data is a piecewise linear (PL) scalar field $f : \mathcal{M} \rightarrow \mathbb{R}$ defined on a PL d -manifold \mathcal{M} , with $d \leq 3$ in our applications. The field f is characterized by its values at the vertices of \mathcal{M} , as scalar values are linearly interpolated on the simplices of \mathcal{M} of higher dimension. f is enforced to be injective on the vertices of \mathcal{M} , using a symbolic perturbation inspired by Simulation of Simplicity [25].

2.2 Critical Points

We define a *sub-level* set of f as the pre-image of $(-\infty, w)$ by f , noted $f_{-\infty}^{-1}(w) = \{p \in \mathcal{M} \mid f(p) < w\}$. When continuously increasing w , the number of topological features of $f_{-\infty}^{-1}(w)$ (its connected components, independent cycles and voids) changes only at specific vertices, called the *critical points* of f [54].

A local characterization of critical points, based on their *link*, was introduced by Banchoff [1]. For two simplices $\tau, \sigma \in \mathcal{M}$, we note $\tau < \sigma$ if τ is a *face* of σ , i.e. if τ is defined by a non-empty, strict subset of the vertices of σ . We call *star* of a vertex $v \in \mathcal{M}$ the set of its co-faces, noted $St(v) = \{\sigma \in \mathcal{M} \mid v < \sigma\}$. The *link* of v , noted $Lk(v) = \{\tau \in \mathcal{M} \mid \tau < \sigma, \sigma \in St(v), \tau \cap v = \emptyset\}$, is defined as the set of all the faces τ of the simplices of $St(v)$ that do not intersect v . Intuitively, the link of v is the *boundary* of a small simplicial neighborhood $St(v)$ of v . The *lower link* $Lk^-(v)$ is the set of simplices of $Lk(v)$ that are *lower* than v in terms of f values: $Lk^-(v) = \{\sigma \in Lk(v) \mid \forall v' \in \sigma, f(v') < f(v)\}$. Symmetrically, the *upper link* of v is given by $Lk^+(v) = \{\sigma \in Lk(v) \mid \forall v' \in \sigma, f(v') > f(v)\}$. If both $Lk^-(v)$ and $Lk^+(v)$ are non-empty and simply connected, v is a *regular point*. In any other case, v is a critical point, of *criticality* or *critical index* $\mathcal{I}(v)$: 0 for a local minimum ($Lk^-(v) = \emptyset$), d for a local maximum ($Lk^+(v) = \emptyset$), or i for a i -saddle ($0 < i < d$).

2.3 Persistence Diagrams

As an isovalue w continuously increases, the topology of $f_{-\infty}^{-1}(w)$ evolves. For a regular point v , the sub-level sets enter $St(v)$ through $Lk^-(v)$ and exit through $Lk^+(v)$ without any local change in the topology, as $Lk^-(v)$ and $Lk^+(v)$ both have exactly one connected component. Otherwise, if v is critical, a topological feature (a connected component, a cycle or a void) of $f_{-\infty}^{-1}(w)$ is either created or destroyed at the value $f(v)$. For instance, the connected components of $f_{-\infty}^{-1}(w)$ are created at the local minima of f , said to be their *birth* points. When two topological features meet at a critical point, we use the Elder rule [21] to state that the younger one (the one created last, in terms of f values) *dies* at the benefit of the oldest. In particular, the connected components of $f_{-\infty}^{-1}(w)$ die at 1-saddles. Each topological feature of $f_{-\infty}^{-1}(w)$ is thus characterized by a unique pair (c_0, c_1) of critical points, corresponding to its *birth* and *death*, with $f(c_0) < f(c_1)$ and $\mathcal{I}(c_0) = \mathcal{I}(c_1) - 1$. The *topological persistence* [24] of this pair, noted $p(c_0, c_1) = f(c_1) - f(c_0)$,

denotes the lifetime of the corresponding feature in $f_{-\infty}^{-1}(w)$ in terms of its scalar range. The persistence of the connected components in $f_{-\infty}^{-1}(w)$ is encoded by minimum/1-saddle pairs. In 3D, $(d-1)$ -saddle/maximum pairs characterize the persistence of the voids of $f_{-\infty}^{-1}(w)$ while 1-saddle/2-saddle pairs encode the lifetime of its independent cycles. In practice, the features of interest in the data are often characterized by the extrema of the field f . Thus we will only focus on extremum/saddle pairs in the following.

The topological persistence of each critical pair gives a measure of importance on the corresponding extremum of the field, that has been shown to be reliable to distinguish between noise and important features. The *Persistence diagram* [21] of f , noted $\mathcal{D}(f)$, is a visual representation of the ensemble of features in the data, where each persistence pair (c_0, c_1) is embedded as a point in the 2D plane, at coordinates $(f(c_0), f(c_1))$. The persistence of each pair can thus be read in the diagram as the height of the point to the diagonal. Consequently, each topological feature of $f_{-\infty}^{-1}(w)$ can be visualized in the diagram as a bar (Fig. 2), whose height indicates its importance in the data. Large bars corresponding to high persistence features stand out visually, while low persistence pairs, likely to be associated with noisy features, are represented by small bars in the vicinity of the diagonal. The persistence diagram is a concise visual depiction of the repartition of features in the data and has been shown to be a stable [15] and useful tool for data summarization tasks. As seen in Fig. 2, it encodes the number, ranges and salience of features of interest, and gives hints about the level of noise in the data.

The *Wasserstein distance* is a well-established metric between persistence diagrams. Originally from the field of Transportation theory, it is defined as the cost of an optimal assignment between two diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$, where the cost of matching a point $a = (x_a, y_a) \in \mathcal{D}(f)$ to a point $b = (x_b, y_b) \in \mathcal{D}(g)$ is given by a pointwise distance d_q in the 2D birth/death space of the diagrams:

$$d_q(a, b) = (|x_b - x_a|^q + |y_b - y_a|^q)^{1/q} = \|a - b\|_q \quad (1)$$

By convention, $d_q(a, b) = 0$ if both a and b are on the diagonal ($x_a = y_a$ and $x_b = y_b$). The L_q -Wasserstein distance between $\mathcal{D}(f)$ and $\mathcal{D}(g)$ is then given by:

$$W_q(\mathcal{D}(f), \mathcal{D}(g)) = \min_{\phi \in \Phi} \left(\sum_{a \in \mathcal{D}(f)} d_q(a, \phi(a))^q \right)^{1/q} \quad (2)$$

where Φ is the set of all possible maps ϕ mapping each point $a \in \mathcal{D}(f)$ to a point $b \in \mathcal{D}(g)$ or to its projection onto the diagonal. When q converges to infinity, the Wasserstein distance converges to the *Bottleneck* distance, another practical metric that measures the worst mismatch between $\mathcal{D}(f)$ and $\mathcal{D}(g)$:

$$W_\infty(\mathcal{D}(f), \mathcal{D}(g)) = \min_{\phi \in \Phi} \left(\max_{a \in \mathcal{D}(f)} \|a - \phi(a)\|_\infty \right) \quad (3)$$

The persistence diagram is *stable* under these two metrics, which intuitively means that a small variation in the scalar field entails a small difference in the resulting distances. In particular, the Bottleneck distance between two diagrams is bounded by the L_∞ distance between the corresponding scalar fields [15]:

$$W_\infty(\mathcal{D}(f), \mathcal{D}(g)) \leq \|f - g\|_\infty \quad (4)$$

2.4 Edge-Nested Hierarchical Data Representation

To design progressive algorithms for topological data analysis, Vidal et al. [78] used a progressive representation of the input data. This representation is based on an *edge-nested* multiresolution hierarchy of the input domain \mathcal{M} . The hierarchy $\mathcal{H} = \{\mathcal{M}^0, \mathcal{M}^1, \dots, \mathcal{M}^h\}$ is a growing sequence of PL manifolds, such that all the vertices

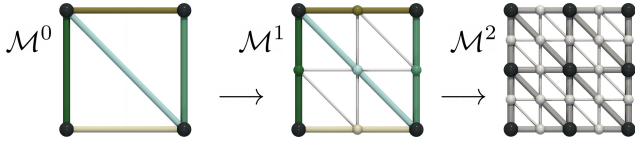


Figure 3: Edge-nested triangulation hierarchy of a regular grid. New vertices are only inserted on existing edges, one per edge, splitting each old edge in two. Old and new edges are shown (right, \mathcal{M}^2) in grey and white respectively; old and new vertices are shown in black and white respectively. New edges only connect new vertices.

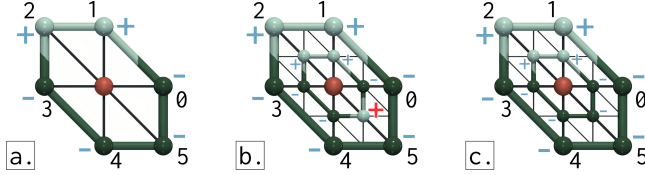


Figure 4: Evolution of the upper link (light green) and the lower link (dark green) of a vertex v (red sphere) when progressing down the hierarchy. The topological structure of the link at level i (a) remains unchanged at level $i+1$, thanks to the edge-nested property of the hierarchy. In the example (b), neighbor 5 is *non-monotonic*: it switches its polarity (from blue minus to red plus), which modifies the criticality of v (saddle point in \mathcal{M}^{i+1}). If the link polarity of v is *not* changed (c), v is called a *Topologically Invariant* vertex and it maintains its criticality.

present in one level \mathcal{M}^i of \mathcal{H} are present in all the following levels \mathcal{M}^j , $j > i$. This naturally yields a hierarchy of PL scalar fields $\{f^0, \dots, f^h\}$ such as each vertex maintains the same scalar value from its level of insertion i forward: $\forall v \in \mathcal{M}^i, \forall j \mid i \leq j \leq h, f^j(v) = f^i(v) = f(v)$. The *edge-nested* property states that *new* vertices inserted at a level i of the hierarchy are exclusively inserted on existing edges (connecting *old* vertices) at level $i-1$, one per edge, and that new edges only connect new vertices (see Fig. 3). We refer the reader to Vidal et al. [78] for a detailed formalization.

Edge-nested hierarchies can be easily generated from regular grids. Given a d -dimensional regular grid \mathcal{G}^0 , a valid PL d -manifold \mathcal{M}^h can be obtained from \mathcal{G}^0 by considering its Kuhn triangulation [44] (Fig. 3). A sequence $\{\mathcal{G}^0, \mathcal{G}^1, \dots, \mathcal{G}^h\}$ of decimated grids can be recursively defined from \mathcal{G}^0 , where \mathcal{G}^i is obtained for $i > 0$ by sub-sampling only vertices with even coordinates in \mathcal{G}^{i-1} . The edge-nested triangulation hierarchy $\mathcal{H} = \{\mathcal{M}^0, \mathcal{M}^1, \dots, \mathcal{M}^h\}$ is then defined by considering the triangulation \mathcal{M}^i of each level \mathcal{G}^{h-i} . Further implementations details can be found in Vidal et al. [78].

2.5 Topological Invariant Vertices

The advantage of using an edge-nested triangulation hierarchy to process the data resides in the fact that topological information can be computed on the vertices of \mathcal{M}^i and remain valid throughout the rest of the hierarchy. In particular, as discussed by Vidal et al. [78], when progressing from a level of the hierarchy to the next, the topological structure of the link is invariant for all old vertices: their number of neighbor vertices remains the same, and the adjacency relations between neighbors are preserved. This important property (Fig. 4) enables the efficient identification of vertices that do not change criticality when progressing to the next hierarchical level.

For each vertex v of \mathcal{M}^i , its *link polarity* (blue $+/-$ glyphs in Fig. 4) indicates for each one of its neighbors $n \in Lk(v)^i$, whether $n \in Lk^+(v)^i$ (positive polarity) or $n \in Lk^-(v)^i$ (negative polarity). As the structure of $Lk(v)^i$ is invariant, the criticality of v is characterized by the polarity of its link. In particular, if the link polarity of v stays unchanged when progressing a level down the hierarchy, the criticality of v remains the same and does not need to be recomputed. Given an edge (v_0, v_1) of \mathcal{M}^i that gets subdivided into two old edges

(v_0, v_n) and (v_n, v_1) along the new vertex v_n of \mathcal{M}^{i+1} , v_n is said to be *monotonic* if $f(v_n) \in (f(v_0), f(v_1))$, assuming that $f(v_0) < f(v_1)$. Otherwise, v_n is called a *non-monotonic* vertex, and the polarity of the link of v_0 or v_1 is necessarily changed, as shown in Fig. 4b. On the opposite, an old vertex keeps the same link polarity (and thus the same criticality) if all its new neighbors are *monotonic* (Fig. 4c). It is then called a *Topologically Invariant old vertex*. Potentially, depending on the regularity of the data, a large part of the old vertices at level i can be topologically invariant. Similarly, the criticality of a part of the *new* vertices can be deduced from the hierarchy with the identification of non-monotonic vertices. It can be shown [78] that if a new vertex v of \mathcal{M}^i is monotonic and if all its *new* neighbors are also monotonic, then v is necessarily a *regular* point. It is called a *Topologically Invariant new vertex*.

It has been shown [78] that Topologically Invariant vertices (TIs) represent a large part of the data in practice (around 70% on average on a diverse sample of real-life data sets). Thus, the edge-nested hierarchical representation of the data enables a highly efficient detection of these TI vertices. It proved itself useful to carry out topological data analysis tasks such as the extraction of critical points or the computation of a persistence diagram, as significant shortcuts can be made in the computation for TI vertices.

3 OVERVIEW

Fig. 5 provides an overview of our approach, which revisits the progressive framework of Vidal et al. [78], to derive a fast approximation algorithm with strong guarantees. First, we exploit their multiresolution hierarchy of the input data (Sec. 2.4) to quickly update, down to the finest hierarchy level, the polarity of each vertex (a local information used to identify critical points). This step is described in Sec. 4.1. During the hierarchy traversal, in contrast to their original approach, we artificially increase the number of *topologically invariant vertices* (Sec. 2.5) in order to significantly speedup the computation, through a procedure called *vertex folding*, which artificially degrades the input data. This step is described in Sec. 4.2. The data degradation induced by the vertex folding procedure is precisely controlled in the process, to provide strong guarantees on the approximation error of the output. This is described in Sec. 4.3. Finally, we describe in Sec. 4.4 how to handle degenerate configurations such as flat plateaus. Overall, our new approach involves three major differences to the progressive framework of Vidal et al. [78], which are detailed in the rest of this section:

1. Our new approximation algorithm is *not* progressive: it does not generate a sequence of progressively refined outputs. Instead, our traversal of the multiresolution hierarchy only updates a minimal amount of information (the vertex polarity). The criticality of each vertex is only evaluated *after* the hierarchy traversal is finished (while the criticality is updated at each hierarchy level in [78]).
2. Our approximation approach is based on a multiresolution degradation of the input data, which accelerates the overall computation, while maintaining a controlled output error.
3. Overall, in contrast to the progressively interrupted results of Vidal et al. [78], our approximated output is provided with strong guarantees on its approximation error (in terms of relative Bottleneck distance) and with a significantly improved practical accuracy (in terms of the L_2 Wasserstein distance).

4 TOPOLOGY APPROXIMATION

This section presents our novel approach for the controlled approximation of an extremum/saddle persistence diagram. In the following, we focus on the case of minimum/1-saddle pairs ($(d-1)$ -saddle/maximum pairs being treated symmetrically). Our method is based on *folding* operations for non-monotonic vertices. As the

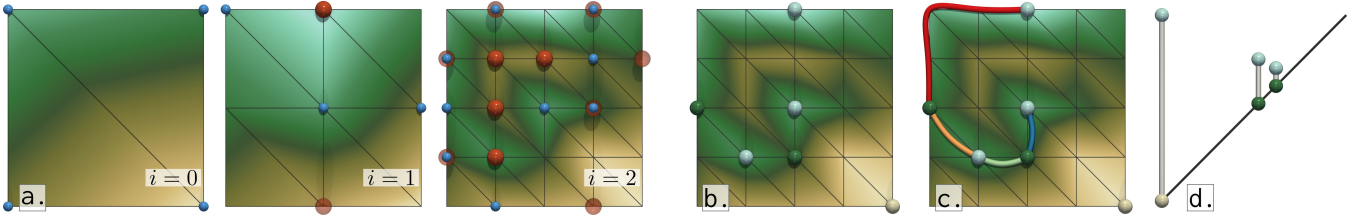


Figure 5: Overview of our approach. First, the traversal of the hierarchy (a) enables the efficient detection of the vertices (blue spheres) that are **not** topologically invariant (TI), and for which the criticality must be computed. Non-monotonic vertices (red spheres) can be *folded* (transparent red spheres) during the traversal, *i.e.* reinterpolated to artificially increase the number of TI vertices. Second (b), the criticality of all non-TI vertices detected in the first step is computed, to identify the critical points of the approximated field (spheres, cyan: maxima, green: saddles, beige: minimum). Third, the saddle points are used to seed integral lines ending at extrema (c), from which the persistence diagram is deduced (d).

hierarchy is processed, non-monotonic vertices are inserted at each level. When a non-monotonic vertex is inserted in the hierarchy on a new edge, we can purposely decide to reinterpolate this vertex to enforce its monotony, and hence accelerate the computation of its criticality. The resulting error is the difference between the real scalar value at this vertex and the interpolated value, which bounds the Bottleneck error on the diagram estimation, as detailed next.

4.1 Hierarchy Processing

This section presents our computation strategy in the case where the approximation error ε is set to 0 (*i.e.* exact computations). Given an input edge-nested triangulation hierarchy $\mathcal{H} = \{\mathcal{M}^0, \mathcal{M}^1, \dots, \mathcal{M}^h\}$, persistence diagrams are evaluated in three steps. First, the hierarchy is completely traversed, from the coarsest level \mathcal{M}^0 to the finest \mathcal{M}^h , to efficiently detect topologically invariant vertices, for which the criticality can be efficiently estimated in a second step, at the last level of the hierarchy only (\mathcal{M}^h). Third, the persistence diagram is computed from the saddle points identified at the second step.

1) Hierarchy traversal: In order to identify topologically invariant vertices, we compute the link polarity (Sec. 2.5) of the vertices for each level of the hierarchy. The link polarity of a vertex v is encoded in our setting as an array of bits, one per neighbor n of v , denoting whether n is higher or lower than v . The size of the link polarity is the same for each level of \mathcal{H} (Sec. 2.5): at most 6 bits in 2D and 14 bits in 3D. At each level i , the polarity of new vertices is initialized, while the polarity of old vertices is updated to account for the insertion of non-monotonic vertices. The detection of non-monotonic vertices enables the fast identification of regular points: *new* topologically invariant vertices are known to be regular points of f^i , and *old* topologically invariant vertices keep the same criticality at level i than at level $i-1$ (Sec. 2.5). We leverage these informations to avoid the explicit computation of the criticality for some of the vertices. As a new topologically invariant vertex v is identified in \mathcal{M}^i , it is flagged as a regular vertex. If its link polarity is not changed in the remaining levels (*e.g.* v is topologically invariant through the rest of the hierarchy), v is guaranteed to be a regular vertex of \mathcal{M}^h and no further computation will be needed for it.

2) Critical points: At \mathcal{M}^h , we compute explicitly the criticality of the vertices which are not yet guaranteed to be regular (as described above). The criticality of a vertex v is computed by enumerating the connected components of $Lk^+(v)$ and $Lk^-(v)$ (Sec. 2.2).

3) Persistence diagram: The minimum/1-saddle persistence diagram is deduced from the critical points. In particular, for each saddle point s , a backward integral line is launched from each connected component of $Lk^-(s)$, if there are more than one. These integral lines end at a local minimum m . For each integral line, we back-propagate the vertex index of m back to s . This way, we build for each saddle s the list $\{m_0, m_1, \dots, m_j\}$ of found local minima, yielding at least a minimum per connected component of sub-level sets merging at s . A merge event can thus be represented by a triplet (s, m_i, m_j) . All found triplets are then sorted in ascending values of

s , and the merge events are processed in this order with a UnionFind (UF) data structure [18]. A persistence pair is created when two different components are merged. We refer to Vidal et al. [78] for more details, as this last step of our method is identical to theirs.

4.2 Vertex Folding

This section describes the variations of the above strategy for the case where $\varepsilon \neq 0$, to decrease the computational cost. Specifically, we present a strategy to artificially increase the number of topologically invariant vertices during the hierarchy traversal (step 1), which will consequently result in skipping the estimation of vertex criticality (step 2) for a larger number of vertices (hence the overall speedup).

A new vertex n , appearing at a level i of the hierarchy, is inserted at the center of an old edge (o_0, o_1) of \mathcal{M}^{i-1} (Sec. 2.4). Assuming that $f(o_0) < f(o_1) < f(n)$, n is a non-monotonic vertex and impacts the link polarity of o_1 . The apparition of such vertices reduces the overall performance as these will trigger explicit criticality computations in step 2. To reduce the number of non-monotonic vertices inserted in \mathcal{M}^i , we can choose to reinterpolate a non-monotonic vertex n between its two new neighbors to enforce its monotony. We note the resulting monotonic vertex \hat{n} and say that this vertex is *folded*. We define its new approximated value $\hat{f}(\hat{n})$ as the interpolation of the approximated values of its two old neighbors: $\hat{f}(\hat{n}) = (\hat{f}(o_0) + \hat{f}(o_1)) / 2$. The values $\hat{f}(o_0)$ and $\hat{f}(o_1)$ are themselves either the result of a linear interpolation if o_0 or o_1 have been previously *folded*. Otherwise, they are equal to $f(o_0)$ and $f(o_1)$. The linear interpolation is preferred here to alternatives as it provides a good balance between accuracy and efficiency.

Formally, we build a sequence $\{f^0, \hat{f}^1, \dots, \hat{f}^h\}$ of PL scalar fields defined on each hierarchy level. The sequence is defined recursively:

1. $f^0 = f^0$
2. For each old vertex o of \mathcal{M}^i , $\hat{f}^i(o) = \hat{f}^{i-1}(o)$
3. For each new vertex n of \mathcal{M}^i that is not *folded*, $\hat{f}^i(n) = f^i(n)$
4. For each *folded* new vertex \hat{n} of \mathcal{M}^i on the edge (o_0, o_1) ,

$$\hat{f}^i(\hat{n}) = \frac{\hat{f}^{i-1}(o_0) + \hat{f}^{i-1}(o_1)}{2}$$

We note \mathcal{F}^i the set of folded vertices at level i (with $\mathcal{F}^i \subset \mathcal{M}_0^i$). By construction, folded new vertices are monotonic. Fig. 6 illustrates how a higher amount of folded vertices at level i implies a higher number of topologically invariant vertices identified on \mathcal{M}^i . In particular, if all non-monotonic vertices are folded at level i , all vertices of \mathcal{M}^i are topologically invariant.

4.3 Bottleneck Error Control

Computing persistence diagrams with *vertex folding* results in approximations of the exact result $\mathcal{D}(f)$, given by $\mathcal{D}(\hat{f})$ (diagram of the approximated field \hat{f}). Then the resulting approximation error (in terms of Bottleneck distance) is given by [15]:

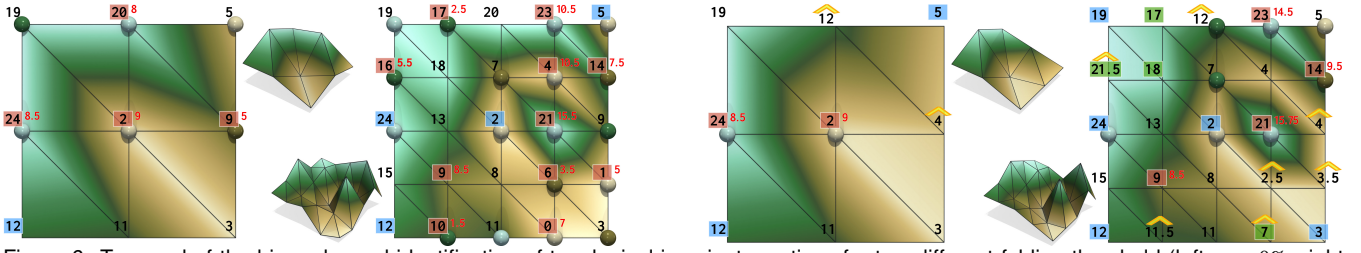


Figure 6: Traversal of the hierarchy and identification of topological invariant vertices for two different folding threshold (left: $\varepsilon = 0\%$, right: $\varepsilon = 30\%$). The numbers denote the values of the approximated scalar field \hat{f} . Red squares indicate the non-monotonic new vertices, whose folding error δ_ε is labelled in red. On the left, no vertex is folded and no approximation is made on the scalar field (i.e. $\hat{f} = f$). It results in a high number of non-monotonic vertices and a low number of topologically invariant old and new vertices (respectively blue and green squares). In contrast, a folding threshold $\varepsilon = 30\%$ is applied on the right. Every non-monotonic vertex n with a folding error $\delta(n) \leq 8$ gets folded (yellow hats). This reduces the number of non-monotonic vertices and more than doubles the number of TI vertices (blue and green squares) of the full precision approach.

$W_\infty(\mathcal{D}(\hat{f}), \mathcal{D}(f)) \leq \|\hat{f} - f\|_\infty$, which is rather easy to estimate. For each new vertex n inserted at level i of the hierarchy on the edge (o_0, o_1) , we define its *folding error* $\delta(n)$ as the difference between its original scalar value and its reinterpolation value at its level of insertion: $\delta(n) = \left| \left(\hat{f}^i(o_0) + \hat{f}^i(o_1) \right) / 2 - f(n) \right|$. Then, we have:

$$\|\hat{f} - f\|_\infty = \max_{\hat{n} \in \mathcal{F}^h} \delta(\hat{n}) = \max_{\hat{n} \in \mathcal{F}^h} |\hat{f}(\hat{n}) - f(\hat{n})|$$

In the light of these observations, we use the following folding strategy as we process the hierarchy. Given a target approximation error ε , the hierarchy is processed as described in Sec. 4.1, except that vertex polarity is estimated from the approximated field \hat{f} . For each level i , we choose to fold non-monotonic vertices n with an error $\delta(n) < \varepsilon$. Monotonic vertices or non-monotonic vertices with a higher folding error get added into the hierarchy without being reinterpolated. Let $\hat{f}_\varepsilon : \mathcal{M} \rightarrow \mathbb{R}$ be the final field approximation (after the hierarchy traversal is completed), given the target approximation error ε . Then it is clear that $\forall v \in \mathcal{M}$, $\delta(v) \leq \varepsilon$. Thus:

$$W_\infty(\mathcal{D}(\hat{f}_\varepsilon), \mathcal{D}(f)) \leq \|\hat{f}_\varepsilon - f\|_\infty = \max_{\hat{n} \in \mathcal{F}^h} \delta(\hat{n}) \leq \varepsilon$$

The computation must reach the last level h , to accurately capture $\max \delta(\hat{n})$ (required for guarantees), thus it is not progressive. In the remainder, ε is noted as a percentage of the data range ($\varepsilon = 0\%$ is the exact computation, $\varepsilon = 100\%$ folds all non-monotonic vertices).

4.4 Monotony offsets

We now discuss how to handle degenerate flat plateaus, which become more frequent given our vertex folding strategy. The input scalar field f is injective on the vertices of \mathcal{M} (Sec. 3). This is enforced in practice with an offset field $\mathbb{O} : \mathcal{M}_0 \rightarrow \mathbb{N}$, typically corresponding for each vertex to its offset in memory (Fig. 7a). \mathbb{O} is then used to disambiguate vertices with identical scalar values.

In theory, a *folded* vertex \hat{n} is guaranteed to be monotonic. However in practice, if $\delta(\hat{n})$ falls below the precision of the data type used to encode the scalar field, a flat plateau emerges. This occurs frequently for instance when the input data is expressed with integers. Then, given a new folded vertex \hat{n} inserted on an edge (o_0, o_1) , we may have: $\hat{f}(\hat{n}) = \hat{f}(o_0)$, which means \hat{n} and o_0 will be disambiguated in the algorithm by their offset \mathbb{O} . However, this can introduce undesired monotony changes (Fig. 7b, red squares).

To guarantee the monotony of folded vertices, we introduce a *monotony offset* on each vertex v , noted $\mathbb{M}(v)$, which is modified when the vertex gets folded. The purpose of the monotony offset \mathbb{M} is to take over the regular offset \mathbb{O} if it contradicts the monotony of newly folded vertices. Given a new *folded* vertex \hat{n} inserted on an edge (o_0, o_1) at level i , that is non-monotonic with respect to o_0 (i.e. $\hat{f}(\hat{n}) < \hat{f}(o_0) < \hat{f}(o_1)$ or $\hat{f}(\hat{n}) > \hat{f}(o_0) > \hat{f}(o_1)$), we set:

$$\mathbb{M}(\hat{n}) = \begin{cases} \mathbb{M}(o_0) - 2^{h-i} & \text{if } \hat{f}(\hat{n}) < \hat{f}(o_0) < \hat{f}(o_1) \\ & \text{and } \mathbb{O}(\hat{n}) > \mathbb{O}(o_0) \\ \mathbb{M}(o_0) + 2^{h-i} & \text{if } \hat{f}(\hat{n}) < \hat{f}(o_0) < \hat{f}(o_1) \\ & \text{and } \mathbb{O}(\hat{n}) < \mathbb{O}(o_0) \\ \mathbb{M}(o_0) & \text{else} \end{cases}$$

The monotony offset is initially set to zero for all new vertices and modified only in case of vertex folding. Then, the field \mathbb{M} explicitly encodes the monotony of newly folded vertices (Fig. 7). The monotony offsets are used to disambiguate the comparison of two vertices of identical approximated scalar value in the rest of the approach (criticality estimation, integral lines, etc).

4.5 Parallelism

Our approach can be easily parallelized using shared-memory parallelism. The first step of our approach, the traversal of the hierarchy, can be trivially parallelized on the vertices, as all operations are local to a vertex (Sec. 4.2). However the hierarchy must be processed in sequential, which implies synchronization between the different levels. The computation of the criticality for non topologically invariant vertices is also completely parallel. The computation of the persistence diagram from the critical points can be parallelized on the saddle points, however locks are necessary for the parallel computation of integral lines, in order to back-propagate the indices of found extrema. The saddle points are sorted in parallel, using the GNU implementation [69]. Finally, processing the merge events represented by the triplets is a sequential task, but represents only a fraction of the total sequential computation time (less than 1% in practice).

4.6 Uncertainty

By construction, our approach induces a Bottleneck error of ε , which corresponds to a maximum mismatch of ε between the pairs of $\mathcal{D}(\hat{f}_\varepsilon)$ and these of $\mathcal{D}(f)$. This means that some approximated persistence pairs (with a persistence below 2ε) may not be present in the exact diagram (as they may be matched to the diagonal at a cost lower than ε). We call these pairs *uncertain*. In contrast, the approximated pairs with a persistence beyond 2ε will certainly be present in the exact diagram, and their exact location is bounded within a square of side 2ε . We call these pairs *certain*. Thus, we can visually convey the level of uncertainty of our approximations directly in the persistence diagrams (Fig. 8). For this, we use a red band to indicate *uncertain* pairs, and we draw the bounding squares for *certain* pairs (Sec. 5.3).

5 RESULTS

This section presents experimental results obtained on a variety of datasets, available on public repositories [48, 75]. We implemented our approach in C++, as modules for the Topology ToolKit (TTK) [6, 73]. The experiments were carried out on a desktop computer with two Xeon CPUs (3.0 GHz, 2×4 cores) and 64 GB of RAM.

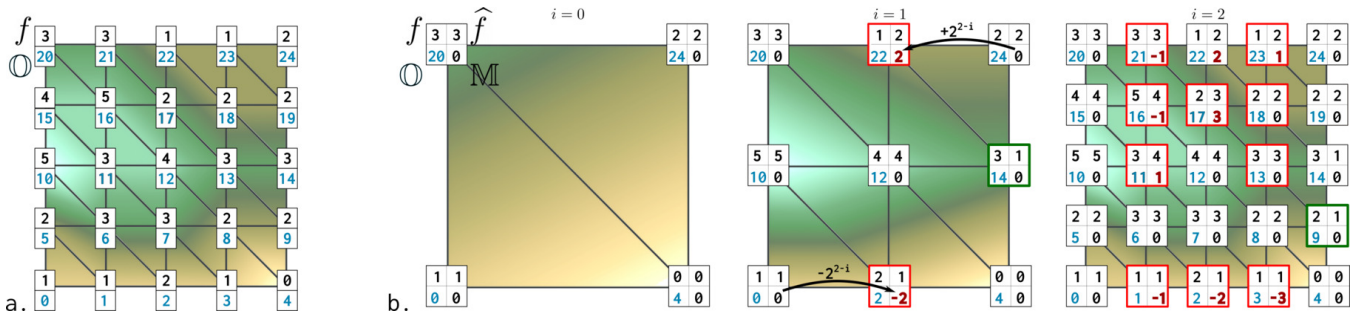


Figure 7: Monotony offsets on a toy example of an **integer** field f (a: black numbers) defined on a 2D grid. The injectivity of f is guaranteed by the offset field \odot (a: blue numbers). As the hierarchy is processed (b), some vertices get folded (red and green squares) according to a given error threshold. Due to the precision of the field (here integer precision), their interpolated value (numbers on the right) might be identical to that of an old neighbor. In some cases (red squares), the folding actually entails a monotony change as the offset field \odot (blue numbers) provides the wrong order relation between neighbor vertices. If such an event occurs, the monotony offset of the folded vertex is updated (red numbers) to enforce monotony. Green squares denote folding cases where \odot does not contradict the folding monotony. At the finest resolution (b, rightmost), this results in plateaus (bottom row of the grid) where the injectivity of \hat{f} is guaranteed and where the monotony is correctly enforced.

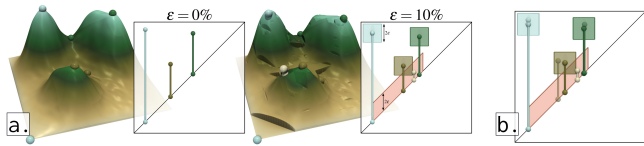


Figure 8: Approximation uncertainty visualization. The diagram at $\epsilon = 10\%$ (a., right) exhibits one *uncertain* pair within the red band, which is absent from the exact result (a., left). Squares bound the correct location of *certain* pairs (b., transparent: exact pairs).

Table 1: Increase of the number of topologically invariant (TI) vertices for different levels of approximation. For real-world datasets (*MinMax* and *Random* excluded), the average proportion of TI vertices rises from 70% for the full precision approach (i.e. $\epsilon = 0\%$) to 94% for a relative Bottleneck error of 5% ($\epsilon = 5\%$).

Dataset	$\sum_{i=0}^h \mathcal{M}_i^h $	% TI			
		$\epsilon = 0\%$	$\epsilon = 1\%$	$\epsilon = 5\%$	$\epsilon = 10\%$
At	931,102	64.7	95.3	96.1	96.3
SeaSurfaceHeight	1,384,636	61.4	90.8	96.7	98.3
Ethanediol	2,057,380	77.7	97.4	97.6	97.6
Hydrogen	2,413,516	73.5	97.8	97.3	97.4
Isabel	3,605,596	44.5	80.8	91.6	93.5
Combustion	4,378,378	65.5	89.6	96.3	97.3
Boat	4,821,318	88.5	96.9	97.0	97.2
MinMax	18,994,891	99.3	99.5	99.5	99.5
Aneurism	19,240,269	95.6	96.1	97.4	98.2
Foot	19,240,269	64.4	66.6	73.7	86.9
Heptane	31,580,914	82.2	96.4	98.2	98.7
Random	18,117,510	0.9	0.9	1.0	1.1
Backpack	111,929,613	39.1	77.2	94.7	97.7

5.1 Time Performance

The time complexity of our approach is similar to the complexity of the non-interrupted algorithm by Vidal et al. [78]. The first two steps of our approach, that amount to the hierarchy traversal and the computation of critical points, have a linear complexity in the number of vertices: $\mathcal{O}(\sum_{i=0}^h |\mathcal{M}_i^h|)$. The third step, to compute the saddle-extremum persistence from the critical points, is identical to the approach by Vidal, except that we compute the persistence diagram exclusively on the last, finest level of the hierarchy. It has a practical time complexity [78] of $\mathcal{O}(|\mathcal{M}_h^h| + n_s \log n_s + n_s \alpha(n_m))$, where α stands for the inverse of the Ackermann function, and n_s and n_m respectively denote the number of saddle points and extrema.

Table 1 reports the number of TI vertices for all datasets, for various approximation errors. The column $\epsilon = 0$ corresponds to the numbers of TI vertices reported by Vidal et al. [78] in the exact case. For the majority of datasets, we observe a large increase in

Table 2: Sequential computation times (in seconds) of our approach for the approximation of persistence diagram, for different approximation errors. The column *Default* reports the run time of the default approach in the Topology Toolkit [34]. The last column indicates the speedup of our approach with an approximation error of 5%, against the fastest of the two exact methods (left). Bold numbers indicate the smallest ϵ providing a speedup over reference approaches.

Dataset	Default [34]	Progressive [78]	Ours			5% speedup
			$\epsilon = 1\%$	$\epsilon = 5\%$	$\epsilon = 10\%$	
At	0.27	0.25	0.14	0.15	0.17	38.5%
SeaSurfaceHeight	0.48	0.38	0.29	0.26	0.25	30.9%
Ethanediol	0.48	0.43	0.28	0.31	0.33	28.7%
Hydrogen	0.99	0.64	0.43	0.48	0.47	24.6%
Isabel	1.29	1.49	0.95	0.89	0.92	30.7%
Combustion	2.55	1.37	0.99	0.90	0.85	34.1%
Boat	1.22	0.82	0.97	1.06	1.02	-28.1%
MinMax	4.01	1.92	2.14	2.28	2.13	-18.4%
Aneurism	4.66	3.43	3.62	3.52	3.00	-2.7%
Foot	9.86	10.42	10.38	8.14	6.14	17.4%
Heptane	8.09	7.41	5.41	5.18	5.16	30.1%
Random	37.29	30.77	28.95	29.04	30.46	5.6%
Backpack	77.28	107.31	62.06	40.11	31.76	48.1%

the number of TI vertices, from a proportion of 70% on average on the real-life datasets to a proportion of 90% for a small error of 1%, and 94% for a mild tolerance of 5% on the approximation. The largest increase in the number of TI vertices is reported for *Backpack* (being a large and noisy dataset). This table confirms that our strategy of *vertex folding* (Sec. 4.2) indeed implies a sensible increase in the number of TI vertices, even for mild approximation errors. Regarding the criticality estimation (step 2, Sec. 4.1), as no computation is needed for the vertices which remained topologically invariant throughout the hierarchy (Sec. 4.1), a higher proportion of TI vertices in the data is thus likely to significantly decrease the computational workload, resulting in lower computation times.

Table 2 details the sequential computation times of our approach for different approximation errors. They are compared with public implementations of exact algorithms, both available in TTK [73]: the *progressive* approach by Vidal et al. [78] (run up to the finest resolution, hence producing an exact result), and the *default* algorithm used in TTK [34] (run at the finest hierarchy level). The last column of Tab. 2 present the speedups obtained with a Bottleneck error tolerance of 5%, compared with the fastest of either reference approaches. We observe an average reduction of the run times of 18% on real-world datasets, which confirms that our strategy of maximizing the number of TI vertices effectively reduces the computation times. The observed speedups are consistent with the increases in the proportion of TI vertices reported in Tab. 1: a large increase in the number of TI vertices implies an important reduction of the computation time. Interestingly, we find our method most beneficial on datasets that initially present a low amount of TI vertices. This usually corresponds to a high level of noise, which impedes both

Table 3: Parallel computation times (in seconds, on 8 physical cores) of our algorithm with different approximation errors. The presented speedups relate to the sequential run times.

Dataset	Default [34]	Progressive [78]	$\epsilon = 1\%$	speedup	$\epsilon = 5\%$	speedup	$\epsilon = 10\%$	speedup
At	0.20	0.07	0.05	2.61	0.07	2.24	0.07	2.25
SeaSurfaceHeight	0.21	0.13	0.08	3.65	0.07	3.91	0.07	3.77
EthaneDiol	0.21	0.11	0.08	3.69	0.09	3.60	0.10	3.35
Hydrogen	0.72	0.20	0.13	3.18	0.16	2.95	0.17	2.80
Isabel	0.47	0.46	0.54	1.77	0.28	3.15	0.30	3.05
Combustion	0.34	0.51	0.54	1.83	0.31	2.92	0.25	3.41
Boat	0.29	0.29	0.33	2.94	0.37	2.85	0.34	2.98
MinMax	0.80	0.61	0.69	3.11	0.54	4.23	0.54	3.97
Aneurism	2.03	1.51	1.62	2.23	1.36	2.59	1.33	2.25
Foot	3.12	2.34	2.65	3.91	1.91	4.25	2.04	3.01
Heptane	2.44	2.35	2.08	2.60	1.33	3.88	1.51	3.41
Random	27.04	8.47	7.15	4.05	7.77	3.74	8.89	3.43
Backpack	30.36	24.88	12.63	4.92	8.50	4.72	7.02	4.53

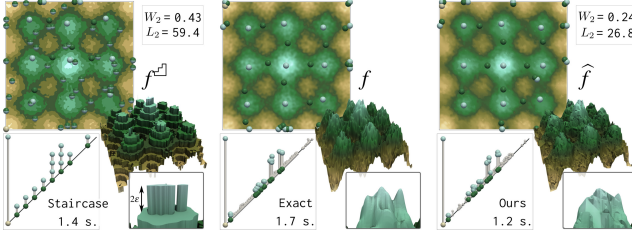


Figure 9: Comparison of our approach to a naive baseline approximation (*staircase* field), for the same tolerance of 5% on the Bottleneck error. Our approach (right) provides an approximated persistence diagram that resembles the exact result (center), and is around 2 times closer in terms of the Wasserstein distance (W_2). The high persistence critical points (spheres, 2D domain) capture more precisely the features of the data in our case. Our approximated field is also two times closer (L_2 norm) to the exact field than the naive approximation.

reference methods. In particular, the highest speedup is achieved on *Backpack*, our largest, noisiest real-world dataset, with a reduction of computation time of nearly 50%. In contrast, our method fails to reduce the computation times on smooth datasets such as *MinMax*, *Boat* or *Aneurism*, for which the *progressive* approach really shines. These datasets exhibit high initial proportions of TI vertices, which limits the increase in TI vertices enabled by our approach (Tab. 1). For *Random*, the interpolation cost of folded vertices seems to counterbalance the speedup induced by the increase in TI vertices.

Table 3 lists the computation times obtained with the parallel version of our algorithm. We find an overall average parallel efficiency of 43% with an error level of 5%, which is on par with the progressive approach [78]. Although the traversal of the hierarchy can be trivially parallelize over vertices, it is subject to synchronization steps between hierarchy levels. The last step of the approach, deducing the persistence diagram from the critical points, is less balanced. Indeed, the parallel computation of integral lines between saddle points and extrema (Sec. 4.1) necessitates locks.

Table 4: Accuracy comparison between our approach and a naive baseline approximation based on the computation of a *staircase* function, for the same relative Bottleneck error of 5%. Our approximations are more accurate in average, both in terms of the L_2 distance of the approximated field (2 times more accurate) and in terms of the L_2 -Wasserstein distance to the exact persistence diagram (5 times).

Dataset	Staircase L_2	Ours L_2	Ratio	Staircase W_2	Ours W_2	Ratio
At	276.66	75.31	3.67	3.31	1.01	3.29
SeaSurfaceHeight	92.3	58.05	1.59	8.75	1.69	5.17
EthaneDiol	337.55	73.2	4.61	1.78	0.61	2.9
Hydrogen	11.0	13.0	0.85	25.69	12.73	2.02
Isabel	3,591.99	1,569.98	2.29	42.26	8.08	5.23
Combustion	38.04	17.59	2.16	0.87	0.21	4.1
Boat	24.18	9.11	2.66	2.14	0.49	4.37
MinMax	117.55	0.37	314.77	0.0	0.0	-
Aneurism	6.0	9.0	0.67	1,198.14	128.61	9.32
Foot	16,006.11	10,784.33	1.48	5,859.67	1,419.51	4.13
Heptane	5.0	12.0	0.42	716.1	61.12	11.72
Random	29,270.73	3,468.14	8.44	23,863.56	1,038.1	22.99
Backpack	142.0	142.0	1.0	17,941.13	1,933.83	9.28

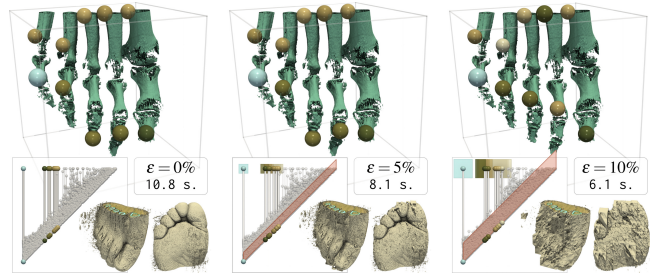


Figure 10: Approximated persistence diagrams on the *Foot* dataset. The 3D top views show the ten most persistent maxima (spheres), corresponding to the bones of the foot (isocontour, computed on the approximated field \hat{f}_ϵ). Our approximated diagrams correctly capture the most persistent features at a reduced computational cost.

5.2 Approximation Accuracy

By design, our approach produces approximated persistence diagrams with a guaranteed bound on the Bottleneck distance to the exact result (Sec. 4.3). In the following, we additionally evaluate the accuracy of our approximation by considering the L_2 -Wasserstein distance (Sec. 2.3, computed with the efficient progressive approximation [77] of TTK [73]) between our approximations and the exact result. We also evaluate $\|\hat{f} - f\|_2$ to quantify the pointwise error of the approximated field \hat{f} . Results are given in Table 4.

For comparison, we perform the same evaluation for a naive baseline approximation which provides identical guarantees. This baseline consists in computing, with an exact algorithm, the diagram of a *staircase* function f^\square , i.e. a quantized version of the input data f , with a quantization step of 2ϵ . By construction, the staircase function verifies $\|f^\square - f\|_\infty < \epsilon$ (Fig. 9) and its persistence diagram is then guaranteed not to exceed a Bottleneck error of ϵ [15]. Table 4 compares the accuracy of this baseline approximation to our algorithm. In terms of L_2 distance, our approximations of the input scalar fields are around 2 times more accurate on average (on real-world datasets, *Random* and *MinMax* excluded). This difference could be expected, as our method performs local and adaptive linear interpolations, while the staircase approach systematically flattens the data. However, our approach is also significantly more accurate with regards to the L_2 -Wasserstein distance to the exact persistence diagrams: 4 times in average on our real-world datasets.

Figure 9 further illustrates the limitations of the *staircase* baseline. For a given approximation error (5%), our method gives an approximated diagram that is visually more similar to the exact result, and which better depicts the number of salient features, as well as the noise in the data. In contrast, the diagram produced by the *staircase* approximation is more difficult to interpret, as the positions of persistence pairs are quantized on a grid in the 2D birth/death space, resulting in several co-located pairs, which cannot be distinguished visually. Our approximation of the scalar field is also closer to the exact field, both visually and in terms of the L_2 norm, enabling more accurate critical point approximations in the domain (top).

5.3 Qualitative Analysis

This section discusses the utility of our approximations from a qualitative point of view, for data analysis and visualization purposes. Fig. 1 shows the result of our approach on the *Backpack* dataset. In this example, our approximated diagrams correctly capture the high persistence maxima of the scalar field, which correspond to high density objects inside the bag (bottles, wires, and metallic parts of the bag). The approximate field \hat{f}_ϵ resulting from the vertex folding (Sec. 4.2) is more precise in the vicinity of features of high persistence. Indeed, the volume rendering in the top views of Fig. 1 shows clearly the objects inside the bag (high persistence maxima), while isocontours capturing the cloth of the bag (Fig. 1, bottom) illustrate

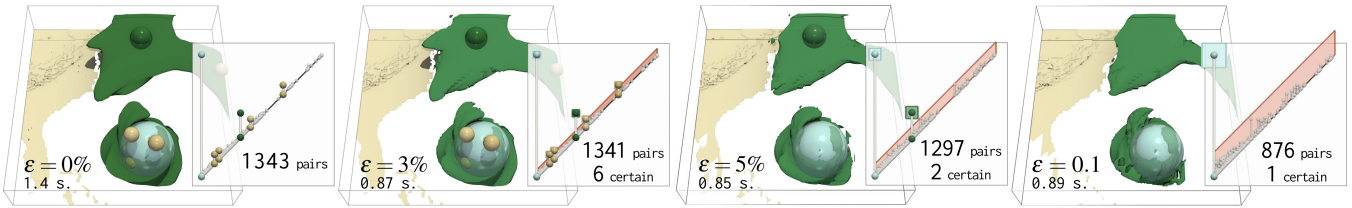


Figure 11: Approximated diagrams for the *Isabel* hurricane dataset. The most persistent maxima of the field (the magnitude of the wind velocity) are represented in the 3D view as spheres (scaled by persistence). The approximation uncertainty is visualized in the diagram: in the red band indicates *uncertain* persistence pairs that may not exist in the exact persistence diagram, and colored squares indicate a bound on the location of the persistence pairs that are *certain* to exist in the exact diagram. Our approximated persistence diagrams correctly capture these certain pairs in the order of their persistence, with higher persistence features being detected at higher tolerance on the Bottleneck error.

the deterioration of the field in this region. The same phenomenon can be noted for the *Foot* dataset (Fig. 10). More vertices are folded in the vicinity of low persistence features, typically the skin of the foot, and the level of deterioration of the field increases with the approximation error. Conversely, high persistence features (bones of the toes) are well captured.

The above observation suggests that our method provides a better approximation for persistent features, and a more degraded evaluation for less persistent structures, which was an original motivation for our approach (to focus the computational efforts on relevant structures). This is confirmed in Fig. 1 by the number of *noisy* pairs (of low persistence, within the red band) in the approximated diagrams, which is significantly lower than the amount of noisy pairs (of identical persistence) in the exact diagram (Fig. 1, in parenthesis).

Figure 12 compares our approximations to the progressive approach by Vidal et al. [78]. To generate an approximated result with the progressive approach, we interrupt its computation at the penultimate hierarchy level (the computation would become exact at the final level). As documented by its authors, such intermediate results can provide useful previews, but however, with no guarantee on the approximation error. This is illustrated in Fig. 12, where the progressive approximation fails at correctly capturing the maximum of largest persistence. In contrast, our method produces persistence diagrams that correctly convey the salience of the features, and are five times more accurate, in terms of the Wasserstein distance.

Indications about the approximation uncertainty (Sec. 4.6) can be displayed in the output diagrams. Figure 11 shows our approximations for the *Isabel* dataset. For each approximation error, the red band indicates uncertain pairs, which may not be part of the exact result. *Certain* pairs are represented with a square bounding their correct location. These glyphs give a good sense of the approximation uncertainty, and are useful to assess the reliability of the diagram. For instance, a large pair in the uncertain zone may indicate the presence of a medium persistence feature in the data. This can be confirmed with the computation of a slightly better estimation, as illustrated in Fig. 12 for the fourth most persistent feature.

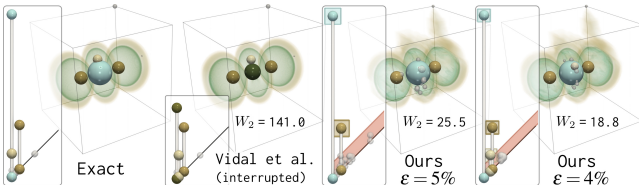


Figure 12: Approximation of persistence diagrams on the *hydrogen* dataset. The best interrupted result of Vidal et al. [78] fails at correctly capturing the global maximum (accurately detected only at the last level), resulting in a diagram that is 5 times less accurate than our 5% approximation (L_2 -Wasserstein distance to the exact result). In contrast, our approximation correctly captures the high persistence features of the data. On the far right, our 4% approximated diagram detects as *certain* the fourth most persistent maxima, which was marked as uncertain with $\epsilon = 5\%$.

6 LIMITATIONS AND DISCUSSION

Our approximations tend to generate much less low-persistence features than exact algorithms (Fig. 1), which can be an issue if features of interested are hidden among noisy features near the diagonal. On the upside, this characteristic of our approximations make them well suited for subsequent analysis and processing (e.g. distances and clustering), as diagrams are often thresholded in practice prior to further computations, to remove low persistence pairs anyway.

An important limitation of our approach, compared to the work of Vidal et al. [78], is its lack of progressivity. Indeed, to provide strong approximation guarantees, the hierarchy has to be completely traversed in our work and no intermediate result can be provided.

Another limitation is that our approach only supports saddle-extremum persistence pairs at the moment. However, from our experience, these correspond in practice to the key features users tend to be interested in.

Finally, our approach provides strong guarantees on the Bottleneck distance. Future work is needed for the theoretical study of the impact of our approximations on the L_2 Wasserstein metric.

7 CONCLUSION

This paper introduced a method for the approximation of the persistence diagram of a scalar field. Our work revisits the progressive approach by Vidal et al. [78], that generated preview diagrams upon interruption of a progressive framework. We addressed the main drawback of their approach, namely the lack of guaranteed error bounds on the diagram estimations. In contrast, we presented a novel algorithm that efficiently computes the approximation of a persistence diagram within a user controlled approximation error on the Bottleneck distance to the exact result. We showed that the approximated persistence diagrams are relevant for visualization and data analysis tasks, as they correctly describe the high persistence features in the data (*i.e.* the number and salience of important features), and they are more concise in practice than the exact diagrams. The uncertainty related to our approximations can be effectively depicted visually inside the diagrams.

We believe that the development of approximative approaches (with guarantees) for data analysis and visualization is an important and exciting research direction. They are especially relevant in the field of Topological Data Analysis, as most of the computational workload is typically spent in practice on the capture of small scale features (Fig. 1), whereas they usually are less relevant in the applications, and post-process simplification techniques are often applied to eliminate them anyway. In this context, a logical avenue for future work would be the development of approximative methods to revisit existing topological data representations (merge trees, Morse-Smale complexes, Reeb graphs).

ACKNOWLEDGMENTS

This work is partially supported by the European Commission grant ERC-2019-COG "TORI" (ref. 863464, <https://erc-tori.github.io/>).

REFERENCES

- [1] T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly*, 1970.
- [2] J. Bey. Simplicial grid refinement: on Freudenthal’s algorithm and the optimal number of congruence classes. *Numer. Math.*, 85:1–29, 1998.
- [3] 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.
- [4] S. Biasotti, B. Falcidieno, and M. Spagnuolo. Extended Reeb Graphs for Surface Understanding and Description. In *Discrete Geometry for Computer Imagery*, 2000.
- [5] S. Biasotti, D. Giorgio, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *TCS*, 2008.
- [6] 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. Lukaszcyk, D. Sakurai, M. Soler, P. Steneteg, J. Tierny, W. Usher, J. Vidal, and M. Wozniak. An Overview of the Topology Toolkit. In *TopoInVis*, 2019.
- [7] 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.
- [8] R. L. Boyell and H. Ruston. Hybrid techniques for real-time radar simulation. In *Proc. of the IEEE Fall Joint Computer Conference*, 1963.
- [9] P. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A Multi-Resolution Data Structure for 2-Dimensional Morse Functions. In *Proc. of IEEE VIS*, 2003.
- [10] 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 Transactions on Visualization and Computer Graphics*, 2011.
- [11] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Symp. on Dis. Alg.*, 2000.
- [12] H. A. Carr and D. J. Duke. Joint Contour Nets. *IEEE Transactions on Visualization and Computer Graphics*, 2014.
- [13] H. A. Carr, J. Snoeyink, and M. van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *IEEE VIS*, 2004.
- [14] H. A. Carr, G. H. Weber, C. M. Sewell, and J. P. Ahrens. Parallel peak pruning for scalable SMP contour tree computation. In *IEEE Symposium on Large Data Analysis and Visualization*, 2016.
- [15] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *S. o. C. G.*, 2005.
- [16] D. Cohen-Steiner, H. Edelsbrunner, J. Harer, and Y. Mileyko. Lipschitz functions have lp-stable persistence. *Foundations of Computational Mathematics*, 2010.
- [17] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. In *S. o. C. G.*, 2003.
- [18] T. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2009.
- [19] 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.
- [20] H. Doraiswamy and V. Natarajan. Output-Sensitive Construction of Reeb Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 2012.
- [21] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2009.
- [22] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-smale complexes for piecewise linear 3-manifolds. In *S. o. C. G.*, 2003.
- [23] H. Edelsbrunner, J. Harer, and A. K. Patel. Reeb spaces of piecewise linear mappings. In *S. o. C. G.*, 2008.
- [24] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Disc. Compu. Geom.*, 2002.
- [25] H. Edelsbrunner and E. P. Mucke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. on Graph.*, 1990.
- [26] G. Favelier, C. Gueunet, and J. Tierny. Visualizing ensembles of viscous fingers. In *IEEE SciVis Contest*, 2016.
- [27] R. Forman. A User’s Guide to Discrete Morse Theory. *Advances in Mathematics*, 1998.
- [28] T. Gerstner and R. Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proc. of IEEE VIS*, pp. 259–266, 2000.
- [29] B. F. Gregorski, M. A. Duchaineau, P. Lindstrom, V. Pascucci, and K. I. Joy. Interactive view-dependent rendering of large isosurfaces. In *Proc. of IEEE VIS*, pp. 475–482, 2002.
- [30] 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.
- [31] D. Guenther, J. Reininghaus, S. Prohaska, T. Weinkauff, and H.-C. Hege. Efficient computation of a hierarchy of discrete 3d gradient vector fields. In *Proc. of TopoInVis*, pp. 15–29, 2012.
- [32] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Contour forests: Fast multi-threaded augmented contour trees. In *IEEE Symposium on Large Data Analysis and Visualization*, 2016.
- [33] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based Augmented Merge Trees with Fibonacci Heaps. In *IEEE LDAV*, 2017.
- [34] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-Based Augmented Contour Trees with Fibonacci Heaps. *IEEE Trans. Parallel Distrib. Syst.*, 2019.
- [35] 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.
- [36] A. Gyulassy, P. Bremer, R. Grout, H. Kolla, J. Chen, and V. Pascucci. Stability of dissipation elements: A case study in combustion. *Comp. Graph. For.*, 2014.
- [37] 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.
- [38] 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.
- [39] 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.
- [40] 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.
- [41] H. Freudenthal. Simplicialzerlegungen von beschränkter Flachheit. *Annals of Mathematics*, 43:580–582, 1942.
- [42] 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. *Comp. Grap. For.*, 2016.
- [43] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proc. of ACM SIGGRAPH*, 2001.
- [44] H.W. Kuhn. Some combinatorial lemmas in topology. *IBM Journal of Research and Development*, 45:518–524, 1960.
- [45] F. Iuricich and L. D. Floriani. Hierarchical forman triangulation: A multiscale model for scalar field analysis. *Comput. Graph.*, 66:113–123, 2017.
- [46] J. Bey. Tetrahedral grid refinement. *Computing*, 55:355–378, 1995.
- [47] 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.
- [48] P. Klacansky. Open Scientific Visualization Data Sets. <https://klacansky.com/open-scivis-datasets/>, 2020.
- [49] D. E. Laney, P. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2006.
- [50] T. Lewiner, L. Velho, H. Lopes, and V. Mello. Hierarchical isocontours extraction and compression. In *SIBGRAPHI*, pp. 234–241, 2004.
- [51] C. Loop. Smooth Subdivision Surfaces Based on Triangles. Master’s

- thesis, University of Utah, 1987.
- [52] S. Maadasamy, H. Doraiswamy, and V. Natarajan. A hybrid parallel algorithm for computing and tracking level set topology. In *Proc. of HiPC*, 2012.
- [53] 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 *Proc. of IEEE PacificVis*, 2016.
- [54] J. Milnor. *Morse Theory*. Princeton University Press, 1963.
- [55] D. Morozov and G. H. Weber. Distributed contour trees. In *Topological Methods in Data Analysis and Visualization III, Theory, Algorithms, and Applications*. 2014.
- [56] 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.
- [57] S. Parsa. A deterministic $o(m \log m)$ time algorithm for the reeb graph. In *S. o. C. G.*, 2012.
- [58] V. Pascucci and C. L. Bajaj. Time critical isosurface refinement and smoothing. In *Proc. of the Volume Visualization and Graphics Symposium*, pp. 33–42, 2000.
- [59] V. Pascucci and K. Cole-McLaughlin. Parallel Computation of the Topology of Level Sets. *Algorithmica*, 2004.
- [60] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *Proc. IASTED conference on visualization, imaging, and image processing*, 2004.
- [61] 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.
- [62] G. Patanè, M. Spagnuolo, and B. Falcidieno. Reeb graph computation based on a minimal contouring. In *Shape Modeling International*, 2008.
- [63] R.E. Bank, and A.H. Sherman, and A. Weiser. Refinement algorithms and data structures for regular local mesh refinement. *Scientific Computing*, pp. 3–17, 1983.
- [64] G. Reeb. Sur les points singuliers d’une forme de Pfaff complètement intégrable ou d’une fonction numérique. *Comptes Rendus des séances de l’Académie des sciences*, 222(847-849):76, 1946.
- [65] 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.
- [66] S. Zhang. Successive subdivision of tetrahedra and multigrid methods on tetrahedral meshes. *Houston Journal of Mathematics*, 21:541–556, 1995.
- [67] N. Shivashankar and V. Natarajan. Parallel Computation of 3D Morse-Smale Complexes. *Comp. Graph. For.*, 2012.
- [68] 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. <http://vgl.serc.iisc.ernet.in/felix/index.html>.
- [69] J. Singler and B. Konsik. The GNU libstdc++ Parallel Mode: Software Engineering Considerations. In *Proc. of International Workshop on Multicore Software Engineering*, 2008.
- [70] D. Smirnov and D. Morozov. Triplet Merge Trees. In *TopoInVis*, 2017.
- [71] T. Sousbie. The persistent cosmic web and its filamentary structure: Theory and implementations. *Royal Astronomical Society*, 2011. <http://www2.iap.fr/users/sousbie/web/html/indexd41d.html>.
- [72] J. Tierny and H. A. Carr. Jacobi Fiber Surfaces for Bivariate Reeb Space Computation. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2016.
- [73] 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/>.
- [74] J. Tierny, A. Gylassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2009.
- [75] TTK Contributors. TTK Data. <https://github.com/topology-tool-kit/ttk-data/tree/dev>, 2020.
- [76] K. Turner, Y. Mileyko, S. Mukherjee, and J. Harer. Fréchet Means for Distributions of Persistence Diagrams. *Disc. Compu. Geom.*, 2014.
- [77] J. Vidal, J. Budin, and J. Tierny. Progressive wasserstein barycenters of persistence diagrams. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2019.
- [78] J. Vidal, P. Guillou, and J. Tierny. A progressive approach to scalar field topology. *IEEE Transactions on Visualization and Computer Graphics*, 2021. doi: 10.1109/TVCG.2021.3060500
- [79] K. Weiss and L. D. Floriani. Diamond hierarchies of arbitrary dimension. *Comput. Graph. Forum*, 28(5):1289–1300, 2009.
- [80] K. Weiss and L. D. Floriani. Supercubes: A high-level primitive for diamond hierarchies. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 15(6):1603–1610, 2009.