STAG: Smart Tools and Apps in computer Graphics (2016)
Giovanni Pintore and Filippo Stanco (Editors)

Persistent homology: a step-by-step introduction for newcomers

U. Fugacci[1], S. Scaramuccia[2], F. Iuricich[3] and L. De Floriani[2]

[1]Department of Computer Science & UMIACS, University of Maryland, College Park, MD, USA
[2]Department of Computer Science, Bioengineering, Robotics, and Systems Engineering, University of Genova, Genova, Italy
[3]Department of Geographical Sciences, University of Maryland, College Park, MD, USA

Abstract

Persistent homology is a powerful notion rooted in topological data analysis which allows for retrieving the essential topological features of an object. The attention on persistent homology is constantly growing in a large number of application domains, such as biology and chemistry, astrophysics, automatic classification of images, sensor and social network analysis. Thus, an increasing number of researchers is now approaching to persistent homology as a tool to be used in their research activity. At the same time, the literature lacks of tools for introducing beginners to this topic, especially if they do not have a strong mathematical background in algebraic topology. We propose here two complementary tools which meet this requirement. The first one is a web-based user-guide equipped with interactive examples to facilitate the comprehension of the notions at the basis of persistent homology. The second one is an interactive tool, with a specific focus on shape analysis, developed for studying persistence pairs by visualizing them directly on the input complex.

Keywords: Persistent homology, Topological data analysis, Shape analysis.

Categories and Subject Descriptors (according to ACM CCS): K.3.2 [Computer and Education]: Computer and Information Science Education—Computer science education

1. Introduction

A data scientist has become a relevant figure in the process of elaborating and analyzing data. In the general field of data science, Topological Data Analysis (TDA) gathered a lot of attention in the last few years as a complementary framework to more classical machine learning techniques.

Persistent homology is one of the foundational tools for TDA. The relevance of persistence homology in several application domains already led to few books and surveys describing its theoretical aspects in detail. Worth to be mentioned are the survey and the book by H. Edelsbrunner and J. Harer [EH08, EH10] and the books by A. Zomorodian [Zom05] and by R. Ghrist [Ghr14]. In the two works by H. Edelsbrunner and J. Harer, the authors propose an introduction to persistent homology, and, more generally, to computational topology, that can be useful for students of computer science or mathematics with a substantial background in topology and algorithms. Similarly, in [Zom05], the main task is to show the significance and utility of topological concepts for solving problems in computer science. In [Ghr14], the author gives a personal overview of some basics notions of applied topology. Persistent homology is viewed from a representation-theoretic aspect and it is presented as a combination of homology and sequences. In the survey by N. Otter et al., the pipeline for the computation of persistent homology is overviewed and a special emphasis is given to the available software tools [OPT*15].

Even though these are the most valuable resources for approaching the world of homology and persistent homology, the formalism required for defining the basic concepts in an algebraic framework may push away readers lacking such a mathematical background. Few examples exist that aim at introducing these concepts in an intuitive way.

The short paper by Weinberger [Wei11] gives a brief and intuitive description of persistent homology presenting it as a way to handle discrete datasets through mathematical tools. Recently, a video giving a brief and intuitive description of the basics notions behind persistent homology has been proposed and it is available on YouTube (`https://www.youtube.com/watch?v=2PSqWBIrn90&feature=youtu.be`) [Wri16].

The aim of our work is to provide three different contributions for a person moving his/her first steps into persistent homology. As the main resource, we provide an interactive website including all the basic notions and the description of the standard algorithm for computing persistent homology. The formal definitions are matched with intuitive descriptions and interactive examples to get confidence with the basic concepts. A full description of the website is provided in Section 4.

DOI: 10.2312/stag.20161358

The second tool is a visualization framework for triangulated surfaces. Provided an input triangulation and a filtration the program computes the resulting persistent homology. The persistence pairs obtained are visualized both in an interactive persistence diagram (where each pair is represented as a point in the graph) and in a 3D scene (where each pair is depicted as a pair of simplices of the surface). By interactively playing with different filtration/triangulations and filtering out persistence pairs from the graph, a beginner can study the effect of using different input filtrations and the relations between each persistence pair and the original data. This tool is described in Section 5.

The third contribution is included in this paper. In Section 3, we describe the history of persistent homology reviewing the preliminary works that brought to the definition of the theory as it is used nowadays. Moreover, we provide an overview of the state-of-the-art methods. This last contribution is intended for students and researchers that gained some confidence on the basics and want to study in deep the argument.

2. Background notions

In this section, we introduce the background notions at the basis of our work, namely simplicial complexes, simplicial and persistent homology.

2.1. Simplicial complexes

A classical way to represent discretized objects is through simplicial complexes, a collection of well-glued bricks called simplices. Formally, a *k-simplex* σ is the convex hull of $k+1$ affinely independent points. A 0-simplex is a single point, a 1-simplex an edge, a 2-simplex a triangle, a 3-simplex a tetrahedron, and so on. Given a k-simplex σ, the *dimension of* σ is defined to be k and denoted as $dim(\sigma)$. Any simplex which is the convex hull of a non-empty subset of the points generating σ is called a *face* of σ. A *simplicial complex* Σ is a finite set of simplices which satisfies the gluing conditions requiring that each face of a simplex in Σ belongs to Σ, and each non-empty intersection of any two simplices in Σ is a face of both. We define the *dimension* of a simplicial complex Σ, denoted as $dim(\Sigma)$, as the largest dimension of its simplices.

2.2. Homology and persistent homology

Simplicial homology is a powerful tool in shape analysis, providing invariants for shape description and characterization.
Given a simplicial complex Σ, it is possible to define the *chain complex* associated with Σ, denoted as $C_*(\Sigma) := (C_k(\Sigma), \partial_k)_{k\in\mathbb{Z}}$, where $C_k(\Sigma)$ is the free Abelian group generated by the k-simplices of Σ, and $\partial_k : C_k(\Sigma) \to C_{k-1}(\Sigma)$ is a homomorphism, called *boundary map*, which encodes the boundary relations between the k-simplices and the $(k-1)$-simplices of Σ such that $\partial^2 = 0$. We denote as $Z_k(\Sigma) := \ker(\partial_k)$ the group of the k-cycles of Σ and as $B_k(\Sigma) := Im(\partial_{k+1})$ the group of the k-boundaries of Σ. Then, we define the k^{th} *homology group of* Σ as

$$H_k(\Sigma) := H_k(C_*(\Sigma)) = \frac{Z_k(\Sigma)}{B_k(\Sigma)}$$

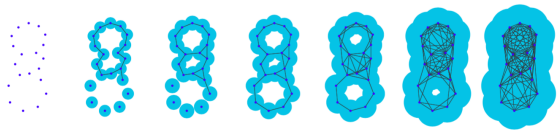


Figure 1: *A filtration of a simplicial complex Σ (image from [Bub15]). The filtration is obtained starting from a set of points and increasing the radius of the balls centered in each point. A k-simplex generated by vertices $v_0,\dots,v_k$ is created when every two balls centered in $v_0,\dots,v_k$ have a non-null intersection.*

Roughly speaking, homology groups reveal the presence of "holes" in a shape. For each degree k, the k^{th} Betti number β_k is defined as the rank of $H_k(\Sigma)$ and it counts the number of independent k-cycles which do not represent the boundary of any collection of simplices of Σ. In dimension 0, β_0 coincides with the number of connected components of the complex, in dimension 1, its tunnels and its holes, in dimension 2, the shells surrounding voids or cavities, and so on.

Persistent homology [EH10] is an important tool in topological shape analysis, which aims at overcoming intrinsic limitations of classical homology by allowing for a multi-scale approach to shape description. In a nutshell, persistent homology describes the changes in homology that occur to an object which evolves with respect to a parameter. Given a simplicial complex Σ, a *filtration* of Σ is a finite sequence of subcomplexes $\Sigma^f := \{\Sigma^p \,|\, 0 \le p \le m\}$ of Σ such that $\emptyset = \Sigma^0 \subseteq \Sigma^1 \subseteq \cdots \subseteq \Sigma^m = \Sigma$.
For $p,q \in \{0,\dots m\}$ such that $p \le q$, the *(p,q)-persistent k-homology group* $H_k^{p,q}(\Sigma)$ of Σ consists of the k-cycles included from $C_k(\Sigma^p)$ into $C_k(\Sigma^q)$ modulo boundaries. Formally, it can be defined as

$$H_k^{p,q}(\Sigma^f) := Im(i_k^{p,q})$$

where $i_k^{p,q}$ denotes the linear map between $H_k(\Sigma^p)$ and $H_k(\Sigma^q)$ induced by the inclusion of complexes between Σ^p and Σ^q.
As already mentioned, persistent homology provides more information about a shape than classical homology. While homology captures cycles in a shape by factoring out the boundary cycles, persistent homology allows for the retrieval of cycles that are non-boundary elements in a certain step of the filtration and that will turn into boundaries in some subsequent step. The persistence of a cycle during the filtration gives quantitative information about the relevance of the cycle itself for the shape.

In Figure 1, an example of filtration for a simplicial complex Σ is shown. Persistent homology allows for detecting the changes in the homology of Σ. For instance, the birth and the death of 1-cycles occurring during the filtration of Σ.

Both classical and persistent homology can be defined in terms of any Abelian group as coefficient group. The most complete homological information is retrieved by choosing $\mathbb{Z}$ as coefficient group. In spite of this, for simplicial complexes embeddable in $\mathbb{R}^3$, the information obtained by considering coefficients in $\mathbb{Z}$ or in $\mathbb{Z}_2$ is the same (see [AH35], Chapter *X*). So, in practical cases, classi-

cal and persistent homology is usually considered with coefficients in $\mathbb{Z}_2$.

3. A short history of persistent homology

In this section, we give a short overview of the evolution of persistent homology considering the techniques used to visualize the information recoverable thanks to persistent homology as well as the approaches for computing it.

Defining persistent homology. The concept of persistence sprang independently from different perspectives between the 90's and the beginning of the new millennium. The roots of persistence are to be found in the context of *size theory* [Fro90]. Size theory aims at detecting (dis)similarities between shapes. Shapes are represented by pairs including a topological space together with a continuous real-valued function defined on it. In 1992, P. Frosini developed the notion of *size function* [Fro92]. According to the continuous function values, the topological space is filtered by sub-level sets. The size function counts how many connected components are maintained from one sub-level set to another. As the reader may notice, this is precisely the core idea in persistent homology, though limited at that time to the analysis of the degree 0 of homology.

In 1999, exploratory works on persistent homology (not yet defined with this name) pointed towards the area of shape recognition. V. Robins [Rob99] applied α-shape approximations to the study of homological features of fractal sets. Starting from a sampled fractal set, an increasing sequence of discrete spaces called α-shape was built to approximate the original shape. The focus was on analyzing the history of births of homological features in the final object. The *persistent Betti numbers* were introduced to count, for any intermediate complex, how many homological features contribute to the homology of the final object.

While the latter work contained the essence of persistent homology, the well-established definition associated with a general increasing sequence of simplicial complexes and defined for any ordered pair of filtration parameters p and q is due to H. Edelsbrunner, D. Letcher, and A. Zomorodian [ELZ02]. At the beginning, the definition was limited to complexes embedded in the 3-dimensional Euclidean space. Afterwards, it reveals itself as valid in a more general context. The notion of persistent Betti numbers took the current form to indicate, in general for any two ordered parameters $p \leq q$, the number of homology classes survived from p to q.

Visualizing persistent homology. Due to its relevance and effectiveness in topological data analysis, persistence homology has been the object of in-depth studies and investigations. Fundamental contributions in the area focus on methods for effectively interpreting and visualizing the topological features detected by the persistent homology.
The notion of *persistence pair* has met this need. Intuitively, each persistence pair represents a non-null contribution in persistent homology. Given a filtration $\{\Sigma^p \,|\, 0 \leq p \leq m\}$ of a simplicial complex Σ, a persistence pair (p,q) is represented as an element in $\{0,\ldots,m\} \times (\{0,\ldots,m\} \cup \infty)$ such that $p < q$. Specifically, if both p and q are natural numbers, the pair (p,q) represents a non-trivial homology class that is born at step p of the filtration and dies at step q. In this case, the value $q-p$ corresponds to the lifespan of that homology class and allows for discriminating between relevant and negligible classes. Differently, a pair (p,∞) reflects the existence of a non-trivial homology class that is born at step p and persists along all the following filtration steps.
Due to the intuitive meaning of persistence pairs, much research has focused on their properties and visualization. A crucial step has been made by G. Carlsson and A. Zomorodian [CZ05]. They investigated the algebraic nature of persistent homology by introducing the *persistence module* and proved a classification result. Thanks to the structure theorem for graded modules over PID (i.e., a principal ideal domain), they proved that the knowledge of all persistence pairs of a filtration completely characterizes the persistence module. So, filtrations having different persistence pairs necessarily have non-isomorphic persistence modules.

As far as visualization is concerned, the literature has indicated many possibilities along the years which find preferences according to the applicative purposes. The notion of persistence pair has a counterpart in size theory. Any size function, which corresponds to a persistent Betti number, is completely described by a discrete number of its values called *corner points*, for $q < \infty$, and *corner lines*, for essential classes [FL01].
At the time in which persistent homology has been introduced [ELZ02], authors visualized persistence pairs either as half-open intervals $[p,q)$ on the real line or triangles in the index-persistence plane spanned by $(p,0)$, $(q,0)$ and $(p,q-p)$, where the index is a total ordering on the simplices in the domain and persistence is the space of the differences $q-p$. By intervals on the real line, it is very intuitive to compare lifetimes of the classes. By triangles, it is more intuitive to relate persistent Betti numbers to persistence pairs. Indeed, for any pair in the persistence-index plane, the persistent Betti number equals the number of triangles containing the pair.
These two main ideas have led to the most common equivalent ways to visualize persistence pairs. The interval approach has led to the current notion of *barcode*, widely surveyed in [Ghr08] and first called by this name in [CZCG05]. Properly, a barcode is a graphical representation of the persistent homology of a filtered simplicial complex as a collection of horizontal line segments obtained by considering each persistence pair (p,q) as an interval.
The triangle approach and the size function representation by corner points has led to the most adopted visualization of persistence pairs called *persistence diagrams* [EH08]. The persistence diagram of a given dimension k is just the representation of points in the Cartesian plane of the persistence pairs (p,q) representing a homology class of dimension k. In such a representation, all the homology classes lie above the diagonal of the first quadrant and the lifespan of a class (p,q) is measured in terms of the distance of the point (p,q) from the diagonal in the ∞-norm. Differently, persistent homology classes (p,∞) are represented in the upper part of the first quadrant. Each persistence pair indicates either a triangle, spanned by (p,q), (p,p) and (q,q) if $q < \infty$, or an open area, limited by the vertical line in $(p,0)$ and the diagonal. For any other point in the plane, its corresponding persistent Betti number is the number of triangles or open areas including it.
Recently, a novel representation for the visualization of persistence

pairs, called *persistence landscape*, has been introduced [Bub15]. A persistence landscape is a representation halfway between the persistence diagram and the barcode. It can be roughly considered as a horizontal version of the persistence diagram and it has been studied for better combining topological data analysis with statistics and machine learning. Figure 2 displays the three representations here described for the visualization of the persistence pairs.

The relevance of persistence pairs as descriptors is confirmed by *stability* results with respect to two main notions of pseudo-distances between persistence diagrams. The first one is the *matching distance*. It has been first introduced in [LF97] within the context of size functions, then defined for persistent diagrams in [CSEH07] under the name of *bottleneck distance* to underline the relation with a classical graph-theory notion of distance. A matching is a bijective correspondence between pairs in two diagrams. A pair can either be matched to another pair or to a diagonal point in order to allow for comparisons of diagrams of different cardinality. Any two matched pairs imply a cost according to their distance in the ∞-norm. The general cost of the matching is the maximum cost of any two pairs. The infimum of all costs as the matching varies among all possible ones is taken as the matching distance between the two diagrams. In size theory, we have a continuous function defined on the shape. The matching distance is stable with respect to perturbations of the original function with respect to the l_∞-norm [DFL10].
In [CSEH07], authors proved a crucial stability result for the bottleneck distance under small perturbations of the filtration assuming the filtering function to be tame, which includes Morse functions and PL functions.
In the same paper, as a second possibility to compare persistence diagrams, the *Hausdorff distance* is considered and proved to be stable. Informally, two sets are close in the Hausdorff distance if every point of either set is close to some point of the other set. In other words, the Hausdorff distance is the greatest of all the distances from a point in one set to the closest point in the other set. The Hausdorff distance provides a lower bound for the matching/bottleneck distance. This latter is currently the most used, though the former is easier to compute. Stability of persistence diagrams has also an algebraic counterpart due to Chazal et al. [CCSG*09], where the bottleneck distance is proven to be stable with respect to algebraic perturbations of the persistence module called *interleavings*, without any explicit reference to any filtering function.

Computing persistent homology. Together with the notion of persistent homology also the computation of the homological information has evolved over the years. At the beginning, persistent homology has been computed by extending techniques developed for retrieving homology. Nowadays, the spread of persistent homology has led to a dual situation. The classical way to retrieve the homology with coefficients in a PID of a simplicial complex Σ is based on a matrix reduction, the *Smith Normal Form (SNF) reduction*, applied to the matrices encoding the boundary maps of Σ [Mun84, Ago05]. Similarly, persistent homology with coefficients in a PID of a filtered simplicial complex can be retrieved by using a matrix-based reduction [ZC05]. The specialization to coefficients in a field of this algorithm is usually denoted as *standard algorithm* and, in practical cases, it has a linear complexity in the number of the simplices of the complex. Although the methods based on a matrix reduction are theoretically valid in any dimension, their worst-case complexity is super-cubical in the number of simplices of the complex.

Improvements have been proposed since then. They can be subdivided based on the strategy they adopt, namely:

- direct optimizations [MMS11, CK13, CK11, DMV11, BM14],
- distributed approaches [BCA*11, LZ14, LSV11, MNV13, BKR14a, BKR14b],
- methods based on annotations [DFW14, BDM13].

We refer as *direct optimizations* all those methods that improve the efficiency of the standard algorithm. Differently, *distributed approaches* efficiently retrieve the homological information of a complex through parallel and distributed computations. Recently, persistent homology is efficiently computed using *annotations* which are vectors associated with each simplex of the complex encoding in a compact way the homology class of the simplex itself.
A different and fundamental subfield of methods studied for persistent homology are the so-called, *coarsening or pruning approaches* [MB09, MW10, BDMZ12, DHKS13, Zom10, RWS11, HMMN14, FID14, DW14]. These methods reduce the size of the input complex without changing its homological information by applying iterative simplifications.

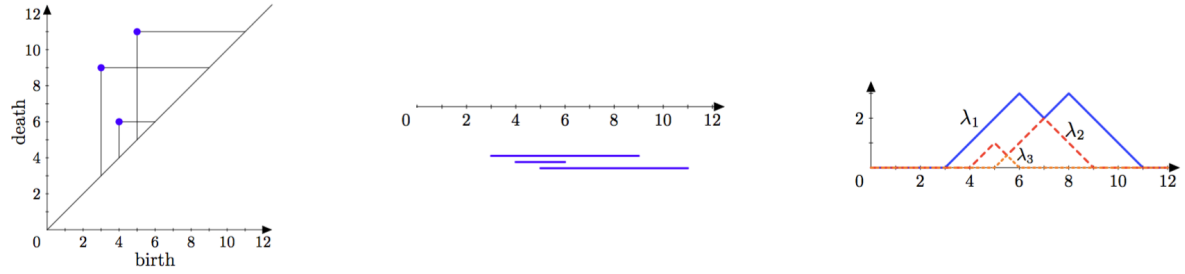
4. Interactive user-guide to persistent homology

In order to meet the increasing attention on persistent homology by a wider and wider class of researchers, we have created a web-based user-guide on the same topic. The main contributions of this tool are:

- an intuitive and self-contained introduction to persistent homology;
- an introduction to the standard algorithm for computing persistent homology;
- an overview of the persistent homology state of the art to orient beginners.

For each of these contributions, our work addresses a twofold task. On the one hand, we present an intuitive introductive guide enriched with interactive examples in order to be comprehensible for a large class of users. On the other hand, the theoretical consistency of the introduced notions and algorithms is still preserved. Moreover, mulitple references and links in the text redirect to relevant contents and allow to create a complete guide on persistent homology from its elementary definitions to its computation and applications. According to this double task, the proposed guide reveals to be a useful tool for different kinds of users. It can be a powerful tool for courses on the topic, a step-by-step handbook for beginners and a solid starting point for researchers coming from different research fields.

For purposes of clarity, the web page alternates white to colored backgrounds. White corresponds to the descriptive notions, including formal definitions. Blue colored boxes correspond to



(a) (b) (c)

Figure 2: *Various visualizations of the persistence pairs in degree 1 of the filtration depicted in Figure 1: persistence diagram (a), barcode (b) and persistence landscape (c) (images from [Bub15]).*

interactive examples and useful comments. Red colored boxes indicate involved notions or links to detailed studies. The idea is to make the exposure lighter in the graphics and to combine the mathematical formalism to the graphical intuition. The same complex is carried along all the examples. We have chosen a simplicial complex (see Figure 3(a)) combining both a small size (crucial for the step-by-step presentation of the standard algorithm) and interesting persistence features (such as essential classes and non-trivial 0 and 1-degree homologies). This choice is supposed to help the reader in relating the different definitions with minimum effort. Moreover, the interactivity allows the reader to briefly focus on the examples without long breaks in the reading and it helps in optimizing the web page space required by the examples. This particularly impacts the step-by-step reduction of the 23x23 matrix in the algorithm part, otherwise too heavy (see Figure 3(c)).

Introducing background notions for persistent homology. The core feature of our guide is to be self-contained and minimal with respect to the formal definitions introduced. We made the explicit choice of simplifying the exposition for four aspects.

(1) We assume our simplicial complex to be finite and embedded in an Euclidean space. Thus, there is a precise choice not to define the simplicial complex in an abstract way. This makes it clear from the beginning the difference between the notion of (intrinsic) dimension d of a complex and that of the embedding space n. Newcomers interested in relating standard filtrations in the literature such as the α-shapes and the Vietoris-Rips complexes may take advantage of this approach. In addition, our choice is to denote the faces of dimension $k-1$ of a k-simplex σ simply by $\hat{\sigma}_i$ to indicate directly the removal of vertex of index i. This induces differences, that we believe can help the non-expert reader, with respect to the usual notation in the definition of the boundary maps at the chain complex level and in the definition of a filtering function.
(2) The formal definition of a chain complex restricted to coefficients over the field $\mathbb{Z}_2$ avoids the need of orienting the simplices and considering complicated chains. The same restriction allows also for introducing an algebraic notation while preserving a familiar domain for computer scientists. The notion of chain is a crucial one for the introduction of homology and many examples are provided to visualize directly cycles and boundaries. With respect to the simplicial homology, the choice of considering $\mathbb{Z}_2$ coefficients is not restrictive as long as we consider simplicial complexes embedded in a 3-dimensional Euclidean space. Moreover, it allows for introducing the homology as a vector space, instead of the most general definition of homology as an Abelian group. Before the actual definition, simplicial homology is motivated by means of the informal notion of k-hole, which is supposed to help understanding the interpretation of the Betti numbers.

(3) A single definition of persistence homology is mentioned. Being the most well-established definition of persistent homology in the scientific community, we define it with respect to any pair of ordered parameters p and q as the image of the respective induced linear map of homologies. No other equivalent or related definitions, such as the equivalent one by Edelsbrunner et al. in [ELZ02] and that of persistence module introduced in [CZ05], are mentioned. We first introduce the notion of filtering function and we induce the filtration afterwards. Our motivation is twofold. On the one hand, this is the standard setting when working with shape descriptors. On the other hand, this approach is general since any finite sequence of nested simplicial complexes might come from such a filtering function. In particular, we stress the importance of each sub-level set being a subcomplex, which might be tricky in the first approach to persistence. Other metric-based constructions crucial for applications to point cloud data coming from the topological data analysis context are mentioned as variations of that approach in the final part.

(4) Only the persistence diagram is presented as the persistent homology descriptor. The barcode is mentioned for completeness with suitable references together with the landscapes descriptor. Moreover, in order to provide a definition for the persistence diagram untangled from the structure theorem, we introduce persistent classes h explaining how to associate a birth parameter p_h and a death parameter q_h.

Describing the standard algorithm. A main part of the web-based guide is devoted to the description of the *standard algorithm*. Among all the possible algorithms proposed for computing persistent homology, we choose the latter by considering $\mathbb{Z}_2$ as the

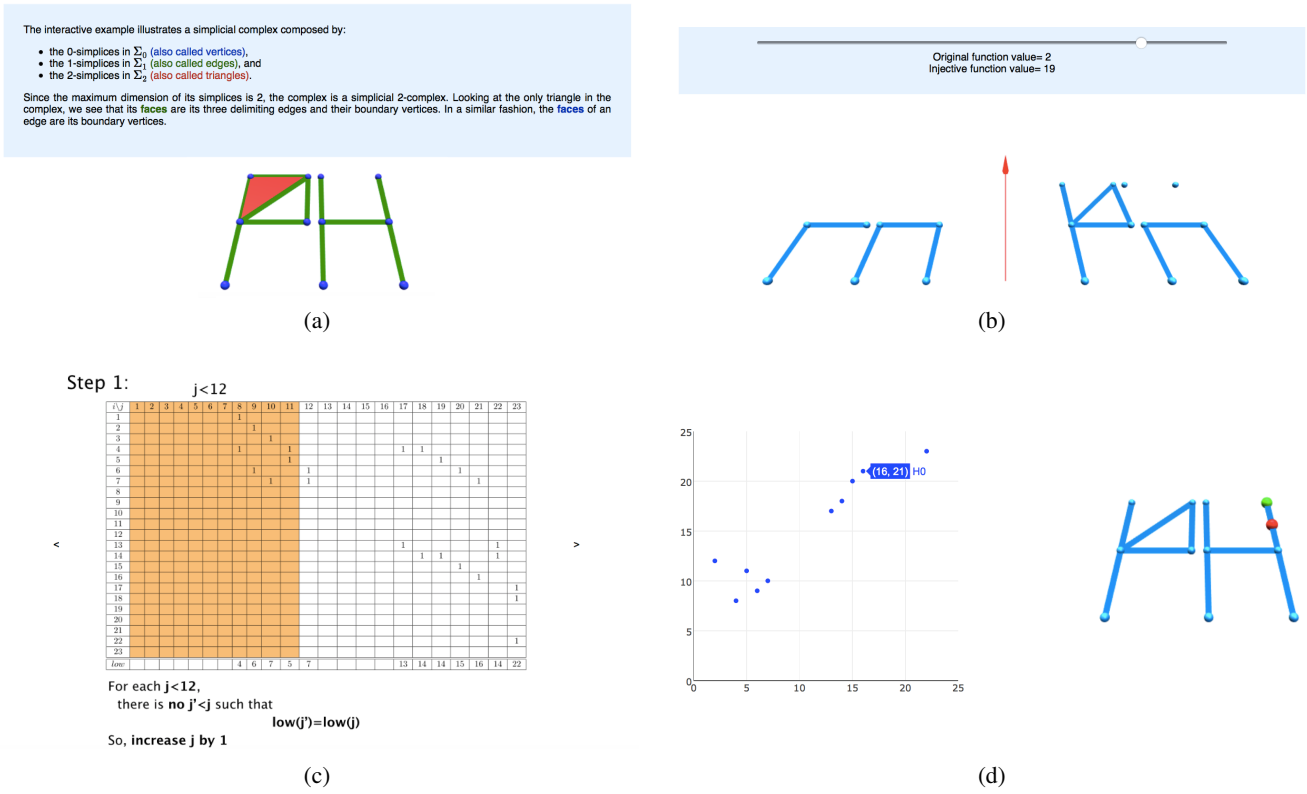


Figure 3: *Few interactive examples coming with the web-based guide. While reading, the user can trigger events to get a visual representation of the content of the text. The events can be triggered moving the cursor over text (a), using a slider (b), clicking on images (c) or interacting with the figures (d).*

coefficient ring. Our choice is due to various reasons. First of all, the standard algorithm is considered in the literature as the reference algorithm for the computation of persistent homology. Further, the choice of considering $\mathbb{Z}_2$ as the coefficient group allows for simplifying the description of the algorithm and enhancing the understanding of its steps. Finally, we describe the version of the standard algorithm proposed in [EH10] rather than the more general one introduced in [ZC05] since the use of polynomials can be too tough for non-expert users.

Our presentation describes the entire pipeline for computing persistent homology, from the input filtered complex, through the step-by-step analysis of the algorithm, to the representation and the interpretation of the obtained persistence pairs.

In the section devoted to the construction of the boundary matrix M associated with a filtered simplicial complex Σ^f, the user has the possibility of better understanding the relationship between the original filtering function f and the indexing of the matrix M. On the website, we explain how such indexing is equivalent to an injective function f' defined on all the simplices of Σ. With a slider, we allow for updating the value of the functions f and f' displaying the corresponding simplicial complexes changing accordingly (see Figure 3(b)).

The standard algorithm is described showing both a pseudo-code description and a step-by-step example (see Figure 3(c)). The example provided allows a user to iterate along the various steps of the algorithm applied to the filtered simplicial complex.

Another crucial point discussed in this paper is the interpretation and visualization of the results. Discriminating between the persistence pairs of Σ with respect to f and to f' is a step often overlooked in the literature. For a better understanding, our guide treats each of these cases separately. After having described how the persistence pairs can be quickly retrieved by displaying the pseudocode of the corresponding algorithm, we define the notion of positive ad negative simplices and we give an interactive way to visualize them. In the proposed examples, one for each filtering function f and f', the user can study the information provided by each persistence pair projected directly on the original simplicial complex. Each example depicts, on the left, the graph with a blue point for each simplex pair leading to a non-trivial persistence pair. Selecting a point on the graph, a simplicial complex is rendered at the filtration step in which the persistence pair is created. The two simplices involved in the pair are indicated with a green sphere for the simplex that created the homology class, and a red sphere, for

the one that killed it (see Figure 3(d)).

By looking at the distribution of the points on the graph, one can get a visual representation of another fundamental information encoded in the persistence pairs, the lifespan. The more a point is depicted far away from the diagonal, the longer the homology class it represents lived in the filtration. In the example, we are also showing the homology classes that never die, depicted with cubes. In the case of the filtering function f', the two blue cubes represent the two connected components that survive at the end of the filtration while the green cube corresponds to the 1-cycle.
The proposed graphical visualization of the information returned by the computation of persistent homology does not properly coincide with persistence diagrams. In fact, for the sake of simplicity, we choose to adopt a slightly different representation that can be much more readable than the persistence diagrams while encoding the same information. First, the proposed representation includes a single scatterplot graph the entire information given by the persistence diagrams of all the homology degrees. Furthermore, differently from the persistence diagram, the introduced visualization represents persistence pairs with components given by f' rather than f, thus allowing to represent as distinct points persistence pairs with multiplicity greater than 1.

After the first steps. The last page of our guide wants to provide material for a reader interested in theoretical or practical advanced details. A first discussion concerns the different types of datasets on which persistent homology can be computed and their connections to specific classes of application domains. Depending from the context, the input data can be discretized through a scalar function defined on a set of points, a point cloud, or a simplicial complex with a filtration value associated with each of its simplices. Especially for non-expert users, this variety can cause misunderstandings making him/her think of being in front of different tools rather than of a unique powerful theory applied to several contexts. For each of the recognized classes, the guide provides a short description and links to relevant dataset repositories also addressing the user to the corresponding application fields.
In order to provide a global point of view on persistent homology computation, the last part of the guide is devoted to the algorithmic approaches for computing persistent homology different from the standard algorithm. The guide provides a classification of such approaches according to the adopted strategies giving, for each of them, the suitable references. Specifically, we can subdivide these methods into direct optimizations of the standard algorithm, distributed approaches, methods based on annotations, and coarsening and pruning approaches. Finally, we collect and classify the software tools for computing persistence homology that have been distributed in the public domain.

5. A Web-GL interface for analyzing persistence pairs

In this section, we describe the tool developed for studying and visualizing persistence pairs on a triangulated surface.

The tool is composed of two distinct packages. The first one takes a filtration as input (described as triangulation Σ and a function defined on the vertices of Σ) and it computes the persistence pairs a format suitable for the second package, the visualization interface.

Algorithm 1 computeBoundaryMatrix(Σ, f)

Input: Σ simplicial complex
Input: $f : \Sigma \longrightarrow \mathbb{R}$ filtering function
Output: M boundary matrix

```
1:  M.setSize(|Σ|)
2:  f' = computeIndexing(Σ, f)
3:  Σ^f' = sortSimplices(Σ,f')
4:  for all σ ∈ Σ^f' do
5:    if bd(σ) == ∅ then
6:      M.pushCol();
7:    else
8:      B = ∅;
9:      for all τ ∈ bd(σ) do
10:       B.push(f'(τ));
11:     M.pushCol(B);
12: return M
```

5.1. Computing persistent homology

The input expected from the software tool is a triangulated surface (currently only .ply and .off formats are supported). The filtering function, provided on a separate file, is defined on the vertices of the complex and extended to all the simplices. Let $f : \Sigma_0 \longrightarrow \mathbb{R}$ be the filtering function defined on the vertices Σ_0 of Σ, f is extended to all the simplices $\sigma \in \Sigma$ as $f(\sigma) = max_{v \in \sigma} f(v)$.

Once the filtering function has been extended, the matrix M representing the boundary relations among the simplices of Σ is computed using Algorithm 1. Starting from the input filtering function, an indexing f' is defined on the simplices of Σ (row 2). The index i of a simplex represents the position of its corresponding column in M. The index values grow with the filtration so, if $f(\sigma) < f(\tau)$ or σ is a proper face of τ, then $f'(\sigma) < f'(\tau)$. Ties are solved in such a way that each simplex has a lower index of the simplices on its boundary. Once f' has been computed, the simplices are processed in order according to it. For each k-simplex σ we extract its immediate boundary (i.e., its faces of dimension $k-1$) using the function bd. If σ is a vertex then we insert an empty column in M (row 11). Otherwise, we initialize a column B by inserting the values of f' for each simplex on the boundary $bd(\sigma)$.

The next step is computing persistence homology. In our implementation, we are using the standard algorithm described in [ELZ02] provided in the PHAT library [BKR13]. Once persistent homology has been computed, the persistence pairs are extracted using a built-in function still provided in the PHAT library. The output files describe the persistence pairs computed in a suitable format to be read by the visualization tool. For clarity, we postpone the description of these files to the next section.

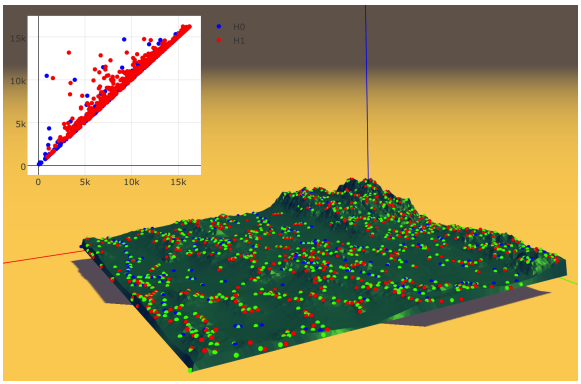


Figure 4: *Main interface of the visualization tool.*

5.2. Visualizing persistent pairs.

Figure 4 shows the main view offered to the user. The 3D scene is implemented using the THREEJS library [3js], a Javascript library based on WebGL. THREEJS is used here to create the 3D scene for the triangulated surface. Persistence pairs are here depicted as spheres colored according to the simplex dimension (vertices-blue, edges-green, triangles-red). Each sphere is drawn at the barycenter of the corresponding simplex. Interactions with the 3D scene are limited to an orbiting camera allowing us to change the point of view.

The scatter plot used for representing the persistence pairs is implemented using the Plotly library [plo15]. Plotly is a high-level charting library built on top of d3js. It provides a series of interactive charts and an API for different programming languages including Python, Javascript and R.

Using Plotly, we can easily develop interactive charts. Operations that can be performed on the graph vary from zooming or panning to selecting a subset of data. The programmer is only responsible for setting up the desired event listener for each interaction.

Our tool takes two files as input. The first one is the file containing the triangulation (temporarily, only the ASCII .ply format is supported). The second one is a .json file listing all the persistence pairs computed by the algorithm described in Subsection 5.1. Pairs are organized based on the homology type that is killed from each of them. We recall that a vertex-edge pair represents the "destruction" of a 0-cycle while edge-triangle pairs represent 1-cycles filled in during the filtration.

Each pair is then associated with eight values representing:

- the coordinates of the vertex in the scatterplot (2 values),
- the coordinates in the 3D scene of the barycenter of the first simplex (3 values),
- the coordinates in the 3D scene of the barycenter of the second simplex (3 values).

The scatter plot represents the dashboard for interacting with the visualization tool.

Likewise in the scatterplot shown in the user-guide, the graph showing the persistence pairs is not a proper persistence diagram. Here vertex-edge and edge-triangle pairs are depicted together. The former are depicted as blue dots while the seconds as red dots. Also, each point is depicted at coordinates (i, j) where i and j are the indices in the boundary matrix M of the two simplices involved. In other words, the coordinates are maintained accordingly to the injective function f'. This way, each point on the graph always corresponds to a single pair of simplices (i.e., the graph is no longer a multiset) and the user's interactions result more intuitive.

We recognize two types of interactions called, *scene oblivious* and *scene modifying*. Scene oblivious interactions provoke updates on the scatter plot only. Through such operations, we can zoom in and zoom out on the scatter plot or pan the portion of the graph rendered.

Scene modifying interactions change the number of critical pairs visualized in the 3D view and are the main tool for the interactive analysis. Two different tools (box or lazo selection tools) can be used for selecting a subset of points from the graph. Pairs excluded from the selection are also removed from the 3D view of the scene. An example of the two selection tools is shown in Figure 5. Moving the mouse cursor over a point in the scatterplot will automatically remove all the persistence pairs from the 3D view but the selected pair. See Figure 6 for an illustrative example.

6. Concluding remarks

In this paper, we have presented a new approach for spreading persistent homology as a practical tool. The interactive guide can be found https://github.com/IuricichF/ICT. The source code of the visualization tool can be found https://github.com/IuricichF/VisualizePH.

We are planning to expand the web-based user-guide including many other concepts rooted in computational topology such as Morse theory, Reeb Graphs, discrete Morse theory by Forman and so on. Our long-term goal is to realize a shared framework where researchers can participate in building user-friendly interactive guides to be used for courses.

In this direction, also the visualization tool could become extremely powerful. We are planning to extend the number of structures with other representations commonly used in computational topology. We think that showing the relations between the many topological structures (such as Morse cells, Reeb graphs, persistence pairs etc.) with such an interactive tool would enhance the learning process of a beginner substantially.

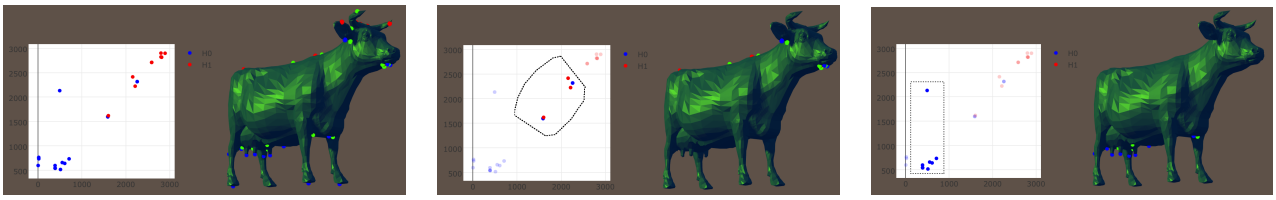
References

[3js] Threejs: Javascript 3d library. Accessed: 05-08-2016. URL: http://threejs.org. 8

[Ago05] AGOSTON M. K.: *Computer Graphics and Geometric Modeling*. Springer Verlag London, 2005. doi:10.1007/b138805. 4

[AH35] ALEXANDROFF P., HOPF H.: *Topologie I.* Springer-Verlag Berlin Heidelberg, 1935. doi:10.1007/978-3-662-02021-0. 2

[BCA*11] BOLTCHEVA D., CANINO D., ACEITUNO S. M., LÉON J.-C., DE FLORIANI L., HÉTROY F.: An Iterative Algorithm for Homology Computation on Simplicial Shapes. *Computer-Aided Design 43*, 11 (2011), 1457 – 1467. doi:10.1016/j.cad.2011.08.015. 4



(a) (b) (c)

Figure 5: *The main interface (a) of the visualization tool. "Lazo" tool (b) and "box" tool (c) for selecting a subset of the persistence pairs to be visualized on the mesh.*

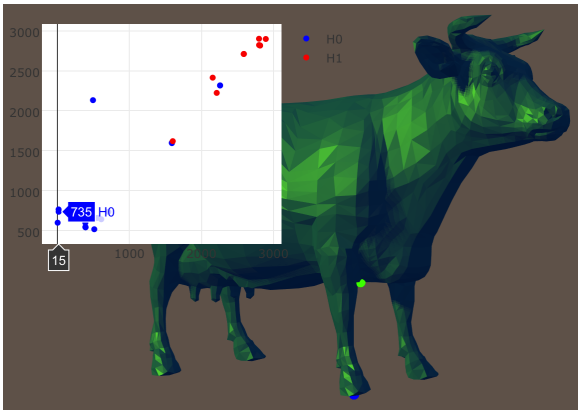


Figure 6: *Selecting a single point from the scatter plot*

[BDM13] BOISSONNAT J.-D., DEY T. K., MARIA C.: The Compressed Annotation Matrix: An Efficient Data Structure for Computing Persistent Cohomology. In *Algorithms–ESA 2013: 21st Annual European Symposium*, vol. 8125 of *Lecture Notes in Computer Science*. 2013, pp. 695–706. doi:10.1007/978-3-642-40450-4_59. 4

[BDMZ12] BRENDEL P., DŁOTKO P., MROZEK M., ŻELAZNA N.: Homology computations via acyclic subspace. In *Proceedings of the 4th international conference on Computational Topology in Image Context* (Berlin, Heidelberg, 2012), CTIC'12, Springer Verlag, pp. 117–127. doi:10.1007/978-3-642-30238-1_13. 4

[BKR13] BAUER U., KERBER M., REININGHAUS J.: Phat: Persistent homology algorithm toolbox. http://bitbucket.org/phat-code/phat, 2013. 7

[BKR14a] BAUER U., KERBER M., REININGHAUS J.: Clear and Compress: Computing Persistent Homology in Chunks. In *Topological Methods in Data Analysis and Visualization III*, Mathematics and Visualization. 2014, pp. 103–117. doi:10.1007/978-3-319-04099-8_7. 4

[BKR14b] BAUER U., KERBER M., REININGHAUS J.: Distributed computation of persistent homology. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments* (2014), pp. 31–38. doi:10.1137/1.9781611973198.4. 4

[BM14] BOISSONNAT J.-D., MARIA C.: Computing Persistent Homology with Various Coefficient Fields in a Single Pass. In *Algorithms - ESA 2014*, vol. 8737 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2014, pp. 185–196. doi:10.1007/978-3-662-44777-2_16. 4

[Bub15] BUBENIK P.: Statistical Topological Data Analysis Using Persistence Landscapes. *Journal of Machine Learning Research 16*, 1 (2015), 77–102. 2, 4, 5

[CCSG*09] CHAZAL F., COHEN-STEINER D., GLISSE M., GUIBAS L. J., OUDOT S. Y.: Proximity of Persistence Modules and their Diagrams. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry* (June 2009), pp. 237–246. doi:10.1145/1542362.1542407. 4

[CK11] CHEN C., KERBER M.: Persistent homology computation with a twist. In *Proceedings 27th European Workshop on Computational Geometry* (2011). 4

[CK13] CHEN C., KERBER M.: An Output-sensitive Algorithm for Persistent Homology. *Computational Geometry 46*, 4 (2013), 435–447. doi:10.1016/j.comgeo.2012.02.010. 4

[CSEH07] COHEN-STEINER D., EDELSBRUNNER H., HARER J.: Stability of persistence diagrams. *Discrete & Computational Geometry 37*, 1 (2007), 103–120. doi:10.1007/s00454-006-1276-5. 4

[CZ05] CARLSSON G., ZOMORODIAN A. J.: Computing Persistent Homology. *Discrete & Computational Geometry 33*, 2 (2005), 249–274. doi:10.1007/s00454-004-1146-y. 3, 5

[CZCG05] CARLSSON G., ZOMORODIAN A. J., COLLINS A., GUIBAS L. J.: Persistence Barcodes for Shapes. *International Journal of Shape Modeling 11*, 02 (2005), 149–187. doi:10.1142/S0218654305000761. 3

[DFL10] D'AMICO M., FROSINI P., LANDI C.: Natural Pseudo-Distance and Optimal Matching Between Reduced Size Functions. *Acta Applicandae Mathematicae 109*, 2 (2010), 527–554. doi:10.1007/s10440-008-9332-1. 4

[DFW14] DEY T. K., FAN F., WANG Y.: Computing Topological Persistence for Simplicial Maps. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry* (2014), pp. 345–354. doi:10.1145/2582112.2582165. 4

[DHKS13] DEY T. K., HIRANI A. N., KRISHNAMOORTHY B., SMITH G.: Edge Contractions and Simplicial Homology. *ArXiv preprint* (Apr. 2013). arXiv:1304.0664. 4

[DMV11] DE SILVA V., MOROZOV D., VEJDEMO-JOHANSSON M.: Dualities in Persistent (Co)Homology. *Inverse Problems 27*, 12 (2011), 124003 (17pp). doi:10.1088/0266-5611/27/12/124003. 4

[DW14] DŁOTKO P., WAGNER H.: Simplification of Complexes of Persistent Homology Computations. *Homology, Homotopy and Applications 16*, 1 (2014), 49–63. doi:10.4310/HHA.2014.v16.n1.a3. 4

[EH08] EDELSBRUNNER H., HARER J.: Persistent Homology - a Survey. *Contemporary mathematics 453* (2008), 257–282. doi:10.1090/conm/453. 1, 3

[EH10] EDELSBRUNNER H., HARER J.: *Computational topology: an introduction*. American Mathematical Society, 2010. 1, 2, 6

[ELZ02] EDELSBRUNNER H., LETSCHER D., ZOMORODIAN A. J.: Topological Persistence and Simplification. *Discrete & Computational Geometry 28*, 4 (2002), 511–533. doi:10.1109/SFCS.2000.89213. 3, 5, 7

[FID14] FUGACCI U., IURICICH F., DE FLORIANI L.: Efficient computation of simplicial homology through acyclic matching. In *Symbolic*

and Numeric Algorithms for Scientific Computing (SYNASC), 2014 16th International Symposium on (2014), pp. 587–593. doi:10.1109/SYNASC.2014.84. 4

[FL01] FROSINI P., LANDI C.: Size Functions and Formal Series. *Applicable Algebra in Engineering, Communications and Computing 12*, 4 (2001), 327–349. doi:10.1007/s002000100078. 3

[Fro90] FROSINI P.: A Distance for Similarity Classes of Submanifolds of a Euclidean Space. *Bulletin of the Australian Mathematical Society 42*, 3 (1990), 407–415. doi:10.1017/S0004972700028574. 3

[Fro92] FROSINI P.: Measuring Shapes by Size Functions. In *Intelligent Robots and Computer Vision X: Algorithms and Techniques* (Feb. 1992), vol. 1607 of *SPIE Proceedings*, pp. 122–133. doi:10.1117/12.57059. 3

[Ghr08] GHRIST R.: Barcodes: the Persistent Topology of Data. *Bulletin of the American Mathematical Society 45*, 1 (2008), 61–75. doi:10.1090/S0273-0979-07-01191-3. 3

[Ghr14] GHRIST R.: *Elementary Applied Topology*. Createspace Independent Pub, 2014. 1

[HMMN14] HARKER S., MISCHAIKOW K., MROZEK M., NANDA V.: Discrete Morse Theoretic Algorithms for Computing Homology of Complexes and Maps. *Foundations of Computational Mathematics 14*, 1 (2014), 151–184. doi:10.1007/s10208-013-9145-0. 4

[LF97] LANDI C., FROSINI P.: New Pseudodistances for the Size Function Space. In *Vision Geometry VI* (Oct. 1997), vol. 3168 of *SPIE Proceedings*, pp. 52–60. doi:10.1117/12.279674. 4

[LSV11] LIPSKY D., SKRABA P., VEJDEMO-JOHANSSON M.: A spectral sequence for parallelized persistence. *ArXiv e-prints* (Dec. 2011). arXiv:1112.1245. 4

[LZ14] LEWIS R. H., ZOMORODIAN A. J.: Multicore Homology via Mayer Vietoris. *ArXiv preprint* (July 2014). arXiv:1407.2275. 4

[MB09] MROZEK M., BATKO B.: Coreduction Homology Algorithm. *Discrete & Computational Geometry 41* (2009), 96–118. doi:10.1007/s00454-008-9073-y. 4

[MMS11] MILOSAVLJEVIĆ N., MOROZOV D., SKRABA P.: Zigzag persistent homology in matrix multiplication time. In *Proceedings of the twenty-seventh Annual Symposium on Computational Geometry* (2011), ACM, pp. 216–225. doi:10.1145/1998196.1998229. 4

[MNV13] MURTY N. A., NATARAJAN V., VADHIYAR S.: Efficient homology computations on multicore and manycore systems. In *20th Annual International Conference on High Performance Computing* (Dec. 2013), IEEE, pp. 333–342. doi:10.1109/HiPC.2013.6799139. 4

[Mun84] MUNKRES J.: *Elements of Algebraic Topology*. Advanced book classics. Perseus Books, 1984. 4

[MW10] MROZEK M., WANNER T.: Coreduction homology algorithm for inclusions and persistent homology. *Computers & Mathematics with Applications 60*, 10 (2010), 2812–2833. doi:10.1016/j.camwa.2010.09.036. 4

[OPT*15] OTTER N., PORTER M. A., TILLMANN U., GRINDROD P., HARRINGTON H. A.: A roadmap for the computation of persistent homology. *ArXiv e-prints* (June 2015). arXiv:1506.08903. 1

[plo15] Plotly technologies inc. collaborative data science, 2015. Motréal, QC. URL: https://plot.ly. 8

[Rob99] ROBINS V.: Towards Computing Homology from Finite Approximations. *Topology Proceedings 24*, 1 (1999), 503–532. URL: http://topology.auburn.edu/tp/reprints/v24/tp24222.pdf. 3

[RWS11] ROBINS V., WOOD P. J., SHEPPARD A. P.: Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images. *IEEE Trans. on Pattern Analysis and Machine Intelligence 33*, 8 (2011), 1646–1658. doi:10.1109/TPAMI.2011.95. 4

[Wei11] WEINBERGER S.: What is... Persistent Homology? *Notices of the American Mathematical Society 58*, 1 (2011), 36–39. 1

[Wri16] WRIGHT M. L.: Introduction to Persistent Homology. In *32nd International Symposium on Computational Geometry (SoCG 2016)* (Dagstuhl, Germany, 2016), vol. 51 of *Leibniz International Proceedings in Informatics*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 72:1–72:3. doi:10.4230/LIPIcs.SoCG.2016.72. 1

[ZC05] ZOMORODIAN A. J., CARLSSON G.: Computing Persistent Homology. *Discrete & Computational Geometry 33*, 2 (2005), 249–274. doi:10.1007/s00454-004-1146-y. 4, 6

[Zom05] ZOMORODIAN A. J.: *Topology for computing*. Cambridge University Press, 2005. URL: http://dl.merc.ac.ir/handle/Hannan/6824. 1

[Zom10] ZOMORODIAN A. J.: The Tidy Set: a minimal simplicial set for computing homology of clique complexes. In *Proceedings of the Annual Symposium on Computational Geometry* (2010), pp. 257–266. doi:10.1145/1810959.1811004. 4