九州大学
KYUSHU UNIVERSITY

Graduate School of Mathematics

Topological Filtering of Graph Signals: Persistence-Based Methods and Applications

Matias de Jong van Lier

PhD Thesis

Supervisor: Prof. Shizuo Kaji
Department: Faculty of Mathematics
Date: June 2025

Abstract

Topological methods have emerged as powerful tools in modern data analysis, providing robust frameworks for capturing the shape and structure of complex datasets. Among these, *persistent homology* has become particularly influential, as it quantifies topological features across multiple scales and distinguishes meaningful structure from noise through the notion of persistence.

Building on this foundation, we formalize the concept of *topological noise* as low-persistence features—those that arise from small perturbations in signal values. This thesis addresses the challenge of filtering such noise from real-valued signals defined on graphs and higher-order structures, with the objective of preserving topologically significant features while maintaining fidelity to the original signal.

We begin by demonstrating that conventional graph signal processing methods, including spectral filters and deep learning-based denoising techniques, are inadequate for removing topological noise without introducing significant distortion to either the signal or its topologically meaningful features. To address this shortcoming, we formalize the problem of persistence-aware filtering under strict structural constraints and show that, in general, no exact solution exists when attempting to simultaneously filter features across multiple homological dimensions. This result highlights a fundamental trade-off between denoising and topological fidelity in multi-dimensional settings.

To overcome this challenge, we propose a relaxed formulation and introduce a data structure—the *Basin Hierarchy Tree*—which encodes persistence information hierarchically and supports efficient, topology-aware filtering. Building on this structure, we develop the *Low Persistence Filter*, a method that selectively removes low-persistence features while preserving high-persistence structures within a controlled approximation bound. The method is first formulated for graph signals and then extended to graphs with faces, enabling the simultaneous filtering of 0- and 1-dimensional features. Theoretical guarantees are established, and our main result ensures topological fidelity under explicitly defined tolerances.

We validate our approach by implementing the filter in an open-source software and by applying it to a series of illustrative examples. Additionally, we extend the filtering framework to incorporate size-based criteria and define summary statistics derived from the Basin Hierarchy Tree. In a synthetic classification task, these statistics outperform traditional persistent homology-based features and even exceed the accuracy of state-of-the-art deep learning models. Collectively, this work advances both the theory and practice of topological signal processing, offering scalable tools for persistence-guided filtering.

Contents

List of Figures

List of Tables

Acknowledgments

This journey would not have been possible without the support, guidance, and kindness of many remarkable individuals to whom I owe my deepest gratitude.

First and foremost, I am profoundly grateful to Professor Shizuo Kaji and Professor Norio Iwase. Their mentorship, encouragement, and insightful guidance were invaluable throughout my doctoral studies. It has been a true privilege to learn from them.

I am sincerely thankful to the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) for granting me the scholarship that made this possible. Their generous support enabled me to pursue my research in Japan and to fully dedicate myself to my studies.

I would also like to express my appreciation to all the administrative and technical staff at Kyushu University. Their professionalism, efficiency, and kindness helped create a smooth and supportive environment for both my academic and daily life.

Special thanks go to Sebastián E. G. Zurita and Junyan Chu, with whom I had the pleasure of collaborating. Their companionship and stimulating discussions greatly enriched this research.

To all the friends I made during my time in Japan: thank you for the warmth, laughter, and unforgettable memories. You made my years at Kyushu University truly enjoyable.

I want to thank my parents, Iara and Quirijn, for their constant support throughout my life and, even now, from afar. I am also especially grateful to other family members—Maíra, Joop, Cláudio, Marlene, and Satie—your support, though distant, has meant a great deal to me.

I am profoundly grateful to my wonderful wife, Carolina, whose patience, encouragement, and loving companionship have supported me through every step of the way.

Chapter I

Introduction

The Rise of Topological Methods. Topology is a fundamental area of mathematics concerned with the properties of spaces that are preserved under continuous transformations such as stretching, bending, and twisting, but not tearing or gluing [Mun2000]. The term itself derives from the Greek words *topos* (place) and *logos* (study), emphasizing its focus on spatial relationships. It is often referred to as "rubber-sheet geometry," reflecting the idea that, in topology, objects can be continuously deformed—like a rubber sheet—without changing their essential properties. Topology thus provides a framework for studying continuity, compactness, and connectedness in a general setting.

The origins of topology can be traced back to the 18th century, particularly to the work of Leonhard Euler, whose solution to the Königsberg bridge problem in 1736 is widely regarded as a foundational moment in the field [Eul1736]. However, topology did not emerge as a distinct mathematical discipline until the late 19th and early 20th centuries, when mathematicians such as Henri Poincaré, Georg Cantor, and Felix Hausdorff formalized many of its concepts [Poi1895, Can1883, Hau1914, Kle1926]. Poincaré's work on analysis situs laid the groundwork for algebraic topology [Poi1895], while Cantor's development of set theory provided the language and tools necessary for modern topological analysis [Can1883]. Since then, topology has evolved into several subfields, including point-set topology [Mun2000], algebraic topology [Hat2002], and differential topology [GP2010], each offering a unique perspective on spatial structure and continuity.

Algebraic topology is a central branch of topology that seeks to understand topological spaces by translating them into algebraic structures. The key idea is to associate algebraic invariants—such as groups, rings, or modules—with topological spaces in such a way that topological properties correspond to algebraic ones. This correspondence enables the study of complex spaces through the well-established tools of algebra. Among the most fundamental concepts in this field are homotopy, homology, and cohomology. These constructions assign algebraic objects to topological spaces, allowing for the extraction of topological invariants such as the number and type of holes in various dimensions.

Historically, the foundations of algebraic topology were laid in the late 19th and early 20th centuries. Henri

Poincaré's pioneering work on *analysis situs* [Poi1895] introduced the concept of the fundamental group and laid the groundwork for homotopy and homology theory. Over time, these early ideas were rigorously formalized and expanded upon by mathematicians such as Emmy Noether, Leopold Vietoris, and Heinz Hopf, who contributed to the development of homology groups and cohomology rings [Noe1927, Vie1927, Hop1931]. The mid-20th century saw further advances through the work of Eilenberg and Steenrod [ES1952], who axiomatized homology theory, and later developments introduced powerful tools like spectral sequences [McC2001] and homotopy theory [Whi2012].

Algebraic topology has become a foundational tool not only in pure mathematics but also in theoretical physics, robotics, and data analysis. In physics, it plays a key role in the classification of topological phases of matter [Wen2017] and in gauge theories [NS1988, Nak2018]. In robotics, topological methods have been applied to the motion planning problem, where configuration spaces are analyzed using homological invariants, and the inherent difficulty of planning is captured by the notion of topological complexity [Far2003, Far2004]. In data analysis, the introduction of persistent homology has revolutionized the use of topological invariants to extract information from data—a development that we explain in greater detail in the following paragraphs.

Topology and data. While algebraic topology has long been a cornerstone of theoretical physics and pure mathematics, its applications have grown increasingly broad and interdisciplinary. One of the most significant recent developments is its role in data science, where topological methods provide tools to extract meaningful structure from complex, high-dimensional data. In particular, this thesis will focus on a specific topological method—*persistent homology*—developed within the field of *topological data analysis* (TDA) [Car2009, EH2010a]. This framework adapts classical ideas from algebraic topology to suit the discrete, noisy, and finite nature of real-world data.

The origins of topological data analysis (TDA) and persistent homology can be traced to several independent developments occurring in the late 20th and early 21st centuries. The concept of persistence emerged simultaneously in three different contexts: the size theory work of Frosini and Ferri [Fro1992, VUFF1993]; the doctoral research of Vanessa Robins [Rob1999]; and the biogeometry project led by Herbert Edelsbrunner [ELZ2002]. Each of these lines of research introduced ideas central to the modern understanding of persistent homology, with key developments unfolding over a period of roughly fifteen years.

The central idea of persistent homology is to study the evolution of topological features—such as connected components, loops, and voids—across a filtration: a nested sequence of topological spaces constructed from data. By tracking the birth and death of these features as the filtration progresses, persistent homology captures not only which features exist, but also over which scales they persist. The result is a multiscale summary of the data's topological structure, where long-lived features are typically interpreted as meaningful signal and short-lived ones as noise.

Persistent homology has evolved into a foundational tool in TDA, supported by both theoretical development and algorithmic innovation. The construction of persistence diagrams and barcodes provides visual and computational representations of topological summaries. Moreover, advances in the theory have shown that these invariants are robust under small perturbations of the input data [CSEH2007], making them suitable for practical applications. Efficient implementations, such as those in the software libraries `GUDHI`, `Ripser` and `Cubical-Ripser` [MBGY2014, Bau2021, KSA2020], have made the method accessible for large-scale analysis.

Applications of persistent homology span a wide array of domains. In neuroscience, it helps characterize

the structure of neural activity [GPCI2015]. In materials science, it captures the geometry of porous media [HNH+2016]. In genomics and bioinformatics, it reveals patterns in complex molecular data [XW2014]. These successes demonstrate that persistent homology is not merely a theoretical innovation but a versatile analytic framework.

Signal Processing Beyond the Grid. The classical theory of signal processing has traditionally focused on functions defined over regular, Euclidean domains such as time or space. However, many modern datasets arise in irregular or structured domains where the underlying geometry is inherently non-Euclidean. Examples include social and communication networks [CF2006, For2010], sensor grids [ASSC2002], biological systems [New2003], and data defined on manifolds or meshes [Tau2000]. These structures are often modeled as graphs or simplicial complexes, where classical notions of convolution, frequency, and smoothness must be redefined.

The growing need to analyze signals on such domains has led to the emergence of *graph signal processing* (GSP), which generalizes signal processing tools to graphs by using spectral graph theory [SNF+2013, SM2013]. In this framework, signals are defined as scalar functions on the nodes of a graph, and graph operators—such as the graph Laplacian—are used to encode the structure of the domain. Spectral filtering in GSP is achieved by projecting graph signals onto the eigenbasis of the Laplacian, allowing frequency-domain interpretations analogous to classical Fourier analysis. Low-frequency components correspond to smooth variations over the graph, while high-frequency components capture rapid changes with respect to the graph topology.

This spectral approach enables the design of graph filters, graph-based convolutions, and notions of smoothness and regularization that respect the intrinsic geometry of the data. These tools have been successfully applied to a wide range of problems, including denoising, compression, semi-supervised learning, and anomaly detection on graphs [CSMK2014, OFK+2018].

Quantifying and Filtering Topological Noise. In this work, we study real-valued signals defined on graphs and on higher-order structures—specifically, graphs augmented with two-dimensional faces, forming a generalized version of a cell complex. These signals are interpreted as functions on the nodes of a graph or cell complex. We investigate their topological structure by analyzing the persistent homology of their sublevel set filtrations. This methodology enables us to quantify the presence of topological features such as connected components and loops, and to evaluate their significance using the concept of persistence.

Persistent features that endure over a wide range of filtration values are considered topologically meaningful, whereas features with low persistence—below a chosen threshold—are typically treated as topological noise. The intuition behind this classification stems from the observation that small perturbations in the function values of a signal over a graph can give rise to new features with proportionally low persistence. In this sense, persistence acts as a stable and scale-sensitive measure for distinguishing noise from signal.

A central question addressed in this thesis is whether such noise can be effectively reduced using graph-based filtering techniques that preserve the integrity of topologically significant features. Specifically, can we construct a filtered approximation of a given signal in which all features below a certain persistence threshold are eliminated, while those above the threshold remain unaffected?

We start by investigating the effect of spectral filtering, denoising methods and deep learning denoising methods on the topological noise of signals. Our findings show that standard spectral filters are insufficient for removing topological noise in a robust and controlled manner. Specifically, two key limitations arise. First,

filtering can alter the signal significantly, shifting its values beyond an acceptable threshold ϵ, thereby distorting the signal's geometric structure. Second, filtering may unintentionally suppress or distort topologically relevant features, diminishing their persistence and obscuring critical global information about the signal's topology.

From Spectral Smoothing to Topological Integrity. The observed limitations of spectral filtering in controlling topological noise motivate the need for alternative approaches. In particular, we aim to address the challenge of designing filtering methods that can effectively suppress low-persistence features—interpreted as topological noise—while preserving the integrity of topologically significant structures. Such methods should exhibit stability under perturbations, ensure minimal distortion of the original signal, and maintain the persistence of relevant features that encode global geometric and topological information.

This thesis is thus driven by the goal of developing a framework for topologically informed signal filtering. By incorporating persistent homology into the filtering process, we seek to construct transformations that are aware of the underlying topological structure and capable of selectively modifying signals in a persistence-sensitive manner.

Structural Constraints and the Limits of Prior Approaches. Several studies have explored how persistent homology can guide the simplification of functions, especially when those functions are defined on surfaces. A foundational contribution by Edelsbrunner, Morozov, and Pascucci [EMP2006] presented a method for approximating scalar fields on manifolds by removing features with low persistence. Similarly, Attali and collaborators [AGH+2009] developed fast algorithms for comparable simplification tasks. Although these techniques are both rigorous and efficient, they frequently require modifications to the domain itself—such as inserting extra vertices and edges or adjusting the signal values on existing edges.

Such structural alterations are unsuitable for numerous graph signal processing applications, where the network topology must remain unchanged and only the signal values associated with the nodes can be modified. Our method is specifically tailored for these more strict constraints.

Within this constrained setting, we identify and formalize inherent barriers to persistent simplification. Through carefully constructed counterexamples, we demonstrate that it is, in general, impossible to eliminate all low-persistence features without also perturbing high-persistence features—highlighting a fundamental trade-off between noise removal and topological fidelity. In response, we propose a theoretically justified and practically implementable approximation strategy that achieves optimal simplification under these constraints, preserving the most relevant topological features while minimizing unnecessary distortion.

Structural Foundations for Efficient Topological Filtering. Before addressing the construction of persistence-based filtering methods, we introduce a specialized data structure designed to efficiently encode the information required for suppressing low-persistence features while preserving those of higher persistence. We refer to this structure as the *Basin Hierarchy Tree*, which is conceptually based on a modification of the classic Union-Find algorithm [GF1964]. Its hierarchical organization reflects the structure of persistent homology: features with higher persistence are positioned higher in the tree, enabling prioritized access and targeted simplification.

The primary advantage of this structure lies in its ability to encode persistence information in a way that is both compact and conducive to efficient traversal and update operations during filtering. We demonstrate that, in general, filtering persistent features within a single homology dimension is relatively straightforward. The main difficulty emerges when attempting to simultaneously filter features across multiple homological

dimensions. In such cases, strict separation of persistence levels becomes infeasible without allowing some degree of interference with high-persistence features. This observation highlights a fundamental complexity in multi-dimensional topological simplification, which we address through a carefully balanced approximation strategy.

Contributions and the Roadmap Ahead. In Chapter II, we begin by introducing essential background from Graph Signal Processing (Section 1) and Topological Data Analysis (Section 2). Our first contribution appears in Section 3, where we demonstrate that spectral filtering, digital signal processing techniques, and deep learning-based denoising methods all tend to reduce topological noise, but ultimately fail to fully eliminate it. Moreover, they do not consistently preserve high-persistence, topologically meaningful features—as evidenced by variations in feature count, increases in Bottleneck distance, and significant distortion of the signal structure, measured via the sup-norm between the original and filtered signals.

In Sections 4 and 5, we formalize the task of filtering topological noise while preserving topologically significant features and maintaining fidelity to the original function (Problem 5.1). We provide a counterexample (Section 5.2) that illustrates the general impossibility of solving this problem under strict constraints. This limitation is shown to stem from the challenge of filtering simultaneously across multiple homological dimensions. Later, we demonstrate that the problem is solvable when restricted to a single homological dimension.

To address this limitation, we propose a relaxed formulation of the problem (Problem 5.4) in which high-persistence features are permitted to vary within a specified tolerance. Under this relaxation, features are only allowed to diminish slightly in persistence—never to increase—thereby avoiding the introduction of artificially prominent topological structures in the filtered signal. This controlled relaxation enables practical filtering that balances topological noise suppression with structural fidelity.

In Chapter III, we develop the core methodology of this thesis. We begin, in Section 6, by introducing the *Basin Hierarchy Tree*, as previously described, a data structure designed to efficiently encode persistence information in a form suitable for persistence-based filtering. This structure forms the computational foundation for the filtering process.

The filtering algorithm itself is presented in Section 7. We first address the case of graph signals, focusing on the filtering of 0-dimensional homological features (i.e., connected components) while preserving high-persistence structures. Subsequently, we extend the method to two-dimensional generalized cell complexes—referred to as *graphs with faces*—which support the representation of both 0- and 1-dimensional features within a unified framework. This generalization introduces additional challenges due to the increased complexity, which we address by developing a persistence-aware approximation strategy that selectively suppresses low-persistence features while preserving the global topological structure as faithfully as possible to the original signal. This preservation is formally guaranteed by our main result, stated in Theorem 7.10.

In Chapter IV, we explore several applications and extensions of the concepts developed in the previous chapter. We begin, in Section 8, with a brief overview of our open-source software implementation of the Low Persistence Filter, which is available at `github.com/mvlier/topapprox`. This is followed by an analysis of the filter's behavior through a series of illustrative examples of varying nature, presented in Section 9.

In Section 10, we introduce an extension of the Low Persistence Filter, termed the *Size-Aware Low Persistence Filter*. This variant takes into account not only persistence levels but also the geometric size of topological

features, requiring two thresholds for filtering. We show that this extension arises naturally from the structure of the Basin Hierarchy Tree (BHT), which is inherently suited to capturing both the persistence and spatial extent of features, thereby enhancing the expressiveness and versatility of the filtering process.

Finally, in Section 11, we define a set of summary statistics derived from the BHT. This is motivated by the observation that the BHT encodes richer structural information than the persistence diagram alone. We demonstrate the utility of these statistics by employing them as feature vectors for a binary classification task involving a synthetic image dataset with varying levels of hierarchical structure. The results show that these BHT-based descriptors outperform all other persistent homology-derived features tested, and even surpass the classification accuracy achieved by state-of-the-art deep convolutional neural networks, emphasizing the practical effectiveness of the BHT framework.

This thesis builds upon the work presented in [vLZK2024].

Chapter II

Framing the Problem: Motivations and Challenges

Signal processing on graphs has emerged as a powerful framework for analyzing data with non-Euclidean structure, enabling tools such as spectral filtering and graph Fourier transforms to be applied in a variety of domains [SNF+2013, OFK+2018]. However, standard graph signal processing (GSP) methods are fundamentally restricted to signals defined over vertices or edges of graphs, and often fail to capture higher-dimensional or topological features inherent in complex datasets. To overcome these limitations, recent extensions of GSP have sought to incorporate higher-order structures, such as simplicial complexes and cellular complexes [SBT2021, PL2022], enabling richer representations and a finer understanding of data supported on 2D and higher-dimensional domains.

In parallel, *Topological Data Analysis* (TDA)—and in particular, *persistent homology*—has established itself as a robust approach for extracting multi-scale topological features from data. Persistence diagrams provide a compact summary of the homological structure of a space across a filtration [EH2010b, Oud2015], highlighting features such as connected components, loops, and voids that persist across different scales. These persistent features offer a compelling target for filtering techniques, especially when the goal is to preserve meaningful topological information while removing noise [CSEH2007, CdSGO2016].

In this chapter, we aim to bridge these two perspectives by introducing a framework for filtering signals supported on *graphs with 2D faces*. We begin by reviewing core concepts in graph signal processing, followed by a brief overview of persistent homology. We then define the central object of our study: the *graph with faces*, a structure that serves as the domain for our signal and topological analysis. These graphs are slightly more general than 2D cell complexes in that they allow for faces with non-disk-like topology, including holes or more intricate embeddings. To handle this generality, we introduce a method for computing their persistent homology that is centered around the *Alexander Duality* [Ale1915], which allows us to reason about the topological features of the embedding from a complementary perspective.

Our ultimate goal is to formulate a filtering strategy that removes topological noise from a signal—interpreted in terms of low-persistence homological features—while preserving significant structures. To this end, we propose a formulation of the filtering problem as a minimization task that incorporates persistent homology. However, we show through a carefully constructed counterexample that this original formulation is, in general, impossible to solve. The core issue arises when attempting to suppress low-persistence features across multiple homological dimensions simultaneously, while preserving the higher persistence features, a task that proves too rigid for practical applications.

This negative result motivates a revision of the problem statement. We conclude the chapter by presenting a more flexible and realistic formulation that relaxes the original constraints while retaining the essential objective: to develop topology-aware filters for signals on graphs with faces. A solution to this revised problem will be developed in the chapters that follow.

SECTION 1

Graph Signal Processing

Graph Signal Processing (GSP) extends classical signal processing concepts to data defined on graphs. A graph $G = (V, E)$ consists of a finite set of vertices V and a set of edges $E \subseteq V \times V$. Let $|V| = n$ and $|E| = m$. We fix an ordering of the vertices, denoted $V = \{v_1, \ldots, v_n\}$, and assume that the graph is *oriented* in the sense that each edge $e = v_i v_j$ is assigned an orientation from the vertex with the lower index to the one with the higher index (i.e., $i < j$).

For the purposes of this discussion on Graph Signal Processing, we assume that all graphs are ; that is, each edge connects two distinct vertices and there are no self-loops or multiple edges between the same pair of vertices. More general graph structures may be considered in later sections.

Definition 1.1 (Graph Signal)**.** A *graph signal* is a function $f : V \to \mathbb{R}$, assigning a scalar value to each node. The space of all such functions is isomorphic to $\mathbb{R}^n$. In the language of algebraic topology, this corresponds to the space of 0-cochains on the graph:

$$C^0(G; \mathbb{R}) = \mathrm{Hom}(C_0(G), \mathbb{R}),$$

where $C_0(G)$ is the space of 0-chains.

Definition 1.2 (Edge Signal)**.** A signal defined on the edges of a graph is a function $g : E \to \mathbb{R}$, often referred to as an *edge flow* in analogy with vector fields. Such signals correspond to 1-cochains:

$$C^1(G; \mathbb{R}) = \mathrm{Hom}(C_1(G), \mathbb{R}),$$

where $C_1(G)$ is the space of 1-chains.

Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, signals $f \in C^0(G)$ and $g \in C^1(G)$ can be represented as column vectors:

$$f = [f(v_1), f(v_2), \ldots, f(v_n)]^\top \in \mathbb{R}^n,$$

$$g = [g(e_1), g(e_2), \ldots, g(e_m)]^\top \in \mathbb{R}^m.$$

Here, a fixed ordering of the vertices and edges is assumed.

We will adopt these conventions throughout the text: $C^0(G) \cong \mathbb{R}^n$ will denote node signals, and $C^1(G) \cong \mathbb{R}^m$ will denote edge signals.

Remark 1.3 . Graph signals are naturally interpreted as elements of the cochain complex of the graph:

$$C^0(G;\mathbb{R}) \xrightarrow{\delta^0} C^1(G;\mathbb{R}) \xrightarrow{\delta^1} C^2(G;\mathbb{R}) \to \cdots,$$

where δ^k are the coboundary operators. For most graphs arising in applications (i.e., 1-dimensional simplicial complexes), $C^k(G) = 0$ for $k \geq 2$.

Definition 1.4 (Gradient and Divergence on Graphs)**.** Let $G = (V, E)$ be a finite oriented graph with $n = |V|$ vertices and $m = |E|$ edges. Let $B \in \mathbb{R}^{n\times m}$ denote the *oriented incidence matrix* of G, which represents the boundary operator $\partial_1 : C_1(G;\mathbb{R}) \to C_0(G;\mathbb{R})$, defined by

$$B_{v,e} = \begin{cases} +1 & \text{if } v = \text{head of edge } e, \\ -1 & \text{if } v = \text{tail of edge } e, \\ 0 & \text{otherwise.} \end{cases}$$

Equivalently, B is the matrix representation of the adjoint of the 0-coboundary operator, that is, $\delta^{0^*} : C^1(G;\mathbb{R}) \to C^0(G;\mathbb{R})$.

Then:

- The *discrete divergence* is given by the boundary operator:

$$\delta^{0^*} : C^1(G;\mathbb{R}) \to C^0(G;\mathbb{R}), \qquad \delta^{0^*} = B,$$

  which maps edge flows to net flux at nodes. For $g \in C^1(G;\mathbb{R}) \cong \mathbb{R}^m$, the divergence at a vertex v_i is

$$(\delta^{0^*} g)(v_i) = \sum_{e=v_i v_j} g(e) - \sum_{e=v_k v_i} g(e).$$

- The *discrete gradient* is given by the coboundary operator:

$$\delta^0 : C^0(G;\mathbb{R}) \to C^1(G;\mathbb{R}), \qquad \delta^0 = B^\top,$$

  which maps node signals to edge differences. For $f \in C^0(G;\mathbb{R}) \cong \mathbb{R}^n$, the gradient along edge e is

$$(\delta^0 f)(e) = f(\text{head}(e)) - f(\text{tail}(e)).$$

Just as in classical signal processing, where one studies functions $f : [0,1] \to \mathbb{R}$ or $F : [0,1]^2 \to \mathbb{R}$, the domain is now generalized to a graph. In the continuous setting, the Laplace operator $\Delta F = -\operatorname{div}(\operatorname{grad}(F))$ admits a complete set of orthonormal eigenfunctions—sines and cosines—that define the Fourier basis.

These eigenfunctions correspond to eigenvalues that grow quadratically and represent squared frequencies. This spectral structure enables the decomposition of any square-integrable function into a sum of oscillatory modes, with the eigenvalues indicating their frequency content.

This motivates the definition of a Laplacian in the graph setting. Noting that B is analogous to the divergence operator and $B^\top$ to the gradient operator, the graph Laplacian is naturally defined as $L = BB^\top$.

Analogously to the Euclidean case, the eigenvectors of the graph Laplacian form a generalized Fourier basis. The corresponding eigenvalues can be interpreted as frequencies: small eigenvalues correspond to smooth, slowly varying signals across the graph (low frequencies), while large eigenvalues correspond to rapidly varying, oscillatory signals (high frequencies). This analogy underpins the definition of the Fourier transform on graphs.

The following definition presents two equivalent formulations of the graph Laplacian. Recall that the degree of a vertex v in a graph is the number of edges incident to it, denoted $\deg(v)$.

Definition 1.5 (Graph Laplacian)**.** Let $A \in \mathbb{R}^{n\times n}$ be the adjacency matrix of G, defined by

$$A_{ij} = \begin{cases} 1, & \text{if } v_i v_j \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Let $D \in \mathbb{R}^{n\times n}$ be the degree matrix, defined by $D_{ii} = \deg(v_i)$ and $D_{ij} = 0$ for $i \neq j$. The *combinatorial Laplacian* (or simply, the *graph Laplacian*) is given by

$$L = D - A.$$

Alternatively, using the oriented incidence matrix $B \in \mathbb{R}^{n\times m}$, the Laplacian can be expressed as

$$L = BB^\top,$$

which is equivalent to the definition above.

The space of signals on a graph inherits a natural inner product from $\mathbb{R}^n$, namely

$$\langle f, g\rangle = f^\top g = \sum_{v\in V} f(v)g(v),$$

making it a finite-dimensional Hilbert space. This inner product allows us to define several standard notions from classical signal processing:

- The ℓ_2-*norm* of a signal $f \in C^0(G)$ is given by

$$\|f\|_2 = \sqrt{\langle f, f\rangle} = \left(\sum_{v\in V} f(v)^2\right)^{1/2}.$$

- Two signals $f, g \in C^0(G)$ are *orthogonal* if $\langle f, g\rangle = 0$.
- The *energy* of a signal f is defined as $\|f\|_2^2 = \langle f, f\rangle$.

- The *projection* of a signal f onto a nonzero signal g is given by

$$\text{proj}_g(f) = \frac{\langle f, g \rangle}{\langle g, g \rangle} g.$$

These structures enable the application of spectral graph theory to analyze and decompose signals using the eigenbasis of the Laplacian, analogous to classical Fourier analysis on $L^2([0,1])$, the space of square-integrable functions on the interval $[0,1]$. This correspondence is formalized by the following theorem.

Theorem 1.6 (Spectral Graph Theory)**.** Let L be the Laplacian of G. Then:

- L has real, non-negative eigenvalues: $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.
- The multiplicity of the zero eigenvalue equals the number of connected components of G.
- The eigenvectors of L form an orthonormal basis of $\mathbb{R}^n$ and define the *graph Fourier basis*.

These results follow from the fact that L is symmetric and positive semi-definite.

The theorem above is a direct consequence of the spectral theorem for real symmetric matrices. The symmetry of L follows from the fact that $L = BB^T$. Moreover, the non-negativity of the eigenvalues follows from the positive semi-definiteness of L. This property is evident since for any $x \in \mathbb{R}^n$,

$$\langle x, Lx \rangle = \langle x, BB^\top x \rangle = \langle B^\top x, B^\top x \rangle = \|B^\top x\|_2^2 \geq 0.$$

The fact that the multiplicity of the zero eigenvalue equals the number of connected components of the graph is a consequence of the Hodge Decomposition (see Theorem 1.9) and the well known fact that $\ker(L) \simeq H_0(G)$.

Definition 1.7 (Graph Fourier Transform)**.** Let $\{u_1, \ldots, u_n\}$ be the orthonormal eigenvectors of the combinatorial Laplacian L with corresponding eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$. For a signal $f \in C^0(G)$, the Graph Fourier Transform (GFT) is defined as:

$$\hat{f}_k = \langle f, u_k \rangle, \quad \text{for } k = 1, \ldots, n,$$

where $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product on $\mathbb{R}^n$.

The inverse Graph Fourier Transform reconstructs the signal as

$$f = \sum_{k=1}^{n} \hat{f}_k u_k.$$

Let $U \in \mathbb{R}^{n \times n}$ be the orthogonal matrix whose columns are the eigenvectors $u_1, \ldots, u_n$ of L, i.e., $L = U \Lambda U^\top$, where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$. Then the Graph Fourier Transform and its inverse can be expressed in matrix form as:

$$\hat{f} = U^\top f, \qquad f = U \hat{f}.$$

Remark 1.8 . In the case where G admits higher-order structure—such as a simplicial or cell complex—the chain complex $(C_\bullet(G), \partial)$ and its dual cochain complex $(C^\bullet(G), \delta)$ provide a natural algebraic-topological

framework for higher-order Graph Signal Processing, sometimes referred to in the literature as *Topological Signal Processing* [BS2020]. The k-th Hodge Laplacian is defined as

$$L_k = \delta^{k-1}\delta^{k-1\top} + \delta^{k\top}\delta^k,$$

and generalizes the graph Laplacian to k-cochains.

Theorem 1.9 (Hodge Decomposition)**.** Let L_k be the k-th Hodge Laplacian acting on $C^k(G;\mathbb{R})$. Then every k-cochain decomposes orthogonally as

$$C^k(G;\mathbb{R}) = \ker L_k \oplus \operatorname{im}(\delta^{k-1}) \oplus \operatorname{im}(\delta^{k\top}).$$

The space $\mathcal{H}^k := \ker L_k$ is called the space of *k-harmonic cochains*, and satisfies the isomorphism

$$\mathcal{H}^k \cong H^k(G;\mathbb{R}),$$

where $H^k(G;\mathbb{R}) = \ker \delta^k / \operatorname{im}(\delta^{k-1})$ is the k-th cohomology group of the graph.

In the case of finite-dimensional chain complexes—as in the setting of graphs—the Universal Coefficient Theorem implies that the homology and cohomology groups are naturally isomorphic over a field:

$$H^k(G;\mathbb{R}) \cong H_k(G;\mathbb{R}).$$

Therefore, the space of harmonic k-cochains is also isomorphic to the k-th homology group:

$$\mathcal{H}^k \cong H_k(G;\mathbb{R}).$$

1.1 Spectral Filtering of Graph Signals

The spectral framework established by the eigenbasis of the Graph Laplacian enables the definition of linear operators acting on graph signals through spectral multipliers, generalizing the classical notion of filtering.

Let $f \in C^0(G) \cong \mathbb{R}^n$ be a graph signal, and let $L = U\Lambda U^\top$ be the eigendecomposition of the Graph Laplacian, where $U = [u_1, \ldots, u_n]$ is the matrix of orthonormal eigenvectors and $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues.

Given a real-valued function $h : \mathbb{R} \to \mathbb{R}$, referred to as the *spectral response* (or *transfer function*), we define the associated *spectral filter* as the operator $H = h(L)$, given explicitly by

$$H = Uh(\Lambda)U^\top,$$

where $h(\Lambda) := \operatorname{diag}(h(\lambda_1), \ldots, h(\lambda_n))$.

Applying H to the signal f yields the filtered signal:

$$Hf = Uh(\Lambda)U^\top f.$$

In the spectral domain, this corresponds to applying the spectral response pointwise to the Graph Fourier transform $\hat{f} = U^\top f$:

$$(\widehat{Hf})_k = h(\lambda_k)\hat{f}_k, \quad \text{for } k = 1, \dots, n.$$

This framework enables the processing of graph signals by modulating their spectral content according to the Laplacian eigenvalues, which capture the structural properties of the graph—analogous to how filtering is performed in classical Euclidean domains. A key difference, however, is that unlike the uniform structure of Euclidean spaces, graphs often exhibit irregular connectivity, leading to a non-uniform distribution of frequencies (i.e., Laplacian eigenvalues).

Definition 1.10 (Spectral Graph Filter)**.** A spectral graph filter with transfer function $h : \mathbb{R} \to \mathbb{R}$ is the linear operator $H : C^0(G) \to C^0(G)$ defined as

$$H = h(L) = Uh(\Lambda)U^\top.$$

Example 1.11 (Spectral Filters)**.**

- **Heat kernel:** The filter
$$h(\lambda) = e^{-\tau\lambda}$$
attenuates high-frequency components, promoting smoothness of the signal across the graph. This corresponds to the *heat kernel* and arises from the solution of the heat diffusion equation $\partial_t f = -Lf$ on the graph. The parameter $\tau > 0$ controls the extent of diffusion and determines the smoothing scale.
- **Ideal low-pass filter:** The sharp cutoff filter
$$h(\lambda) = \mathbb{1}_{\leq \lambda_c} = \begin{cases} 1 & \text{if } \lambda \leq \lambda_c, \\ 0 & \text{otherwise,} \end{cases}$$
retains only the components of the signal associated with low frequencies.

In general, spectral filters lack spatial localization due to the global nature of the eigenvectors. Later, when analyzing the case of a regular grid, we will also consider denoising techniques from Digital Signal Processing, such as Gaussian Blur, Median Filtering, and Total Variation, which account for more localized properties.

Polynomial Filters and Spectral Approximation

A key advantage in practice is that filters can be approximated using polynomials in the Laplacian. A *polynomial filter* of degree K has the form

$$h(\lambda) = \sum_{k=0}^{K} a_k \lambda^k,$$

so that the corresponding operator becomes

$$H = h(L) = \sum_{k=0}^{K} a_k L^k.$$

This formulation avoids the need for Laplacian eigendecomposition and allows efficient, localized implementation using only sparse matrix-vector multiplications. Polynomial filters can approximate a wide class of spectral responses, and techniques such as Chebyshev or Jacobi polynomial expansions are commonly used in applications.

Alternative Laplacian Operators

The discussion above about spectral filtering is not limited to the graph Laplacian; it extends to any symmetric positive semidefinite operator. In what follows, we introduce two useful variants of the Laplacian and discuss their respective advantages. These alternatives—the normalized Laplacian and the Laplacian with Gaussian weights—will be employed in later sections of our analysis.

Normalized Laplacian. Given the unweighted adjacency matrix $A \in \mathbb{R}^{n\times n}$ as defined previously, and the corresponding degree matrix D, the *normalized Laplacian* is defined as

$$\mathcal{L}_{\text{norm}} = I - D^{-1/2} A D^{-1/2}. \tag{1.1}$$

This matrix is symmetric and positive semidefinite, and all its eigenvalues lie in the interval $[0, 2]$. The normalized Laplacian is particularly advantageous in contexts where the node degrees vary significantly, as it mitigates the dominance of high-degree nodes in the spectral representation. This is especially useful in filtering tasks, where a balanced influence of all nodes is desirable for stable and meaningful signal processing.

Laplacian with Gaussian Weights. An alternative to uniform weighting is to use a weighted adjacency matrix based on similarity between nodes. A common choice is the Gaussian (or radial basis function) kernel:

$$W_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), & \text{if } v_i v_j \in E, \\ 0, & \text{otherwise,} \end{cases}$$

where $x_i, x_j \in \mathbb{R}^d$ are feature vectors associated with nodes v_i and v_j, and $\sigma > 0$ is a scale parameter. In some settings, particularly when the graph structure is fixed but the signal varies, it is advantageous to define the weights using the signal values themselves. That is, for a real-valued signal $f : V \to \mathbb{R}$, one can define

$$W_{ij} = \begin{cases} \exp\left(-\frac{(f_i - f_j)^2}{2\sigma^2}\right), & \text{if } v_i v_j \in E, \\ 0, & \text{otherwise,} \end{cases} \tag{1.2}$$

where $f_i = f(v_i)$. This results in a signal-dependent Laplacian that adapts to the local variation in the signal. Such constructions are particularly effective for tasks that benefit from edge-aware smoothing, such as denoising, as they preserve discontinuities in the signal by reducing the weights between nodes with large signal differences.

Given the resulting weighted adjacency matrix W, the corresponding *(unnormalized) Laplacian with Gaussian*

weights is defined as

$$L = D - W, \tag{1.3}$$

where D is the diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$.

This adaptive weighting scheme provides a flexible and data-driven way to encode structure in the graph, making the Laplacian more aligned with the properties of the signal being processed.

These two modifications provide greater flexibility in modeling and processing graph signals, and we will revisit them in the context of our subsequent analysis. Moreover, the two approaches are not mutually exclusive: the normalized Laplacian can also be constructed using Gaussian weights instead of uniform ones, yielding a *normalized Gaussian-weighted Laplacian*. Given a Gaussian-weighted adjacency matrix W defined as in Equation (1.2), the corresponding degree matrix is D, with $D_{ii} = \sum_j W_{ij}$, and the normalized Gaussian-weighted Laplacian is defined as

$$\mathcal{L}_{\text{norm}}^{(\text{Gauss})} = I - D^{-1/2} W D^{-1/2}. \tag{1.4}$$

This construction combines the benefits of degree normalization with geometry- or signal-aware weighting, and is particularly effective in applications where both structural imbalance and local similarity are relevant.

SECTION 2

Topological Data Analysis

Topological Data Analysis (TDA) is a modern approach to the study of complex data using techniques from algebraic topology. At its core, TDA aims to extract and represent topological features of data—such as connected components, loops, and voids—in a way that is robust to noise and suitable for quantitative analysis. Unlike classical statistical or geometric methods, TDA is particularly well-suited for identifying global structural patterns in high-dimensional, incomplete, or noisy datasets. One of its main strengths lies in its ability to provide coordinate-free and scale-independent summaries of data, allowing for a deeper understanding of underlying geometric and topological structures [EH2010b, Oud2015].

This section is devoted to introducing the fundamental tools of TDA that will be employed throughout this thesis. We begin with a formal discussion of *persistent homology*, the principal technique in TDA for capturing topological features across multiple spatial resolutions. Persistent homology is defined in the setting of filtered topological spaces (or simplicial complexes), and we will provide general definitions before focusing on the specific and particularly relevant case of graphs and finite simplicial complexes. In this context, we will illustrate how persistence diagrams are constructed and interpreted [CdSGO2016, CSEH2007, CCSG+2009].

We also introduce metrics on the space of persistence diagrams, namely the *Wasserstein distance* and the *Bottleneck distance* [CGGG+2024, CSEH2007]. These distances quantify the similarity between diagrams and play a crucial role in this work for evaluating how filtering operations applied to signals affect their global topological structure. In particular, they enable us to assess and design filters that minimize topological distortion, thereby preserving significant structural features of the data. As we will show in this section, the Stability Theorem (Theorem 2.15) ensures that it is not necessary to be overly concerned with small variations in these distances: if the input functions are close, then their corresponding persistence diagrams

will also be close.

2.1 Persistent Homology

In this section, we introduce the fundamental concepts of persistent homology. The main references for this material are [EH2010b, Oud2015]. We begin by defining key notions such as filtrations and persistence modules. Subsequently, we present two foundational results that underpin the theory: the interval decomposition theorem and the stability theorem.

The interval decomposition theorem guarantees that any persistence module can be uniquely decomposed into interval modules. This leads to the representation of such modules via persistence diagrams, which succinctly encode topological features across scales [CCSG+2009]. The stability theorem establishes that persistent homology is robust under small perturbations of the input data, making it a reliable tool in applications such as data analysis and machine learning [CSEH2007].

Unless stated otherwise, all homology groups considered in this section are taken with coefficients in the field $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$ of two elements.

Framework

Persistent homology is a method in topological data analysis (TDA) that quantifies the multi-scale topological features of a dataset. It is particularly well-suited for studying the shape of data by tracking the evolution of homological features (such as connected components, cycles, and voids) across a filtration, which is basically a family of indexed topological spaces, as we define next.

Definition 2.1 (Filtration)**.** A of topological spaces is a family $\mathcal{X} = \{X_t\}_{t\in\mathbb{R}}$ such that $X_s \subseteq X_t$ for all $s \leq t \in \mathbb{R}$, and each X_t is a topological space. A filtration is said to be *locally finite* if, for every compact interval $[a, b] \subset \mathbb{R}$, the inclusion maps $X_a \hookrightarrow X_b$ induce homomorphisms on homology vector spaces whose images have finite dimension in each degree (or equivalently, the induced maps have *finite rank*) .

Associated with any filtration is its *persistence module*, an algebraic structure that tracks the evolution of homology across the nested sequence of topological spaces.

Definition 2.2 (Persistence Module)**.** Given a filtration $\{X_t\}_{t\in\mathbb{R}}$, the k-th persistent homology module is the collection $\{H_k(X_t)\}_{t\in\mathbb{R}}$ equipped with a family of linear maps $\{\varphi_{s,t} : H_k(X_s) \to H_k(X_t)\}_{s\leq t}$, where each map $\varphi_{s,t}$ is induced by the inclusion $X_s \hookrightarrow X_t$.

More generally, a *persistence module* $M = \left(\{V_t\}_{t\in\mathbb{R}}, \{\varphi_{s,t} : V_s \to V_t\}_{s\leq t}\right)$ over $\mathbb{R}$ consists of a family of vector spaces V_t indexed by $t \in \mathbb{R}$, together with linear maps $\varphi_{s,t}$ for all $s \leq t$, satisfying the following conditions:

1. $\varphi_{t,t} = \mathrm{id}_{V_t}$ for all $t \in \mathbb{R}$,
2. $\varphi_{r,t} = \varphi_{s,t} \circ \varphi_{r,s}$ for all $r \leq s \leq t$ (functoriality).

Equivalently, a persistence module is a functor $M : (\mathbb{R}, \leq) \to \mathrm{Vec}_{\mathbb{F}}$, where $\mathrm{Vec}_{\mathbb{F}}$ denotes the category of vector spaces over a field $\mathbb{F}$. The k-th persistent homology module is a specific instance in which the persistence module arises from applying homology to a filtration.

Definition 2.3 (Pointwise Finite-Dimensional (p.f.d.) Module)**.** A persistence module $V = \{V_t\}_{t\in\mathbb{R}}$ is said to be *pointwise finite-dimensional* (p.f.d.) if $\dim V_t < \infty$ for all $t \in \mathbb{R}$.

Definition 2.4 (Interval Module)**.** Let $I \subseteq \mathbb{R}$ be an interval. The interval module $\mathbb{I}^I$ is the persistence module defined by

$$\mathbb{I}^I_t = \begin{cases} \mathbb{F} & \text{if } t \in I, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad \varphi_{s,t} = \begin{cases} \mathrm{id}_{\mathbb{F}} & \text{if } s,t \in I \text{ and } s \le t, \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 2.5 (Interval Decomposition)**.** Let V be a pointwise finite-dimensional (p.f.d.) persistence module over $\mathbb{R}$. Then V admits a decomposition as a direct sum of interval modules:

$$V \cong \bigoplus_{i\in\Lambda} \mathbb{I}^{I_i},$$

where Λ is an index set and each $I_i \subseteq \mathbb{R}$ is an interval. This decomposition is unique up to permutation of the summands.

Remark 2.6 . A more general version of the interval decomposition theorem exists under weaker assumptions than pointwise finite-dimensionality. For instance, the result holds for certain classes of *q-tame* persistence modules, where all structure maps have finite-dimensional images; see [CB2015, CdSGO2016] for details. However, for the purposes of this work, the p.f.d. case suffices, as we restrict our attention to persistence modules arising from filtrations of finite graphs and finite cell complexes, which naturally yield pointwise finite-dimensional modules.

The interval decomposition theorem is a foundational result in persistent homology, as it enables the construction of one of the central invariants: the *persistence diagram*. Given a persistence module with decomposition $V \cong \bigoplus_{i\in\Lambda} \mathbb{I}^{I_i}$, its persistence diagram is the collection of intervals $D = \{I_i \mid i \in \Lambda\}$. Since the same interval may appear multiple times in the decomposition, the persistence diagram is formally defined as a *multiset* of intervals. The precise definition is given following the definition of a multiset below.

Definition 2.7 (Multiset)**.** Let A be a set. A *multiset* over A is a function $\mu : A \to \mathbb{N} \cup \{\infty\}$, where $\mu(a)$ denotes the *multiplicity* of the element $a \in A$. The collection of all multisets over A is denoted by

$$\mathcal{M}(A) = \{\mu \mid \mu : A \to \mathbb{N} \cup \{\infty\}\}.$$

A multiset $\mu \in \mathcal{M}(A)$ is said to have *finite multiplicity* if $\mu(a) < \infty$ for all $a \in A$, and to have *finite support* if the set

$$\mathrm{supp}(\mu) = \{a \in A \mid \mu(a) > 0\}$$

is finite. Whenever μ is a multiset, we write $a \in \mu$ to mean $a \in \mathrm{supp}(\mu)$. This is a slight abuse of notation, but it simplifies the exposition.

Convention. In the definition above, we assume that $\mathbb{N}$ includes zero.

Definition 2.8 (Extended Real Line and Plane)**.** Define the *extended real line* as $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$, the *extended plane* as $\overline{\mathbb{R}}^2 := \overline{\mathbb{R}} \times \overline{\mathbb{R}}$, and the diagonal as $\Delta := \{(x, x) \in \overline{\mathbb{R}}^2 \mid x \in \overline{\mathbb{R}}\}$.

Definition 2.9 (Persistence Diagram)**.** Let V be a pointwise finite-dimensional persistence module with interval decomposition $V \cong \bigoplus_{i \in \Lambda} \mathbb{I}^{I_i}$. The *persistence diagram* of V, denoted $\mathrm{Dgm}(V)$, is the multiset of intervals $\{I_i\}_{i \in \Lambda}$. Each interval of the form $[b, d) \subseteq \mathbb{R}$ is represented as a point $(b, d) \in \mathbb{R}^2$, so $\mathrm{Dgm}(V) \in \mathcal{M}(\mathbb{R}^2)$. Following our multiset convention, we write $I \in \mathrm{Dgm}(V)$ whenever the multiplicity of I is positive, that is, $\mathrm{Dgm}(V)(I) > 0$.

Each element $I = (b, d) \in \mathrm{Dgm}(V)$ is called a *persistence interval*, and its length is referred to as its *persistence*:

$$\mathrm{pers}(I) = \mathrm{pers}(b, d) = d - b.$$

Remark: In this work, we consider only intervals of the form $[b, d)$. This convention arises naturally in the setting of a filtration of a cellular complex with finitely many steps: a topological feature is born at time b and persists until time d, meaning it exists at b but has vanished by d. Thus, the feature is active precisely during the interval $[b, d)$, and no other types of intervals are considered. Since we interpret these both as intervals in $\overline{\mathbb{R}}$ and as points in $\overline{\mathbb{R}}^2$, we may use the notations $[b, d)$ and (b, d) interchangeably.

Given a filtration $\mathcal{X} = \{X_t\}_{t \in \mathbb{R}}$, its k-th persistent homology module, as in Definition 2.2, is the persistence module $V = \{H_k(X_t)\}_{t \in \mathbb{R}}$ with structure maps induced by the inclusions $X_s \hookrightarrow X_t$ for $s \leq t$. The *k-persistence diagram* of the filtration, denoted $\mathrm{PD}_k(\mathcal{X})$, is defined as the persistence diagram of this module:

$$\mathrm{PD}_k(\mathcal{X}) := \mathrm{Dgm}\left(\{H_k(X_t)\}_{t \in \mathbb{R}}\right).$$

In other words, $\mathrm{PD}_k(\mathcal{X})$ encodes the multiset of intervals $[b, d)$ representing k-dimensional topological features (such as connected components, cycles, or voids) that appear at scale b, called the *birth*, and disappear at scale d, the *death*, throughout the filtration.

Definition 2.10 . Given a topological space X and a real-valued function $f : X \to \mathbb{R}$, the *sublevel filtration* induced by f is the family of subspaces $\{X_t\}_{t \in \mathbb{R}}$, where each $X_t := f^{-1}((-\infty, t]) \subseteq X$. This defines a filtration of X indexed by $\mathbb{R}$. Whenever we write $\mathrm{PD}_k(f)$, we mean the k-persistence diagram of the sublevel filtration of f, i.e.,

$$\mathrm{PD}_k(f) := \mathrm{PD}_k(\{f^{-1}((-\infty, t])\}_{t \in \mathbb{R}}).$$

Conversely, any filtration $\{X_t\}_{t \in \mathbb{R}}$ of a topological space X, satisfying:

- $X_t \subseteq X$ for all $t \in \mathbb{R}$,
- For every $x \in X$, the set $\{t \in \mathbb{R} \mid x \in X_t\}$ is nonempty and contains a minimum,

can be realized as a sublevel filtration. Define $f : X \to \mathbb{R}$ by

$$f(x) := \min\{t \in \mathbb{R} \mid x \in X_t\}.$$

Then, by construction, $x \in X_{f(x)}$, and $x \notin X_t$ for any $t < f(x)$. It follows that $X_t = f^{-1}((-\infty, t])$, so the original filtration coincides with the sublevel filtration of f. This condition on the existence of a minimum is

satisfied in particular if the filtration is *left-continuous*, meaning that $X_t = \bigcap_{\epsilon>0} X_{t+\epsilon}$ for all $t \in \mathbb{R}$, which in particular is true in the setting of Graphs or Cell complexes.

Computational setting

In most computational settings, persistent homology is computed for a simplicial complex K. In this context, a filtration is typically specified by assigning a real value to each simplex $\sigma \in K$, that is, by defining a function $f : K \to \mathbb{R}$. This function determines the order in which simplices enter the filtration: the filtration at time $t \in \mathbb{R}$ consists of the subcomplex $K_t := \{\sigma \in K \mid f(\sigma) \leq t\}$. To ensure that each K_t is indeed a simplicial complex, the function f must be *monotonic*, meaning that $f(\sigma') \leq f(\sigma)$ whenever $\sigma' \subseteq \sigma$. Thus, a filtration on K is equivalently given by a monotonic function, which induces a total or partial ordering on the simplices consistent with the face relation.

While the function $f : K \to \mathbb{R}$ induces only a partial ordering on the simplices (due to the requirement that $f(\sigma') \leq f(\sigma)$ for all $\sigma' \subseteq \sigma$), in practice we usually refine this to a *total ordering* that is compatible with the face relation. This total order determines a sequence in which simplices are added one at a time, yielding a filtration $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K$, where each K_i is a subcomplex containing the first i simplices in the order. This discrete filtration enables the computation of persistent homology via standard algorithms, such as matrix reduction. In this setting, homological features are said to be *born* at the step where they first appear in the homology of the complex, and *die* when they are no longer non-trivial (when they become boundaries).

Let the simplices of the simplicial complex K be ordered as $\sigma_0, \ldots, \sigma_n$ according to a total ordering compatible with the filtration function $f : K \to \mathbb{R}$. Then, an interval $[a, b) \in \mathrm{PD}_k(f)$ indicates that a k-dimensional homological feature (i.e., a k-cycle) is *born* when the k-simplex σ_i is added to the filtration, with $f(\sigma_i) = a$, and *dies* when the $(k+1)$-simplex σ_j is added, with $f(\sigma_j) = b$. The pair (σ_i, σ_j) is called the *birth-death pair* associated with the persistence interval $[a, b)$. In the case where a feature never dies (i.e., persists indefinitely), the corresponding interval is of the form $[a, \infty)$, and we refer to σ_i as the generator of a persistent class that survives to infinity.

2.2 Distances between persistence diagrams

In a broad sense, the collection of persistence diagrams is defined as

$$\mathcal{D} = \left\{ D \in \mathcal{M}(\overline{\mathbb{R}}^2) \,\middle|\, D(b,d) > 0 \Rightarrow b \leq d \right\},$$

i.e., the set of multisets of points in $\overline{\mathbb{R}}^2$ whose support lies in the extended upper half-plane (including the diagonal), such that every point (b, d) satisfies $b \leq d$.

Definition 2.11 (Persistence Diagrams up to Diagonal Equivalence)**.** We say that two persistence diagrams $D, D' \in \mathcal{D}$ are *equivalent up to the diagonal*, denoted $D \approx D'$, if and only if

$$D(x) = D'(x) \quad \text{for all } x \in \mathbb{R}^2 \setminus \Delta.$$

The quotient space $\mathcal{D}_\Delta := \mathcal{D}/\!\approx$ is called the space of persistence diagrams *up to diagonal equivalence*.

Given $D \in \mathcal{D}$ and $\epsilon \geq 0$, we define the following sub-diagrams (i.e., sub-multisets):

$$\begin{aligned} D_{\geq \epsilon}(b,d) &= \begin{cases} D(b,d), & \text{if } d-b \geq \epsilon, \\ 0, & \text{otherwise} \end{cases} & D_{>\epsilon}(b,d) &= \begin{cases} D(b,d), & \text{if } d-b > \epsilon, \\ 0, & \text{otherwise} \end{cases} \\ D_{\leq \epsilon}(b,d) &= \begin{cases} D(b,d), & \text{if } d-b \leq \epsilon, \\ 0, & \text{otherwise} \end{cases} & D_{<\epsilon}(b,d) &= \begin{cases} D(b,d), & \text{if } d-b < \epsilon, \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{2.1}$$

This notation will be useful later for formulating the filtering problem. In particular, the diagonal equivalence relation can be equivalently expressed as

$$D \approx D' \iff D_{>0} = D'_{>0}.$$

From now on, whenever we refer to persistence diagrams, we mean elements of $\mathcal{D}_\Delta$. For any $D \in \mathcal{D}_\Delta$, there is a representative of the class D with infinite multiplicity across the diagonal Δ, we denote this representative by D_∞, in other words,

$$D_\infty(x) = \begin{cases} D(x), & \text{for } x \in \overline{\mathbb{R}}^2 \setminus \Delta \\ \infty, & \text{for } x \in \Delta \end{cases}$$

Definition 2.12 (Matching between Persistence Diagrams)**.** Let $D, D' \in \mathcal{D}_\Delta$. A *matching* between D and D' is a multiset $\Gamma \in \mathcal{M}(\overline{\mathbb{R}}^2 \times \overline{\mathbb{R}}^2)$ satisfying the following conditions:

1. For each $(x,y) \in \Gamma$, we have $(x,y) \in \operatorname{supp}(D_\infty) \times \operatorname{supp}(D'_\infty)$.
2. For every $x_0, y_0 \in \overline{\mathbb{R}}^2$, the following equalities hold:

$$\sum_{y \in \overline{\mathbb{R}}^2} \Gamma(x_0, y) = D_\infty(x_0), \qquad \sum_{x \in \overline{\mathbb{R}}^2} \Gamma(x, y_0) = D'_\infty(y_0),$$

that is, the number of times x_0 appears as the first coordinate in Γ equals $D_\infty(x_0)$, and the number of times y_0 appears as the second coordinate equals $D'_\infty(y_0)$.

We denote by $\operatorname{Match}(D, D')$ the set of all such matchings between D and D'.

In the following definitions, for $p \in \{1, 2, \ldots, \infty\}$, we extend the ℓ_p-norm from $\mathbb{R}^2$ to $\overline{\mathbb{R}}^2$ using the following convention:

$$\begin{aligned} &\|(-\infty, a) - (-\infty, b)\|_p = |a-b|, && \text{for all } a, b \in \mathbb{R}, \\ &\|(a, +\infty) - (b, +\infty)\|_p = |a-b|, && \text{for all } a, b \in \mathbb{R}, \\ &\|(-\infty, +\infty) - (-\infty, +\infty)\|_p = 0. \end{aligned}$$

Any other combination of intervals involving infinities results in an infinite norm, e.g., $\|(a, +\infty) - (-\infty, b)\|_p = \infty$. The ℓ_2-norm is extended analogously.

Definition 2.13 (Wasserstein and Bottleneck Distances)**.** Let $p \in [1, \infty)$, and let $D, D' \in \mathcal{D}_\Delta$ be two

persistence diagrams. For a matching $\Gamma \in \mathrm{Match}(D, D')$, define its p-cost by

$$\mathrm{cost}_p(\Gamma) := \left(\sum_{(x,y)\in\Gamma} \|x - y\|_2^p \right)^{1/p},$$

and define the *Wasserstein-p* distance between D and D' as

$$W_p(D, D') := \inf_{\Gamma\in\mathrm{Match}(D,D')} \mathrm{cost}_p(\Gamma).$$

Similarly, define the *bottleneck distance* between D and D' as

$$W_\infty(D, D') := \inf_{\Gamma\in\mathrm{Match}(D,D')} \sup_{(x,y)\in\Gamma} \|x - y\|_\infty.$$

Notice that for all $D, D' \in \mathcal{D}_\Delta$ and all $p \in [1, \infty]$, we have

$$W_p(D, D') = W_p(D', D) \geq 0.$$

However, W_p does not define a metric on the entire space $\mathcal{D}_\Delta$. For example, consider a diagram $D \in \mathcal{D}_\Delta$ such that

$$D(0, n) > 0 \quad \text{for all } n \in \mathbb{N}.$$

Then the Wasserstein-p distance between D and the empty diagram $\emptyset$ is infinite:

$$W_p(D, \emptyset) = \infty.$$

This motivates the restriction to diagrams with finite p-moment in the case of $p \in [1, \infty)$, which are exactly those diagrams $D \in \mathcal{D}_\Delta$ for which $W_p(D, \emptyset) < \infty$. On this subspace, W_p defines a genuine metric. In the case $p = \infty$, a similar restriction is necessary (see Chapter 5 in [CdSGO2016] for a detailed treatment).

In our setting, we will be dealing exclusively with persistence diagrams arising from finite graphs and finite cell complexes. These always yield diagrams with finite multiplicity and bounded support. Consequently, the Wasserstein and bottleneck distances as defined above are well-defined and constitute valid metrics for our purposes.

The Stability Theorem

Definition 2.14 (Tame Function)**.** Let X be a triangulable topological space. A function $f : X \to \overline{\mathbb{R}}$ is called *tame* if it satisfies the following two conditions:

1. The sublevel sets $f^{-1}((-\infty, a]) = \{x \in X \mid f(x) \leq a\}$ have finite-dimensional homology over a fixed field for all $a \in \overline{\mathbb{R}}$.
2. There are only finitely many values $a_1 < a_2 < \cdots < a_n \in \overline{\mathbb{R}}$ such that the sublevel set filtration changes homotopy type (i.e., the inclusion-induced maps in homology are not isomorphisms across those levels).

Theorem 2.15 (Stability Theorem for Persistent Homology [CSEH2007])**.** Let $f, g : X \to \overline{\mathbb{R}}$ be two tame functions on a triangulable topological space X. Then for each homological dimension i, the bottleneck distance between their persistence diagrams satisfies

$$W_\infty\big(\mathrm{PD}_i(f), \mathrm{PD}_i(g)\big) \leq \|f - g\|_\infty,$$

where $\|f - g\|_\infty := \sup_{x \in X} |f(x) - g(x)|$.

The stability theorem is one of the foundational results in the theory of persistent homology. It guarantees that small perturbations in the input function—measured in the sup norm—result in small perturbations in the output persistence diagram—measured in the bottleneck distance. This robustness to noise and small deformations makes persistent homology a powerful tool in topological data analysis, especially when working with empirical or discretized data arising from real-world measurements.

The significance of this result lies in its ability to provide continuity of the topological summary (the persistence diagram) with respect to perturbations in the underlying data. In practical applications, one rarely has access to exact data: measurements are subject to noise, discretization, and sampling errors. The stability theorem ensures that persistent homology can still extract meaningful and reliable topological features under such conditions.

Moreover, the theorem establishes a rigorous link between geometric perturbations and the algebraic output of persistent homology. This connection is crucial in validating the use of persistence diagrams as stable topological signatures, allowing them to be used effectively in downstream tasks such as clustering, classification, and statistical inference.

This theorem also suggests that small perturbations in the data often yield persistence intervals that lie close to the diagonal. Consequently, features with low persistence are frequently interpreted as noise [CSEH2007]. In this work, we seek to explore the converse: are all low-persistence features necessarily the result of small data perturbations? If so, can these perturbations be effectively filtered out?

Throughout the remainder of this paper, we use the terms *small persistence features* and *topological noise* interchangeably to describe such phenomena. In the following section, we show that spectral filtering does not reliably remove topological noise, thereby motivating the development of alternative approaches.

SECTION 3

Traditional methods fail to preserve topological information

In this section, we motivate the development of an alternative class of filters by demonstrating that spectral filters are inadequate for suppressing low-persistence features in two-dimensional cell complexes. Specifically, we analyze the effect of conventional low-pass filters on the topology of signals defined over the vertices of such complexes. To make this analysis concrete, we present a simple illustrative example based on an image, where each pixel is treated as a node in a grid graph, and the squares formed between adjacent nodes define the 2-cells—thus forming a cubical complex (see Figure 3.1). Throughout this discussion, we will refer to low-persistence features as *topological noise*. We show that spectral filters not only fail to eliminate this topological noise, but can also significantly distort high-persistence features, which correspond to globally relevant topological structures.

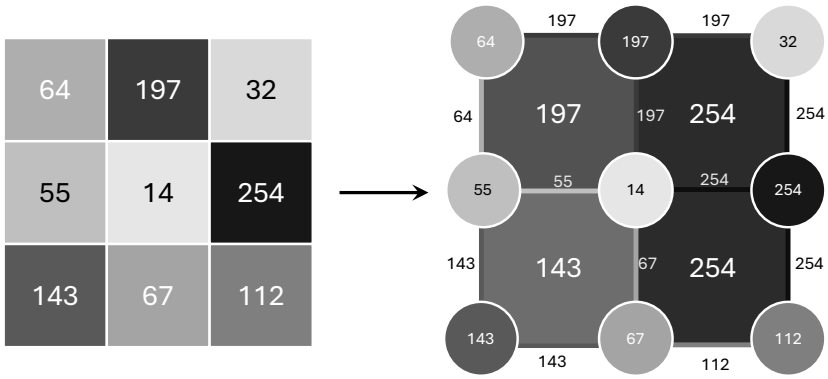


Figure 3.1: A grayscale image interpreted as a regular cell complex. Each pixel is represented as a node with an associated signal value f, corresponding to the pixel intensity. Higher-dimensional cells (edges and squares) are assigned values given by the maximum value of their constituent nodes. This structure allows us to treat the image as a topological space with a piecewise constant signal, suitable for topological data analysis.

Since our particular example of a graph is derived from an image, we can also examine how standard image denoising techniques perform in removing topological noise. Traditional methods such as Gaussian smoothing or total variation minimization are designed to eliminate pixel-level fluctuations while preserving perceptual features like edges. However, these techniques do not explicitly account for the topological structure of the underlying signal and often fail to suppress irrelevant topological features that appear as low-persistence classes in the sublevel filtration of the persistent homology. Moreover, we will demonstrate that such methods can also degrade high-persistence features, thus compromising the integrity of global topological information. This reinforces the need for filters that are sensitive to topological characteristics when denoising signals defined on cell complexes.

3.1 Spectral Filters

All experiments presented in this section are publicly available in the following GitHub repository maintained by the author: `github.com/mvlier/Persistence_Based_Methods`.

In what follows, we apply spectral filters to a 24×24 grayscale image, where all pixel values are normalized to the interval $[0, 1]$. In our analysis, we define topological noise as features with persistence less than a chosen threshold $\varepsilon = 0.1$ (10% of the possible interval). This threshold reflects the assumption that features with persistence below ε are negligible. Consequently, if two images f and g—viewed as functions on the vertices of the graph—satisfy $\|f - g\|_\infty < \varepsilon$, then g can be considered a good approximation of f, and we call it an *ε-approximation*.

Remark 3.1 . Although it is common in many signal processing applications to use the ℓ_2 norm to assess approximation quality—owing to its connection with energy and variance—we instead adopt the ℓ_∞ norm in our framework. The ℓ_2 norm measures the average deviation between signals and may mask significant local errors if they are limited in spatial extent. In contrast, the ℓ_∞ norm captures the maximum pointwise deviation, providing uniform control over the approximation error across the entire domain. This is particularly valuable when targeting the removal of low-amplitude noise, where small but widespread perturbations may not significantly affect the ℓ_2 norm but can alter important structural or topological features. Moreover, the ℓ_∞ norm naturally supports the notion of strict ε-approximations—ensuring that all values in the filtered signal stay within an ε-tube of the original—making it especially suitable in contexts where topological stability or

uniform fidelity is desired.

Despite the relatively low resolution of the image, the impact of low-pass filtering on the topological structure of the signal is clearly observable. As we will show, spectral filtering not only fails to consistently remove topological noise, but also alters high-persistence features, thereby affecting the meaningful global topology of the signal.

Throughout our discussion of spectral filtering, we derive the spectrum from four distinct versions of the graph Laplacian: the unnormalized Laplacian with uniform weights (Definition 1.5), the unnormalized Laplacian with Gaussian weights (Equation (1.3)), the normalized Laplacian with uniform weights (Equation (1.1)), and the normalized Laplacian with Gaussian weights (Equation (1.4)). As we will demonstrate, the most effective results in filtering topological noise are obtained when using the spectrum of the unnormalized Laplacian with Gaussian weights.

Low-pass filters

In Figure 3.2, we illustrate the effect of applying the heat kernel to our example grayscale image. The leftmost panel shows the original image (without filtering), and the subsequent panels display the results of applying the heat kernel with increasing values of the scale parameter τ, specifically $\tau = 1, 10,$ and 100. As expected, larger values of τ lead to progressively smoother images with reduced high-frequency content, visually manifested as diminished noise and fewer oscillations.

However, the corresponding persistence diagrams (shown below each image) reveal that the heat kernel impacts topological features across the entire persistence spectrum. Notably, features with persistence below the threshold $\epsilon = 0.1$ persist across all values of τ, indicating that the filter does not reliably eliminate low-persistence features—that is, topological noise. Furthermore, the filter also attenuates high-persistence features, which are typically associated with meaningful global structures. This limitation is evident in the persistence diagrams and is further quantified by the bottleneck distance: any distance exceeding $\epsilon/2 = 0.05$ shows perturbations in features with persistence greater than $\epsilon = 0.1$, confirming that the filter affects more than just noise.

To perform a more quantitative analysis of the effect of low-pass filters on topological noise, in Section 3.1 we show the result of applying two different low-pass filters with varying parameters. For each filter, we present two plots: the top plot shows how the number of topological features with persistence lower than $\varepsilon = 0.1$—our measure of topological noise—and the number of relevant topological features change as the filter parameter varies; the bottom plot shows how the same parameter affects the $\|\cdot\|_\infty$ distance between the filtered and original images, as well as the bottleneck distance between their persistence diagrams. This allows us to assess both the topological and signal-level distortions introduced by each filtering method. In what follows, we provide a more detailed explanation of the two filters, and of other filters for which we conducted the same analysis.

Remark 3.2 . As expected from the stability theorem (see Theorem 2.15), we observe in all plots of Section 3.1 that the inequality

$$\|f - \tilde{f}\|_\infty \leq W_\infty(\mathrm{PD}_*(f), \mathrm{PD}_*(\tilde{f}))$$

holds throughout.

As mentioned earlier, each filter listed below can be computed using the Fourier transform derived from

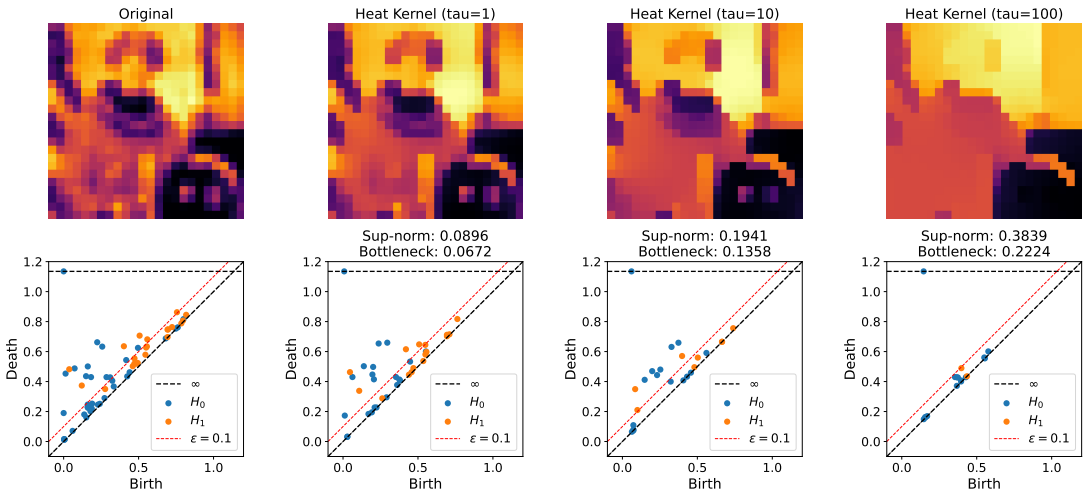
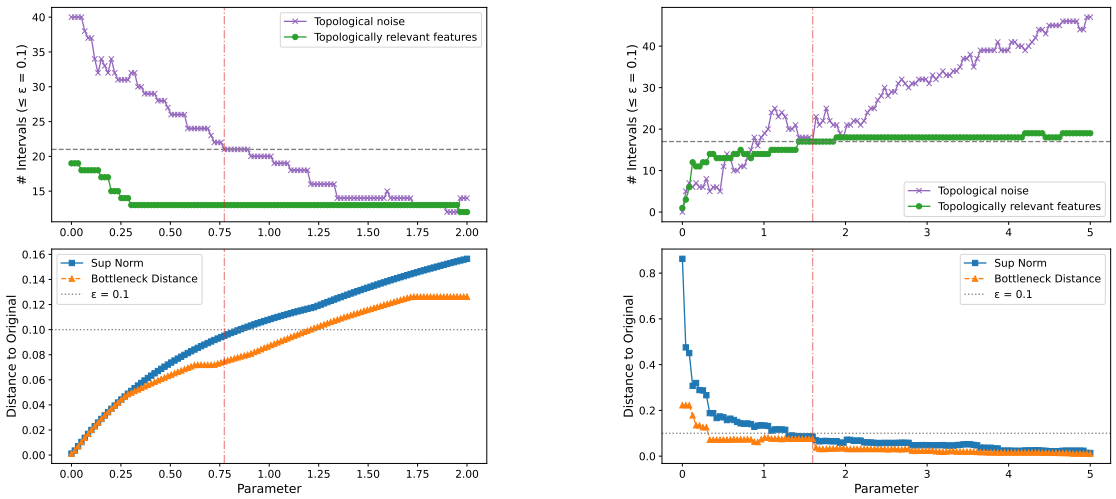


Figure 3.2: Illustrative example using a simple natural image to show the effect of the heat kernel (a low-pass filter) on the persistence diagram as the scale parameter τ increases. The left-most panel shows the original image, followed by filtered versions with increasing values of τ, specifically $\tau = 1$, $\tau = 10$, and $\tau = 100$. The filtered images were derived from the spectrum of the unnormalized Laplacian with Gaussian weights.



(a) Heat Kernel

(b) Ideal low pass

Figure 3.3: Topological noise behavior across 2 different filters. Each subplot presents: (top) the topological noise (purple) and topologically relevant features (green) plotted against the corresponding filter parameter (e.g., scale, cutoff); (bottom) the sup-norm (blue) and bottleneck distance (orange) as functions of the same parameter. Both filters are computed with the non normalized Laplacian with Gaussian weights.

different versions of the graph Laplacian. In this section, we focus on presenting plots corresponding to the unnormalized Laplacian with Gaussian weights, as it yields the most compelling results. Plots for the other Laplacian variants are available in the repository referenced at the beginning of this section.

Ideal low-pass filter. This filter is defined by the transfer function

$$h(\lambda) = \begin{cases} 1 & \text{if } \lambda \leq \lambda_c, \\ 0 & \text{otherwise,} \end{cases}$$

which is parameterized by the cut-off frequency λ_c. Only the spectral components corresponding to eigenvalues $\lambda \leq \lambda_c$ are retained in the filtered signal. One of the main advantages of this filter is its conceptual simplicity and perfect frequency selectivity, allowing precise control over the retained spectral content.

Increasing λ_c allows more high-frequency components to pass through, resulting in less aggressive filtering. Consequently, in Figure 3.3b, we observe that as λ_c increases, the filtered signal $\tilde{f}$ becomes closer to the original signal f, and the sup norm $\|f - \tilde{f}\|_\infty$ correspondingly decreases. Notably, when the sup norm falls below the chosen threshold $\varepsilon = 0.1$—which occurs at approximately $\lambda_c = 1.6$—the number of low-persistence features in the filtered image remains substantial, with roughly 20 such features still present (about half the number in the original image). This demonstrates that even the ideal low-pass filter, when constrained to remain within an ε-approximation of the original signal, does not fully eliminate topological noise. Moreover, even when this constraint is relaxed, topological noise is only entirely removed when the filtered signal reduces to the constant eigenfunction associated with $\lambda = 0$.

It is also important to note that reducing the cutoff λ_c quickly increases the bottleneck distance to values exceeding $\varepsilon/2 = 0.05$. This is a critical observation: if only low-persistence features (those with persistence below $\varepsilon = 0.1$) were being affected—specifically, by being eliminated—the maximum bottleneck distance we would expect to observe is exactly $\varepsilon/2 = 0.05$, corresponding to the cost of matching a persistence interval of length ε to the diagonal. Therefore, observing bottleneck distances larger than this threshold provides clear evidence that higher-persistence features are also being altered by the filter.

Heat kernel. As discussed in Section 1.1, the heat kernel is defined by the transfer function $h(\lambda) = e^{-\tau\lambda}$, which acts as a continuous low-pass filter. Increasing the scale parameter τ results in more aggressive attenuation of higher frequencies, effectively allowing only lower-frequency components to pass. This behavior is analogous to decreasing the cutoff frequency λ_c in an ideal low-pass filter. Conversely, reducing τ slows the decay of $h(\lambda)$, thereby allowing more high-frequency components to pass. In this sense, τ plays a role similar to an inverse cutoff parameter, modulating the degree of smoothing applied to the signal.

Advantage: The heat kernel filter is smooth and analytically tractable, making it a popular choice in graph signal processing. Its exponential decay ensures numerical stability and efficient suppression of high-frequency noise, while preserving a controllable amount of low-frequency structure.

In Figure 3.3a, we observe that as the scale parameter τ increases, both the sup norm and the bottleneck distance increase, while the topological noise decreases. Similar to the ideal low-pass filter, the heat kernel does not produce an ε-approximation of the original function that is free of topological noise. In fact, its best performance in this regard is comparable to that of the ideal filter, reducing the number of low-persistence features to roughly 20.

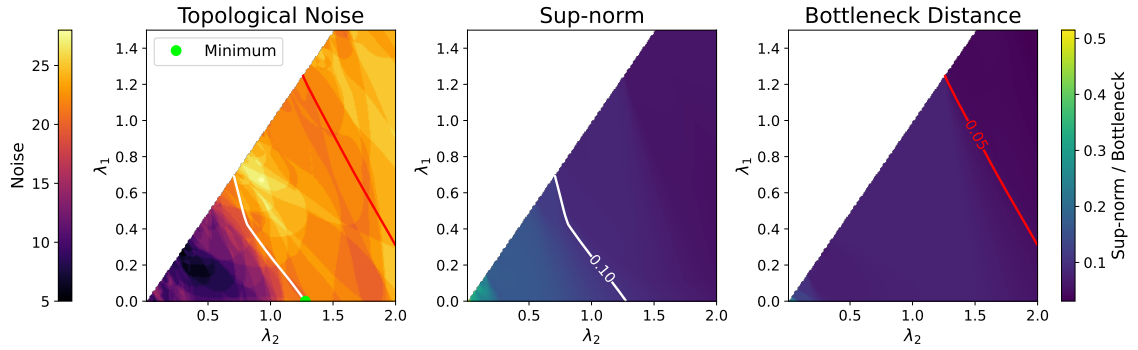


Figure 3.4: Raised Cosine filter applied to the image of Figure 3.2. All plots are shown over the parameter domain (λ_1, λ_2). The left plot displays the topological noise, the middle plot shows the sup-norm, and the right plot presents the bottleneck distance. In the left plot, the green marker indicates the point of minimum topological noise under the constraint that the sup-norm does not exceed $\varepsilon = 0.1$.

Tikhonov filter. The Tikhonov filter is defined by the transfer function $h(\lambda) = \frac{1}{1+\alpha\lambda}$, where $\alpha > 0$ is a regularization parameter. This filter originates from Tikhonov regularization and promotes smoothness by attenuating high-frequency components.

Power law filter. The power law filter employs the transfer function $h(\lambda) = (1+\lambda)^{-\gamma}$ for $\gamma > 0$, attenuating higher frequencies more as γ increases.

Lorentzian filter. The Lorentzian filter is defined by the transfer function $h(\lambda) = \frac{1}{1+(\beta\lambda)^2}$, where $\beta > 0$ controls the sharpness of the attenuation. This filter resembles the frequency response of the Cauchy (or Lorentzian) distribution and provides a smooth, non-exponential decay of spectral components.

Raised Cosine. This filter is defined by the transfer function

$$h_{\cos}(\lambda) = \begin{cases} 1 & \text{if } \lambda \le \lambda_1, \\ \frac{1}{2}\left[1 + \cos\left(\pi \cdot \frac{\lambda-\lambda_1}{\lambda_2-\lambda_1}\right)\right] & \text{if } \lambda_1 < \lambda < \lambda_2, \\ 0 & \text{if } \lambda \ge \lambda_2, \end{cases}$$

where $\lambda_1 < \lambda_2$ define the passband and transition band of the filter. The transfer function provides a smooth transition from full transmission to complete attenuation over the interval $[\lambda_1, \lambda_2]$, forming a raised cosine profile.

In our analysis, the Tikhonov, Power Law and Lorentzian filters exhibit the same general trend as the first two filters. Additionally, for the Raised Cosine filter, which depends on two parameters λ_1 and λ_2, we present in Figure 3.4 the plots of topological noise, sup-norm, and bottleneck distance as functions of these parameters. In the left-most plot, we observe that the minimum topological noise, under the constraint that the sup-norm remains below $\varepsilon = 0.1$, occurs very close to the boundary where the norm is exactly 0.1. This behavior is expected, as we also observe that beyond this threshold—i.e., in the region where the sup-norm exceeds 0.1—the topological noise continues to decrease, but never reaches zero.

From the preceding analysis, we draw the following conclusion: spectral low-pass filters are indeed effective in reducing topological noise, but they do not eliminate it entirely while remaining within the framework of ε-approximations. Even when this constraint is relaxed, increasing the attenuation of high-frequency

components does result in a further reduction of topological noise. However, the number of low-persistence features only drops to zero when the filtered signal converges to the constant eigenfunction associated with the eigenvalue $\lambda = 0$. Thus, although there exists a clear correlation between spectral filtering and the reduction of topological noise, the relationship is not particularly strong. Furthermore, while lighter filtering tends to better preserve high-persistence features, this preservation is generally not maintained in the ε-approximations produced by these filters, as we can see from the drop of the "topologically relevant features" in the plots.

In Table 3.1, we summarize the performance of all applied filters, reporting the best achievable reduction in topological noise under the constraint of ε-approximations. For each filter, we provide the sup-norm $\|f - \tilde{f}\|_\infty$ between the original and filtered images, denoted f and $\tilde{f}$, respectively. In all cases, we observe that the minimum level of topological noise is attained when the sup-norm is close to $\varepsilon = 0.1$. Furthermore, we note that the number of topologically significant features removed is never zero, indicating that some high-persistence features are consistently affected by the filtering process

Table 3.1: Comparison of spectral low-pass filters minimizing topological noise under a sup-norm constraint $\varepsilon = 0.1$, using unnormalized Laplacian with Gaussian weights. From left to right we have columns Filter, Minimal Topological Noise, Topologically relevant features killed (Top. Killed), Sup-norm and Best Parameter.

Filter	$h(\lambda)$	**Min. Top. Noise**	**Top. Killed**	**Sup-norm**	**Best Param**
Heat Kernel	$e^{-\tau\lambda}$	21	13	0.0952	$\tau = 0.773$
Ideal Low-pass	$\mathbb{1}_{\lambda\leq\lambda_c}$	17	17	0.0854	$\lambda_c = 1.597$
Tikhonov	$\frac{1}{1+\alpha\lambda}$	25	13	0.0982	$\alpha = 1.277$
Lorentzian	$\frac{1}{1+(\beta\lambda)^2}$	20	13	0.0902	$\beta = 1.008$
Power Filter	$\frac{1}{(1+\lambda)^\gamma}$	23	13	0.0982	$\gamma = 1.210$
Raised Cosine	$h_{\cos}(\lambda)$	17	2	0.0867	$\lambda_1 = 1.35,\ \lambda_2 = 1.55$

In Tables 3.2 to 3.4, we present the results for the unnormalized Laplacian with uniform weights, the normalized Laplacian with uniform weights, and the normalized Laplacian with Gaussian weights, respectively. Compared to the unnormalized Laplacian with Gaussian weights, these variants perform worse in removing topological noise while preserving topologically relevant features. This further underscores the limitations of spectral filters in effectively addressing topological noise.

Table 3.2: Comparison of spectral low-pass filters minimizing topological noise under a sup-norm constraint $\varepsilon = 0.1$, using unnormalized Laplacian with uniform weights. From left to right we have columns Filter, Minimal Topological Noise, Topologically relevant features killed (Top. Killed), Sup-norm and Best Parameter.

Filter	$h(\lambda)$	**Min. Top. Noise**	**Top. Killed**	**Sup-norm**	**Best Param**
Heat Kernel	$e^{-\tau\lambda}$	37	19	0.0574	$\tau = 0.053$
Ideal Low-pass	$\mathbb{1}_{\lambda\leq\lambda_c}$	40	19	0.0003	$\lambda_c = 7.94$
Tikhonov	$\frac{1}{1+\alpha\lambda}$	37	19	0.0552	$\alpha = 0.055$
Lorentzian	$\frac{1}{1+(\beta\lambda)^2}$	37	17	0.0977	$\beta = 0.192$
Power Filter	$\frac{1}{(1+\lambda)^\gamma}$	36	18	0.0661	$\gamma = 0.134$
Raised Cosine	$h_{\cos}(\lambda)$	37	2	0.0983	$\lambda_1 = 0.0,\ \lambda_2 = 9.52$

Table 3.3: Comparison of spectral low-pass filters minimizing topological noise under a sup-norm constraint $\varepsilon = 0.1$, using normalized Laplacian with uniform weights. From left to right we have columns Filter, Minimal Topological Noise, Topologically relevant features killed (Top. Killed), Sup-norm and Best Parameter.

Filter	$h(\lambda)$	Min. Top. Noise	Top. Killed	Sup-norm	Best Param
Heat Kernel	$e^{-\tau\lambda}$	37	17	0.0844	$\tau = 0.328$
Ideal Low-pass	$\mathbb{1}_{\lambda\leq\lambda_c}$	40	19	6.7×10^{-8}	$\lambda_c = 1.996$
Tikhonov	$\frac{1}{1+\alpha\lambda}$	36	17	0.0901	$\alpha = 0.412$
Lorentzian	$\frac{1}{1+(\beta\lambda)^2}$	39	17	0.0908	$\beta = 0.706$
Power Filter	$\frac{1}{(1+\lambda)^\gamma}$	36	18	0.0909	$\gamma = 0.487$
Raised Cosine	$h_{\cos}(\lambda)$	38	1	0.0987	$\lambda_1 = 0.0,\ \lambda_2 = 2.58$

Table 3.4: Comparison of spectral low-pass filters minimizing topological noise under a sup-norm constraint $\varepsilon = 0.1$, using normalized Laplacian with Gaussian weights. From left to right we have columns Filter, Minimal Topological Noise, Topologically relevant features killed (Top. Killed), Sup-norm and Best Parameter.

Filter	$h(\lambda)$	Min. Top. Noise	Top. Killed	Sup-norm	Best Param
Heat Kernel	$e^{-\tau\lambda}$	34	18	0.0952	$\tau = 0.277$
Ideal Low-pass	$\mathbb{1}_{\lambda\leq\lambda_c}$	43	19	6.8×10^{-4}	$\lambda_c = 2.017$
Tikhonov	$\frac{1}{1+\alpha\lambda}$	35	17	0.0970	$\alpha = 0.328$
Lorentzian	$\frac{1}{1+(\beta\lambda)^2}$	32	18	0.0992	$\beta = 0.580$
Power Filter	$\frac{1}{(1+\lambda)^\gamma}$	36	17	0.0987	$\gamma = 0.416$
Raised Cosine	$h_{\cos}(\lambda)$	32	1	0.0893	$\lambda_1 = 0.0,\ \lambda_2 = 3.16$

Band- and high-pass filters

All cases explored until here aim to achieve the desired result using a low-pass filter. High-pass filters are generally unsuitable for filtering topological noise, as they allow high-frequency components to influence the resulting signal. For similar reasons, there is little justification for using band-pass filters in this context. To illustrate the inefficacy of these alternatives, we present some results below:

When filtering only the constant component ($\lambda = 0$), as expected, there is no effect on the topological noise, yielding $\|f - \tilde{f}\|_\infty \approx 0.44$. For any high-pass filter beyond this, the sup-norm consistently remains around 0.8 or higher, and the topological noise increases until all components are filtered out and the function becomes constant (see Figure 3.5).

Band-pass filters exhibit similar behavior. When $\lambda = 0$ lies outside the pass-band, the sup-norm exceeds 0.1, indicating that such filters are ineffective at suppressing topological noise while remaining faithful to the original filter.

A more promising alternative to a standard band-pass filter is one that combines a low-pass and a band-pass filter, with pass-bands in the intervals $[0, a]$ and $[b, c]$. This corresponds to the composite filter:

$$h(\lambda) = \begin{cases} 1 & \text{if } \lambda \in [0, a] \cup [b, c], \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

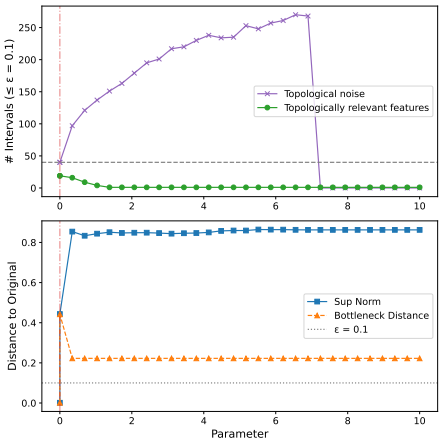


Figure 3.5: Topological noise behaviour for high-pass filter.

We computed this filter for various values of the triple (a, b, c). The dependence of the topological noise on these values is illustrated in Figure 3.6a. Additionally, Figure 3.6b shows the cases where the filtered image $\tilde{f}$ satisfies $\|f - \tilde{f}\|_\infty < \varepsilon = 0.1$, indicating a close approximation to the original signal. In this case, the minimum observed topological noise is 17, which is very close to the best result obtained using low-pass filters alone in Table 3.1. Even when considering all instances, without the restriction $\|f - \tilde{f}\|_\infty < \varepsilon = 0.1$, the topological noise remains no less than 7.

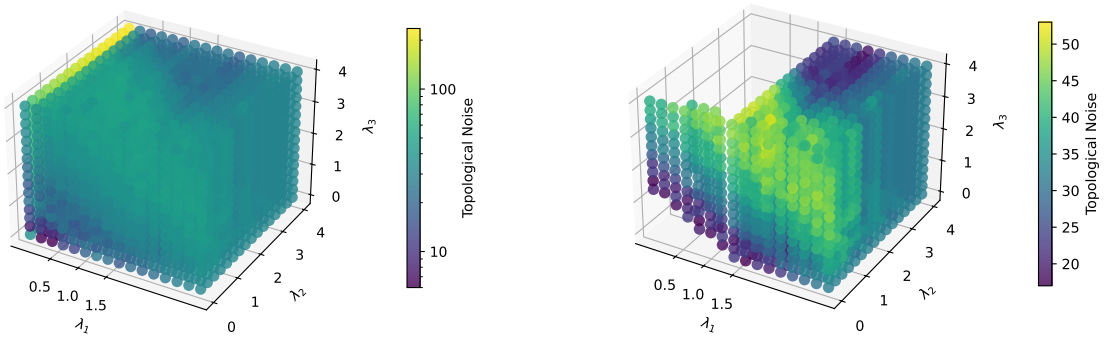
In Table 3.5, we summarize these results. Note that the optimal solution for the band-pass filter effectively reduces to a low-pass filter, as the lower bound of the pass-band is zero.

Table 3.5: Comparison of band-pass, high-pass, and composite spectral filters minimizing topological noise under a sup-norm constraint $\varepsilon = 0.1$, using unnormalized Laplacian with Gaussian weights. From left to right we have columns Filter, Minimal Topological Noise, Topologically relevant features killed (Top. Killed), Sup-norm and Best Parameter.

Filter	Min. Top. Noise	Top. Killed	Sup-norm	Best Param
Ideal Composite	17	2	0.0854	$(\lambda_1 = 10^{-4},\ \lambda_2 = 0,\ \lambda_3 = 1.6)$
Ideal BPF	18	2	0.0849	$(\lambda_1 = 0,\ \lambda_2 = 1.61)$
Ideal HPF	40	0	0.00118	$\lambda_c = 0$

With these results, we conclude the demonstration that although spectral filters—particularly low-pass filters—tend to reduce the topological noise of a signal, they do not achieve its complete elimination (except in the trivial case where only the constant eigenfunction remains). Moreover, they lack the ability to selectively remove only the noise, as variations in higher-persistence elements are also affected.

In the general context of Graph Signal Processing, we have explored the main types of spectral filters. However, given that our particular example is an image (represented as a grid graph), we can also consider conventional image denoising methods and examine their impact on topological noise. These methods typically do not generalize to arbitrary graphs, as they often rely on the grid structure's inherent shift invariance.



(a) All instances **(b)** $\|f - \tilde{f}\|_\infty < 0.1$

Figure 3.6: Topological noise analysis in the case of a composite spectral filter.

3.2 Denoising Methods

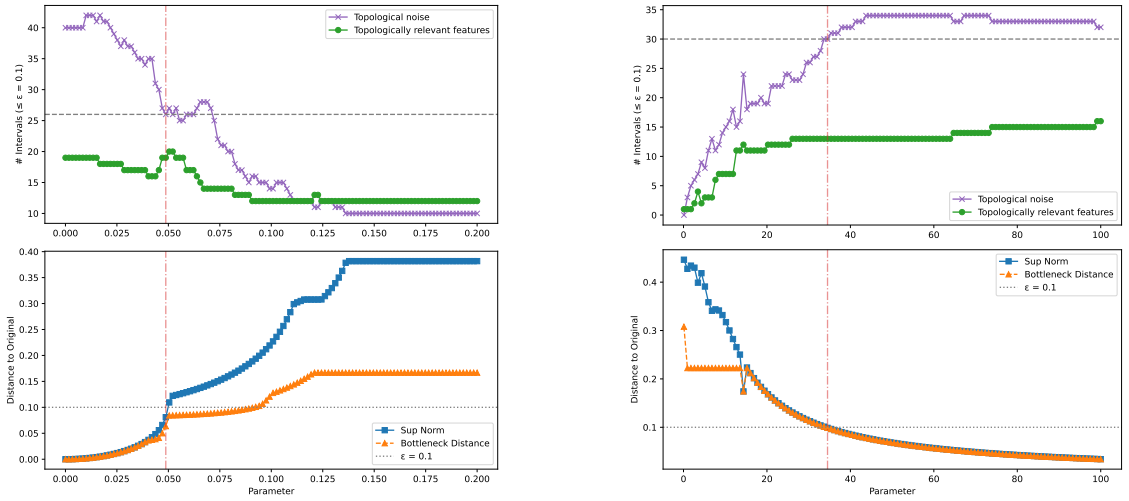
We evaluate a diverse set of classical and modern denoising methods ([GW2018]), each parameterized by a single variable, and assess their impact on topological noise. The filters tested include Gaussian Blur, Median Filter, Total Variation (TV), Non-Local Means (NLM), Wiener Filter, Wavelet Denoising, and Wavelet Thresholding.

As in the analysis conducted for low-pass filters, each denoising method is evaluated across a range of parameter values. Performance is assessed using the four metrics as before: (i) the number of persistence intervals with lifetime at most ε (topological noise), (ii) the number of intervals with liftime more than ε (topologically relevant features) (iii) the sup-norm distance to the original image, and (iv) the bottleneck distance between persistence diagrams. Figure 3.7 presents the corresponding plots for the wavelet and TV denoising methods, while Table 3.6 summarizes the results by reporting the minimum topological noise observed within the context of ε-approximations of the original signal for all the methods mentioned abouve (the remaining plots are in the repository mentioned in the beginning of this section).

These results indicate that standard image denoising techniques are generally ineffective at fully eliminating topological noise. While some methods reduce the amount of low-persistence features, a significant portion typically remains.

Table 3.6: Comparison of DSP-based denoising methods minimizing topological noise under a sup-norm constraint $\varepsilon = 0.1$.

Method	Min. Top. Noise	Top. Killed	Sup-norm	Best Param
Wavelet	26	0	0.0761	$\theta = 0.0483$
TV	30	6	0.0988	$\lambda = 34.52$
Wavelet (raw)	32	5	0.0672	$\theta = 0.0269$
Bilateral	34	1	0.0273	$d = 5.462$
NLM	39	0	0.0140	$h = 0.298$
Guided	39	0	0.0200	$\epsilon = 0.00135$
Wiener	40	0	4.23×10^{-6}	–
Gaussian	37	0	0.0579	$\sigma = 0.417$



(a) Wavelet **(b)** TV

Figure 3.7: Topological noise behavior across two different filters. Each subplot presents: (top) the topological noise (purple) and topologically relevant features (green) plotted against the corresponding denoising method parameter; (bottom) the sup-norm (blue) and bottleneck distance (orange) as functions of the same parameter.

3.3 Deep Learning Methods

Many current image denoising methods are based on deep learning models, most of which are specifically trained to remove Gaussian noise. In Table 3.7, we evaluate the extent to which these models reduce topological noise in our images. For this analysis, we selected three state-of-the-art methods with publicly available pre-trained models from their official repositories: DnCNN [ZZC+2017,ZLZ+2020], a convolutional neural network designed for image denoising via residual learning; SwinIR [LCS+2021], a model based on the Swin Transformer architecture tailored for image restoration tasks; and KBNet [ZLS+2023], a hybrid network that integrates convolutional and attention mechanisms within a feedback architecture.

The results in Table 3.7 indicate that, even among deep learning-based denoising models, none fully eliminate the topological noise. Furthermore, we can see from the sup-norm values that only SwinIR (with $\sigma = 15$) achieves an ε-approximation of the original function, given our selected threshold of $\varepsilon = 0.1$.

Table 3.7: Comparison of Deep Learning denoising methods minimizing topological noise without the sup-norm constraint $\varepsilon = 0.1$.

Filter	Top. Noise	Sup-norm	Bottleneck
DnCNN ($\sigma = 25$)	24	0.196	0.177
SwinIR ($\sigma = 15$)	26	0.094	0.062
SwinIR ($\sigma = 25$)	23	0.204	0.113
SwinIR ($\sigma = 50$)	17	0.391	0.222
KBNet ($\sigma = 15$)	35	0.105	0.064
KBNet ($\sigma = 25$)	27	0.185	0.154

Conclusion

The analysis of various spectral filters presented in this section demonstrates that, although spectral filtering is often highly effective in practice and can reduce topological noise—often in proportion to the strength of the filtering applied—these methods generally fall short of completely eliminating topological noise or preserving topologically significant features with high persistence. This limitation arises from the fact that spectral filters operate solely in the frequency domain, without explicitly accounting for the persistence or relevance of topological features. Consequently, while such filters can effectively smooth signals and isolate specific frequency bands, they are inherently incapable of selectively targeting and suppressing features based on their topological importance.

To address this shortcoming, an alternative approach is required—one that explicitly incorporates topological information into the filtering process. Specifically, we aim to design filters that can attenuate features associated with low-persistence topological classes. This motivation leads to the formulation of a formal problem statement that captures the objective of removing low-persistence structures while preserving the essential characteristics of the original signal.

In the following sections, we present a rigorous formulation of this problem and develop a framework for constructing filters that explicitly target low-persistence features. In subsequent chapters, we introduce an algorithmic solution and evaluate its performance in comparison to conventional spectral filtering techniques.

SECTION 4

Graphs with faces

Before addressing how to formalize the problem discussed in the previous section, we introduce a more general setting in which to work. Rather than restricting ourselves to regular grid-shaped cell complexes, as in Figure 3.1, we consider a broader class of structures that we refer to as *graphs with faces*. This notion generalizes the concept of a 2-cell embedding of a graph, providing a flexible framework for defining topological and geometric operations on discrete domains.

Several notions of *"faces"* on graphs have been proposed in the literature. The most classical is that of a simplicial complex [], which is quite restrictive, as it requires the underlying graph to be simple and composed exclusively of triangular faces. Cubical complexes [] offer an alternative, but similarly impose rigid structural constraints.

A more flexible framework is the concept of a *2-cell embedding*. A **2-cell embedding** of a planar graph G is a planar embedding $\Phi : G \hookrightarrow S^2$ into the 2-sphere such that every connected component of the complement $S^2 \setminus \Phi(G)$ is homeomorphic to an open disk. These components are called the *faces* or *2-cells*. While more general than simplicial or cubical complexes, this notion remains somewhat restrictive, as it requires all faces to share the same disk-like topology.

Several related concepts have been used in the Graph Signal Processing (GSP) literature. Notably, simplicial complexes, cubical complexes, and cellular sheaf models have been explored to extend classical signal processing tools to higher-order domains. For example, the use of simplicial complexes in [SSF+2022, BS2020] and the definition of combinatorial Laplacians on cellular complexes [HJ2013, Lim2020] provide foundations for analyzing signals beyond the vertex and edge level. Our notion of a graph with faces aligns with this

broader trend of using topological structures to enhance the expressive power of signal representations on discrete spaces.

Another generalization is provided by the Planar Hollow Cell Complex (PHCC) [SB2024, §II, Definition 3]. In this framework, faces are defined as regions homeomorphic to an open disk from which a finite number of pairwise non-overlapping closed disks have been removed. Additionally, the boundary of the outer region must be homeomorphic to S^1.

Motivated by the limitations of these existing definitions, we introduce the notion of faces on graphs. While the PHCC framework is highly general, it is defined via a somewhat verbose list of five conditions. The 2-cell embedding, in contrast, is intuitive and simple, but too restrictive. Our proposed definition strikes a balance between these two extremes: it retains the simplicity of the 2-cell embedding while encompassing the generality of PHCCs, and even extends beyond them.

Since our definition is closely related to the concept of a 2-cell embedding, we begin by presenting its formal definition.

Definition 4.1 (2-cell embedding)**.** A *2-cell embedding* of a graph $G = (V, E)$ on a surface Σ is a drawing of G on Σ such that:

- Vertices are represented as distinct points on Σ;
- Edges are represented as simple, non-intersecting curves connecting the corresponding vertex points, with the only allowed intersections being at shared endpoints;
- The connected components of the complement $\Sigma \setminus G$, called *faces*, are homeomorphic to open disks.

Each face defines a 2-cell in a CW-complex structure on Σ, thereby endowing the embedded graph with the topological structure of a cellular embedding.

In our framework, a planar *graph with faces* is a generalization of a 2-cell embedding. One can think of it as extending the standard notion of a 2-cell embedding by allowing faces to be arbitrary connected components, rather than requiring them to be homeomorphic to open disks. While this generalization sacrifices the property that each face is an open disk, it enables us to work within a broader class of graphs, where edges represent pairwise relations and faces can be interpreted as higher-order, multi-way relations.

In what follows, we first introduce a general definition of a graph with faces, and then specialize this definition to the planar setting, which will be the primary focus of our study.

Definition 4.2 . A **face** of a graph $G = (V, E)$ is a finite sequence of (possibly repeated) edges,

$$\tau = e_1 \dots e_N, \quad e_i \in E,$$

such that

$$\sum_{i=1}^{N} \partial_1(e_i) = 0,$$

where ∂_1 is the boundary map defined by $\partial_1(uv) = u + v$ for $uv \in E$, and the coefficients are taken in the field $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$. That is, $\sum_{i=1}^{N} e_i$ forms a 1-cycle. Notice that we are defining a face by saying what its boundary is.

Note: Working over the field $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$ makes the definition orientation-free; that is, the direction in which each edge is traversed is irrelevant. Consequently, the sequence of edges defining a face does not carry any specific ordering—only the multiset of edges matters. For example, the sequences $e_1e_2e_3e_4e_5e_5$ and $e_3e_1e_5e_4e_5e_2$ define the same face.

A **graph with faces** is a triple $\mathcal{G} = (V, E, F)$, where (V, E) is a graph and F is a set of faces as described above.

To relate our abstract notion of graph with faces to geometric concepts like 2-cell embeddings and PHCCs, we restrict to the planar case, which will also be the primary setting for the remainder of this work.

Definition 4.3 . A graph with faces $\mathcal{G} = (V, E, F)$ is said to be **planar** if the underlying graph (V, E) admits an embedding[1] $\gamma : (V, E) \hookrightarrow S^2$ such that there exists an injective map $\phi : F \hookrightarrow \pi_0(S^2 \setminus \mathrm{Im}(\gamma))$ from the set of faces F to the connected components of the complement of the embedding, where each $\phi(\tau)$ has topological boundary

$$\partial\phi(\tau) = \bigcup_{e \in \tau} \gamma(e).$$

The pair $\mathcal{E} = (\gamma, \phi)$ is called a **planar embedding** of $\mathcal{G}$. The **set of holes** of $\mathcal{G}$ with respect to $\mathcal{E}$, denoted $H_{\mathcal{E}}$, is defined as the set-theoretic complement of $\mathrm{Im}(\phi)$ in the set of connected components of $S^2 \setminus \mathrm{Im}(\gamma)$. We denote the **embedded** graph by

$$\mathcal{G}_\Phi := \gamma((V, E)) \cup \bigcup_{\tau \in F} \overline{\phi(\tau)} = S^2 \setminus H_\Phi \subseteq S^2$$

We denote by $\mathcal{P}^2$ the class of all planar graphs with faces.

This definition formalizes the intuitive idea of embedding a graph into the sphere S^2, identifying the connected components of the complement as *potential faces*, and then selecting a subset of these to be labeled as actual *faces*, with the remainder considered as *holes*.

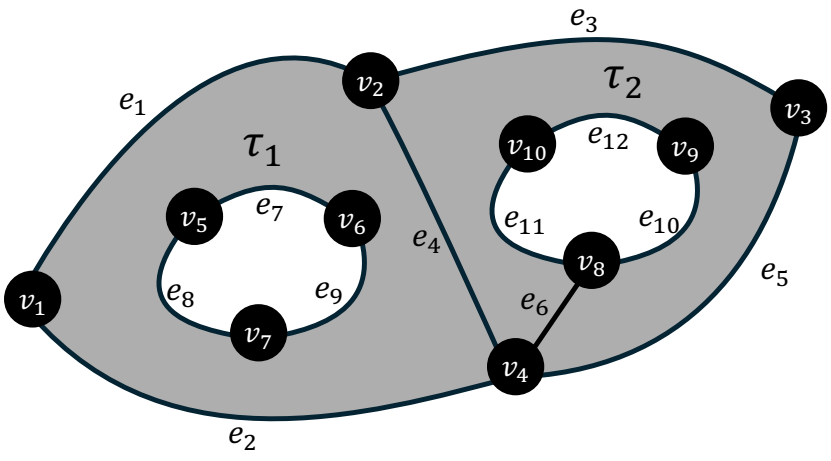


Figure 4.1: Example of a planar graph with faces that is neither a 2-cell embedding nor a planar hollow cell complex.

Example 4.4 . Figure 4.1 shows an example of a planar graph with faces with homology $H_0(\mathcal{G}) = \mathbb{F}_2$ and

[1] A graph is planar if it can be embedded in $\mathbb{R}^2$, which is equivalent to being embeddable in S^2, since $S^2 \setminus \{\mathrm{pt}\} \cong \mathbb{R}^2$.

$H_1(\mathcal{G}) = \mathbb{F}_2^2$. This particular graph with faces is not a 2-cell embedding and is also not a PHCC. We can easily see this since there are two faces $\tau_1 = e_1e_2e_4e_7e_8e_9$ and $\tau_2 = e_3e_4e_5e_6e_6e_{10}e_{11}e_{12}$. The face τ_1 is not homeomorphic to an open disk, so it does not fit the conditions of a 2-cell embedding, while the face τ_2 has a boundary that does not fit the conditions of a PHCC.

Figure 4.1 illustrates a planar graph with faces with homology groups $H_0(\mathcal{G}) = \mathbb{F}_2$ and $H_1(\mathcal{G}) = \mathbb{F}_2^2$. This example is neither a 2-cell embedding nor a planar hollow cell complex (PHCC). Specifically, the face $\tau_1 = e_1e_2e_4e_7e_8e_9$ is not homeomorphic to an open disk, violating the requirements for a 2-cell embedding. The face $\tau_2 = e_3e_4e_5e_6e_6e_{10}e_{11}e_{12}$, on the other hand, has an outer boundary that is not homeomorphic to S^1, and thus does not satisfy the conditions of a PHCC.

When the set of faces F is empty, a graph with faces reduces to an ordinary graph. Given a graph with faces $\mathcal{G} = (V, E, F)$, a **signal** over $\mathcal{G}$ is a real-valued function $f : V \to \mathbb{R}$ defined on the vertex set. For a face $\tau \in F$, we define its vertex set as

$$V(\tau) = \bigcup_{e \in \tau} V(e),$$

in which $V(e)$ is the set of edges incident to e. Furthermore, we define the signal value on the face as the maximum signal value over its vertex set:

$$f(\tau) = \max f(V(\tau)).$$

Similarly, for each edge $e = uv \in E$, we define the signal as

$$f(e) = \max\{f(u), f(v)\}.$$

These choices are motivated by our focus on analyzing the topology of signals through the persistent homology of their sublevel set filtrations. As defined above, each step of the filtration yields a graph with faces. Accordingly, we adopt this maximum-based extension of the signal to higher-order cells for the remainder of this work.

4.1 Homology of a graph with faces

The homology of $\mathcal{G}$ is defined to be the singular homology of its embedding $\mathcal{G}_\Phi = S^2 \setminus H_\mathcal{E}$. Since any planar embedding of the graph (V, E) partitions S^2 into the same amount of connected components, equal to $|F| + |H_\mathcal{E}|$, the homology of $\mathcal{G}$ is independent of the embedding $\mathcal{E}$, and is simply given by $H_0(\mathcal{G}) = \mathbb{F}_2^c$, $H_1(\mathcal{G}) = \mathbb{F}_2^{|H_\Phi|-1}$, and $H_2(\mathcal{G}) = 0$ if $|H_\Phi| > 0$, otherwise $H_2(\mathcal{G}) = \mathbb{F}_2$.

Remark 4.5 . Given a graph with faces $\mathcal{G} = (V, E, F)$, one might be tempted to define its mod 2 homology as the homology of the chain complex

$$C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \longrightarrow 0$$

where each C_i is a vector space over $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$ with basis given by the i-dimensional elements:

- C_0 is spanned by the vertices V,
- C_1 is spanned by the edges E,

- C_2 is spanned by the faces F.

The boundary maps are given by

$$\partial_1(uv) = u + v,$$

and for a face given by a sequence of edges $e_1e_2 \dots e_N \in C_2$, we define

$$\partial_2(e_1e_2 \dots e_N) = \sum_{i=1}^{N} e_i.$$

This defines a chain complex, and consequently homology groups $H_i(\mathcal{G}) = \ker \partial_i / \operatorname{im} \partial_{i+1}$. But we have to be careful, this homology will not coincide with what we are defining above as the homology of a graph with faces. The reason for this being the case is that we might have vertices in the same connected component that are connected by a face instead of an edge, like vertex v_1 and v_5 in Figure 4.1, hence if we defined homology in this way, the dimension of the 0-homology would be equal to the number of connected components in the underlying graph (without considering any faces). On the other hand, the 1-homology will match with the previous definition, so this definition considering the chain complex might be useful for computing the 1-homology of a graph with faces.

4.2 Persistent homology of a graph with faces

Given a signal $\boldsymbol{\mathcal{G}} = (\mathcal{G}, f)$ over a graph with faces $\mathcal{G} = (V, E, F)$, a **$\boldsymbol{\mathcal{G}}$-ordering** (or $(\mathcal{G}, f)$-ordering) is a total order $\prec$ on the set $V \cup E \cup F$ satisfying the following conditions:

$$\sigma_1 \prec \sigma_2 \implies f(\sigma_1) \leq f(\sigma_2), \qquad \text{for all } \sigma_1, \sigma_2 \in V \cup E \cup F, \tag{4.1}$$

$$v \prec e \quad \text{for all } v \in V(e),\ e \in E, \tag{4.2}$$

$$e \prec \tau \quad \text{for all } e \in \tau,\ \tau \in F. \tag{4.3}$$

Such an ordering is essential for defining a filtration in which cells are added one at a time. Condition (4.1) ensures that the order respects the sublevel set structure induced by the signal. Conditions (4.2) and (4.3) ensure that at each step of the filtration, the included set of cells forms a valid graph with faces.

Define $\mathcal{G}_{\prec\sigma}$ and $\mathcal{G}_{\preceq\sigma}$ as the graphs with faces having all vertices, edges and faces coming before σ in the ordering, including σ for the latter. And define $\mathcal{G}_t$ as the graph with faces having all $\sigma \in V \cup E \cup F$, such that $f(\sigma) \leq t$. Then, $\{\mathcal{G}_t\}_{t\in\mathbb{R}}$ defines the sublevel filtration of the signal f. We define the persistent homology of $\boldsymbol{\mathcal{G}}$ as being the persistence homology of this sublevel filtration.

As previously discussed, in the case of a simplicial complex, a filtration is fully determined by a total ordering of its simplices. Similarly, for a signal over a graph with faces $(\mathcal{G}, f)$, a total ordering ((G, f)-ordering)

$$\sigma_0 \prec \sigma_1 \prec \sigma_2 \prec \cdots \prec \sigma_N$$

of elements in $V \cup E \cup F$ determines the filtration. In this setting, if a homology class in dimension i is born at the inclusion of σ_b and becomes trivial upon the inclusion of σ_d, it gives rise to the interval

$$[f(\sigma_b), f(\sigma_d)) \in \mathrm{PD}_i(\mathcal{G}, f),$$

and the pair (σ_b, σ_d) is referred to as the *birth-death pair* associated with this interval.

In the case of a persistent cycle that never becomes trivial (i.e., it remains a non-trivial class after the inclusion of σ_N, also called a *permanent cycle*), if it is born at σ_b, we denote its birth-death pair as $(\sigma_b, \emptyset)$. Throughout this work, we adopt the convention $f(\emptyset) = \infty$ for any signal $f : V \to \mathbb{E}$.

Importantly, the set of birth-death pairs is determined entirely by the graph with faces structure and the total ordering $\prec$. Accordingly, we denote by $\mathrm{BD}(\mathcal{G}, \prec)$ the collection of all birth-death pairs across all homological dimensions induced by $\prec$. When referring to a specific dimension i, we write $\mathrm{BD}_i(\mathcal{G}, \prec)$ to indicate the subset corresponding to i-dimensional homology.

Note that it is possible for a birth-death pair (σ_b, σ_d) to satisfy $f(\sigma_b) = f(\sigma_d)$. In such cases, the corresponding persistence interval has zero length and appears as a diagonal point in the persistence diagram. Although such diagonal elements are typically considered topologically insignificant, they remain part of the set of birth-death pairs.

Remark 4.6 . In the case of simplicial complexes, we can guarantee that if (σ_b, σ_d) is a birth-death pair corresponding to a persistence interval in PD_i, then σ_b is an i-simplex and σ_d is an $(i+1)$-simplex. However, this correspondence does not hold in general for a graph with faces. Specifically, while the birth-death pairs in $\mathrm{BD}_1(\mathcal{G}, \prec)$ are always of the form (e, τ), with $e \in E$ and $\tau \in F$, the birth-death pairs in $\mathrm{BD}_0(\mathcal{G}, \prec)$ take the form (v, σ), where $v \in V$ and $\sigma \in E \cup F$, since either edges and faces can merge distinct connected components.

Computing persistent homology using the original definition of homology for a graph with faces, as presented above, can be quite challenging. In what follows, we propose a simplified method for computing the persistent homology of a signal over a planar graph with faces by constructing a corresponding signal over a graph. From this graph-based representation, we can efficiently extract both the persistent 0- and 1-dimensional homology of the original signal. The approach for computing persistent 1-homology is inspired by Alexander duality [Hat2002, Theorem 3.43].

Definition 4.7 . Let $\mathcal{G} = (V, E, F)$ be a planar graph with faces with an embedding $\mathcal{E}$. Define the graph $G[\mathcal{E}] = (V', E')$, where $V' = V \cup F \cup H_{\mathcal{E}}$ and $E' = E \cup E_F \cup E_{H_{\mathcal{E}}}$. Here, E_F (resp. $E_{H_{\mathcal{E}}}$) consists of all edges connecting the faces to its incident vertices:

$$E_F \triangleq \left\{ (\tau, v) \;\middle|\; \tau \in F \text{ and } v \in \bigcup_{e \in \tau} V(e) \right\}, \tag{4.4}$$

and $E_{H_{\mathcal{E}}}$ is similarly defined. Furthermore, if f is a signal over $\mathcal{G}$, we define the **signal induced on** $G[\mathcal{E}]$ **by** f as

$$f_{G[\mathcal{E}]}(v) \triangleq \begin{cases} f(v), & \text{if } v \in V \cup F, \\ \infty, & \text{if } v \in H_{\mathcal{E}}. \end{cases} \tag{4.5}$$

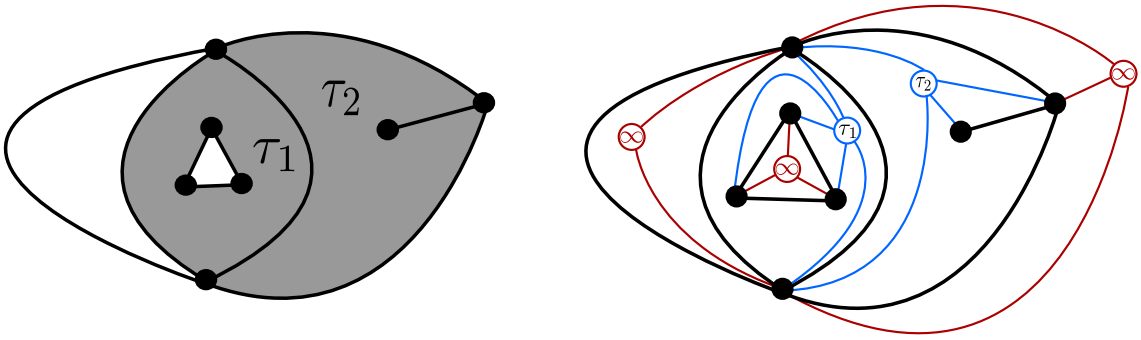


Figure 4.2: A planar graph with faces (left) and the induced graph (right).

Theorem 4.8. If $\mathcal{G} = (\mathcal{G}, f)$ is a signal over a planar graph with faces with an embedding $\mathcal{E}$, then

$$\mathrm{PD}_0(\mathcal{G}, f) = \mathrm{PD}_0(G[\mathcal{E}], f_{G[\mathcal{E}]}) \tag{4.6}$$

and there is a bijection

$$\begin{aligned} \mathrm{PD}_1(\mathcal{G}, f) &\simeq \widetilde{\mathrm{PD}}_0(G[\mathcal{E}], -f_{G[\mathcal{E}]}) \\ [a, b) &\longleftrightarrow [-b, -a) \end{aligned} \tag{4.7}$$

where $\widetilde{\mathrm{PD}}_0$ is obtained by eliminating a permanent interval $[a, \infty)$ from PD_0 with the smallest a.

Proof. Let $\mathcal{G} = ((V, E, F), f)$ be a signal over a planar graph with faces, and let $\mathcal{E} : \mathcal{G} \hookrightarrow S^2$ be an embedding.

First, we show that $\mathrm{PD}_0(\mathcal{G}, f) = \mathrm{PD}_0(G[\mathcal{E}], f_{G[\mathcal{E}]})$.

Let $\prec$ be a $\mathcal{G}$-ordering. We can extend $\prec$ defined on $V \cup E \cup F$ to a $(G[\mathcal{E}], f_{G[\mathcal{E}]})$-ordering defined on

$$V' \cup E' = V \cup F \cup H_{\mathcal{E}} \cup E \cup E_F \cup E_{H_{\mathcal{E}}}.$$

While elements of F become vertices in $G[\mathcal{E}]$, their relative ordering with respect to V and E is kept unchanged. To define this extension, we specify the ordering involving $H_{\mathcal{E}}$, E_F, and $E_{H_{\mathcal{E}}}$. For each $\tau \in F$, all its incident edges in E_F come right after τ in the ordering. At the end of the ordering, we include $H_{\mathcal{E}}$ followed by $E_{H_{\mathcal{E}}}$. Within $H_{\mathcal{E}}$, $E_{H_{\mathcal{E}}}$, and E_F, any internal ordering suffices.

Let (v, σ) be a persistence pair in $PD_0(\mathcal{G})$ with $v \in V$ and $\sigma \in E \cup F$. We will prove that

- if $\sigma \in E$, then (v, σ) is a persistence pair in $\mathrm{PD}_0(G[\mathcal{E}], f_{G[\mathcal{E}]})$.
- if $\sigma \in F$, then there exists $e' \in E_F$ incident to σ such that (v, e') is a persistence pair in $\mathrm{PD}_0(G[\mathcal{E}], f_{G[\mathcal{E}]})$.

Since (v, σ) is a persistence pair, we have $v = \min C(v, \mathcal{G}_{\prec\sigma}) \neq \min C(v, \mathcal{G}_{\preceq\sigma})$.

- If $\sigma \in E$, then $v = \min C(v, G[\mathcal{E}]_{\prec\sigma}) \neq \min C(v, \mathcal{G}[\mathcal{E}]_{\preceq\sigma})$, since all edges coming before σ in $G[\mathcal{E}]$ are the same as the ones appearing before σ in $\mathcal{G}$, plus possibly some edges from E_F. These extra edges connect the same vertices as the corresponding faces do in $\mathcal{G}$.

- If $\sigma \in F$, then there is $e' \in E_F$, incident to σ, such that $v = \min C(v, G[\mathcal{E}]_{\prec e'}) \neq \min C(v, G[\mathcal{E}]_{\preceq e'})$. This follows from the previous case and the fact that if a face $\tau \in F$ connects two vertices in $\mathcal{G}$, then these same vertices are connected by an edge connecting the first vertex to τ, and another edge connecting τ to the second vertex.

Whence, any persistence interval in $\mathrm{PD}_0(\mathcal{G}, f)$ generated by a vertex $v \in V$ is preserved in $\mathrm{PD}_0(G[\mathcal{E}], f_{G[\mathcal{E}]})$. Moreover, if (v, σ) is a nontrivial persistence pair in $\mathrm{PD}_0(G[\mathcal{E}], f_{G[\mathcal{E}]})$, we have $v \in V$, since any element in F and $H_{\mathcal{E}}$ is preceded by an adjacent one in V. This completes the proof for PD_0.

Now, we proceed to the proof that there is a bijection:

$$\begin{aligned} \mathrm{PD}_1(\mathcal{G}, f) &\longrightarrow \widetilde{\mathrm{PD}}_0(G[\mathcal{E}], -f_{G[\mathcal{E}]}) \\ [a, b) &\longmapsto [-b, -a) \end{aligned}$$

For a $\mathcal{G}$-ordering $\prec$, it is not difficult to see that there is a $(G[\mathcal{E}], -f_{G[\mathcal{E}]})$-ordering $\prec_o$ such that for any two faces $\tau_1, \tau_2 \in F$, we have

$$\tau_1 \prec \tau_2 \implies \tau_2 \prec_o \tau_1, \tag{4.8}$$

$$\tau_1 \prec_o v \text{ for } v \in V(\tau_1). \tag{4.9}$$

Any nontrivial interval $[a, b) \in \mathrm{PD}_1(\mathcal{G})$ is associated to a persistence pair (e, τ), with $e \in E$ and $\tau \in F$, with $f(e) = a < b = f(\tau)$. This means that the edge e is responsible for forming a cycle $\alpha = \sum_i e_i$, which delimits a region Ω in $\mathcal{G}$ with τ as its maximum. Therefore, τ is a local minimum of $-f_{G[\mathcal{E}]}$ in $G[\mathcal{E}]$, hence it generates a class in $\widetilde{\mathrm{PD}}_0(G[\mathcal{E}], -f_{G[\mathcal{E}]})$ born at $-f_{G[\mathcal{E}]}(\tau) = -b$, with persistence pair (τ, e'), for some $e' \in E' = E \cup E_F \cup E_{H_{\mathcal{E}}}$.

We show, by contradiction, that $-f_{G[\mathcal{E}]}(e') = -a$.

(Case 1): Suppose $-f_{G[\mathcal{E}]}(e') < -a$. Then, at $G[\mathcal{E}]_{-a}$, τ would be connected to some σ with $\sigma \prec_o \tau$. In $\mathcal{G}$, this implies that there is a path from τ to σ consisting of vertices having signal values greater than a. Since the cycle α is born at a, none of these vertices belong to the cycle. Thus, the entire path lies within the region delimited by α, which contradicts the fact that τ is the maximum in that region.

(Case 2): Suppose $-f_{G[\mathcal{E}]}(e') > -a$. Then, we have

$$\min C(\tau, G[\mathcal{E}]_{-a'}) = \tau \tag{4.10}$$

for some $a' \in \mathbb{R}$ with $-f_{G[\mathcal{E}]}(e') > -a' > -a$, in which the minimum is taken with relation to $\prec_o$. Hence, $H_{\mathcal{E}} \cap C(\tau, G[\mathcal{E}]_{-a'}) = \emptyset$. So if we define the 2-chain in $\mathcal{G}$ given by

$$\Gamma = \sum_{\substack{\tau' \in F \\ V(\tau') \cap C(\tau, G[\mathcal{E}]_{-a'}) \neq \emptyset}} \tau' \tag{4.11}$$

then the cycle $\beta = \partial_2 \Gamma$ delimits a region containing Ω. By construction, this cycle is born at or before a'. In $\mathcal{G}_{\prec \tau}$, β is homologous to α, contradicting the fact that τ is responsible for killing α in $\mathrm{PD}_1(\mathcal{G}, f)$. □

SECTION 5

Filtering problem framework

Having established a precise definition of a graph with faces and outlined the computation of persistent homology for signals defined on such structures, we now turn to the central problem of interest: the removal of topological noise from these signals. In this section, we present a natural and theoretically appealing formulation of this filtering task. However, we show—through concrete counterexamples—that this "ideal" formulation is, in general, unattainable. This limitation motivates a relaxation of the original problem, ultimately leading to the practical formulation that we aim to address.

5.1 Ideal target

Given a graph with faces $\mathcal{G} = (V, E, F)$, we define $\mathcal{S}(\mathcal{G})$ to be the set of all signals $f : V \to \overline{\mathbb{R}}$ over $\mathcal{G}$. For a signal $\mathcal{G} = (\mathcal{G}, f)$ and a threshold $\epsilon > 0$, we define the following subset of signals:

$$\mathcal{S}(\mathcal{G})_f^{\epsilon} = \left\{ g \in \mathcal{S}(\mathcal{G}) \;\middle|\; \begin{array}{l} \mathrm{PD}_i(\mathcal{G}, g) \approx \mathrm{PD}_i(\mathcal{G}, f)_{\geq \epsilon} \text{ for } i = 0, 1, \\ \text{and } \mathrm{BD}(f) = \mathrm{BD}(g) \end{array} \right\}, \tag{5.1}$$

in which when writing $\mathrm{BD}(f) = \mathrm{BD}(g)$ we mean that there is a $(\mathcal{G}, f)$-ordering and a $(\mathcal{G}, g)$-ordering that have the same birth-death pairs. We also define a less strict set of functions as

$$\mathcal{S}(\mathcal{G})^{\epsilon} = \{ g \in \mathcal{S}(\mathcal{G}) \mid \mathrm{PD}_i(\mathcal{G}, g)_{<\epsilon} \approx \emptyset \text{ for } i = 0, 1 \}.$$

In other words, $\mathcal{S}(\mathcal{G})_f^{\epsilon}$ consists of signals that share the same birth-death pairs as f, and whose persistence diagrams match that of f after removing all features with persistence less than ϵ. The set $\mathcal{S}(\mathcal{G})^{\epsilon}$ is a broader class, containing all signals on $\mathcal{G}$ whose persistence diagrams contain no features with persistence less than ϵ. Hence, we have the inclusion $\mathcal{S}(\mathcal{G})_f^{\epsilon} \subseteq \mathcal{S}(\mathcal{G})^{\epsilon}$.

In Section 3 we showed that if $H : \mathcal{S}(\mathcal{G}) \to \mathcal{S}(\mathcal{G})$ is a traditional filter (any kind of low-, band- or high-pass filter, or any denoising method) then $Hf \notin S(\mathcal{G})^{\varepsilon}$, in other words, if $\mathcal{H}$ is the space of all filtered functions for some particular spectral filter, we always have $\mathcal{H} \cap \mathcal{S}(\mathcal{G})^{\varepsilon} = \emptyset$. We are disconsidering here that in the particular case that ε is big enough we might have one element in that intersection, which is the constant eigenfunction in the spectral decomposition.

With this notations in hands we can formalize the main problem in the following way.

Problem 5.1 . Given a signal over a graph with faces $\mathcal{G} = (\mathcal{G}, f)$ and a threshold $\varepsilon > 0$, find a solution to

$$\min_{g \in \mathcal{S}(\mathcal{G})_f^{\varepsilon}} \|f - g\|_{\infty}, \tag{5.2}$$

that is, approximate the signal f by a signal g such that g has no persistence intervals with persistence less than ε, while preserving all other persistence intervals of f as well as all birth-death pairs.

This problem can be interpreted as filtering out all low-persistence features in the persistence diagram and reconstructing a signal close to f from the filtered diagram. This process is analogous to low-pass filtering in the spectral domain: the Fourier transform decomposes a signal into frequency components, selected frequencies are filtered, and the signal is then reconstructed via the inverse transform. In contrast, the

persistence diagram lacks a straightforward "inverse transform"—there is no canonical method to reconstruct a signal after modifying its persistence diagram.

This formulation of the problem aligns with the earlier observation that, in many situations, features with low persistence are considered irrelevant and may even originate from noise. Consequently, removing such low-persistence features while remaining close to the original function can yield a cleaner representation of the signal. However, relying solely on the persistence diagram would result in the loss of spatial and structural information about where these features occur. To mitigate this, the definition of $\mathcal{S}(\mathcal{G})_f^\epsilon$ explicitly incorporates birth-death pair information.

An equally important objective is the preservation of high-persistence features, which are typically associated with meaningful global structures in the data. This requirement is explicitly encoded in the formulation through the definition of $\mathcal{S}(\mathcal{G})_f^\epsilon$, ensuring that significant topological information is retained.

It is important to note that, in the remainder of this work, rather than seeking the exact minimizer of Problem 5.1 (or its later variants), we focus on constructing solutions that lie within the feasible set and remain within an acceptable margin of the original signal—specifically, within a distance less than ε in the ℓ_∞-norm. This approach is practically justified, as we assume that ε is sufficiently small to disregard classes with persistence less than ε.

We may also consider a variant of the problem:

Problem 5.2 . Given a signal over a graph with faces $\mathcal{G} = (\mathcal{G}, f)$ and a threshold $\varepsilon > 0$, find a solution for

$$\min_{g \in \mathcal{S}(\mathcal{G})^\varepsilon} |f - g|_\infty, \tag{5.3}$$

In this alternative version of the problem, we can expect the achievable minimum to be smaller since we have a much larger space of signals to choose from, once there is no longer a requirement to preserve intervals with persistence greater than ϵ. Instead, the goal is simply to approximate f by a function that contains no persistence intervals with persistence less than ϵ.

5.2 Counter-Example

Now that the problem has been properly defined, we begin by showing that Problem 5.1 may have no solution. To this end, consider the signal $f : V \to \mathbb{R}$ depicted in Figure 5.1. The ordering (respecting f) of the elements of $\mathcal{G}$ is given by

$$v_0, v_1, v_2, v_3, e_1, v_4, e_2, e_3, e_4, v_5, e_5, e_6, v_6, e_7, e_8, \tau_1, \tau_2$$

which results in the following set of birth-death pairs

$$\mathrm{BD}(\mathcal{G}, f) = \Big\{ \overbrace{(v_3, e_1), (v_4, e_2), (v_5, e_5), (v_6, e_7), \boldsymbol{(v_2, e_3), (v_1, e_6)}}^{0-\text{homology}}, \underbrace{\boldsymbol{(e_4, \tau_2)}, (e_8, \tau_1)}_{1-\text{homology}} \Big\}$$

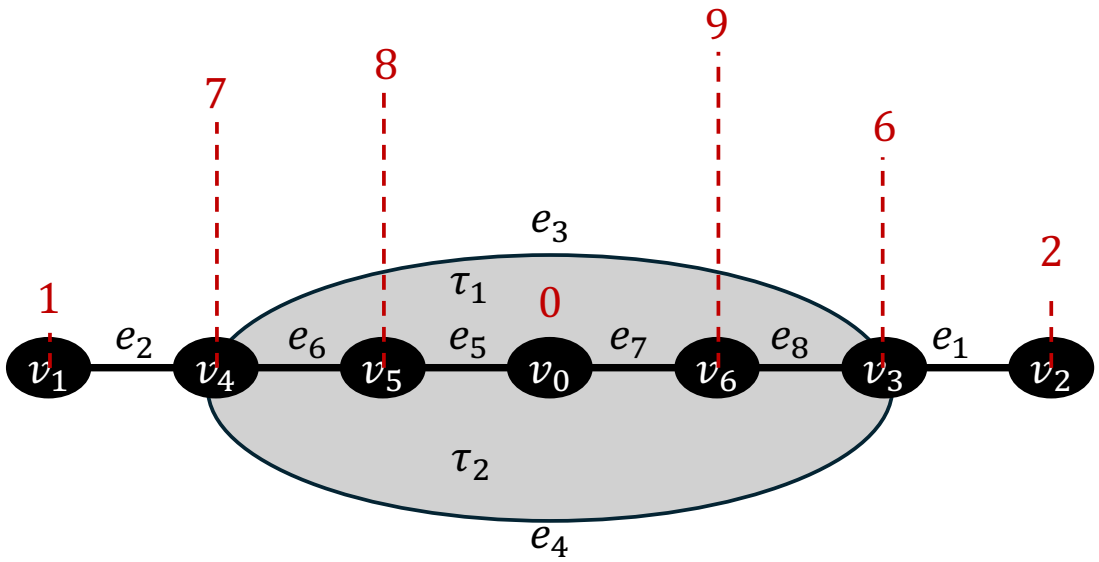


Figure 5.1: Example of a signal over a graph with faces, in which it is impossible to eliminate the only existing interval in the 1-dimensional persistence diagram without affecting the intervals in the 0-dimensional persistence diagram.

in which the bold pairs are the only ones with non-zero persistence, so we have

$$\mathrm{PD}_0(\mathcal{G}, f) = \left\{ \overbrace{[2,7)}^{[f(v_2),f(e_3))} \;,\; \overbrace{[1,8)}^{[f(v_1),f(e_6))} \right\}; \qquad \mathrm{PD}_1(\mathcal{G}, f) = \left\{ \overbrace{[7,9)}^{[f(e_4),f(\tau_2))} \right\}.$$

Suppose we wish to eliminate all homology classes with persistence less than $\epsilon = 3$, while preserving the remaining classes and the birth-death pairs, as described in Problem 5.1. We claim that this is impossible. In fact, in this case, we can prove that $\mathcal{S}(\mathcal{G})_f^\epsilon = \emptyset$.

To see this, assume by contradiction that there exists a function $g \in \mathcal{S}(\mathcal{G})_f^\epsilon$. Then we must have $\mathrm{PD}_0(\mathcal{G}, g) \approx \mathrm{PD}_0(\mathcal{G}, f)$, $\mathrm{PD}_1(\mathcal{G}, g) \approx \emptyset$, and $\mathrm{BD}(\mathcal{G}, g) = \mathrm{BD}(\mathcal{G}, f)$. From the latter two conditions, it follows that $g(e_4) = g(\tau_2)$, which implies

$$\max\{g(v_0), g(v_3), g(v_4), g(v_5), g(v_6)\} = \max\{g(v_3), g(v_4)\}.$$

This leads to the conclusion that $g(v_5) \leq \max\{g(v_3), g(v_4)\}$.

On the other hand, the condition $\mathrm{PD}_0(\mathcal{G}, g) \approx \mathrm{PD}_0(\mathcal{G}, f)$ implies that $g(e_3) = 7$ and $g(e_6) = 8$. Therefore,

$$\max\{g(v_3), g(v_4)\} = 7 < 8 = \max\{g(v_4), g(v_5)\},$$

which in turn implies $\max\{g(v_3), g(v_4)\} < g(v_5)$—a contradiction. Hence, no such function g can exist, and we conclude that $\mathcal{S}(\mathcal{G})_f^\epsilon = \emptyset$.

This example highlights an interesting phenomenon: even in a very simple case with only 7 vertices, 7 edges, and 2 faces, Problem 5.1 admits no solution.

5.3 Revised problem

As previously discussed, the ideal formulation of the problem is rather restrictive. To address this limitation, we now introduce a relaxed version, which will serve as the primary focus for the remainder of this work. We will demonstrate that, for the class of functions where an ideal solution exists, our solution to the relaxed problem remains adequate.

An appealing alternative to Problem 5.1 is Problem 5.2, which is significantly less restrictive. However, this increased flexibility may lead to undesirable behavior in the output. For instance, instead of producing a function that effectively filters out features with persistence below the threshold, the solution might instead increase the persistence of these features, thereby moving them out of the threshold region and unintentionally emphasizing them. Similarly, it may amplify the persistence of cycles that were already outside the threshold region. To address these issues and ensure that irrelevant features are effectively "filtered" without artificially enhancing relevant ones, we introduce the following definition.

Definition 5.3 . Let f and g be signals defined over a graph with faces $\mathcal{G}$. We write

$$g \lesssim f$$

if and only if there exist a $(\mathcal{G}, f)$-ordering $\prec$ and a $(\mathcal{G}, g)$-ordering $\prec'$ such that there is a bijection

$$\begin{aligned} \mathrm{BD}(\mathcal{G}, \prec) &\to \mathrm{BD}(\mathcal{G}, \prec') \\ (\sigma_b, \sigma_d) &\mapsto (\sigma_b, \varphi(\sigma_d)), \end{aligned}$$

and for every $(\sigma_b, \sigma_d) \in \mathrm{BD}(\mathcal{G}, \prec)$, the corresponding persistence intervals satisfy

$$\mathrm{pers}\big(f(\sigma_b), f(\sigma_d)\big) \geq \mathrm{pers}\big(g(\sigma_b), g(\varphi(\sigma_d))\big). \tag{5.4}$$

In other words, each birth cell σ_b in g generates a persistence interval that is no longer than the interval it generated in f. We do not require that the death cell σ_d be preserved, which is reasonable since the birth cell typically determines the spatial or structural location of the topological feature, whereas the death cell can vary more freely. The bijection φ thus maps death cells in $\prec$ to those in $\prec'$, without requiring identity. In the special case where φ is the identity map, we obtain the stronger condition $\mathrm{BD}(\mathcal{G}, \prec) = \mathrm{BD}(\mathcal{G}, \prec')$.

The essence of this definition is that filtering should not introduce new topological features nor amplify existing ones, but should only attenuate the persistence of the original features. In other words, filtering is intended to preserve or simplify the topological structure of the data.

We now adapt the definition of the solution space given in Equation (5.1) to the following set:

$$\tilde{\mathcal{S}}(\mathcal{G})_f^{\varepsilon} = \left\{ g \in \mathcal{S}(\mathcal{G}) \;\middle|\; \begin{array}{ll} g \lesssim f & \text{for } i = 0, 1, \\ \text{and } \mathrm{PD}_i(\mathcal{G}, g)_{<\varepsilon} \approx \emptyset \end{array} \right\}, \tag{5.5}$$

this set contains all functions g which at most atenuate all topological features of f while completely eliminating those with persistence less than ε. This is never empty since g = constant function is always an element of the set. Even better, later we will show there is always an ε-approximation of f in $\tilde{\mathcal{S}}(\mathcal{G})_f^{\varepsilon}$ (see Theorem 7.10).

With this definition in place, we can now reformulate Problem 5.1 as follows:

Problem 5.4 . Given a signal over a graph with faces $\mathcal{G} = (\mathcal{G}, f)$ and a threshold $\varepsilon > 0$, find a solution to

$$\min_{g \in \tilde{\mathcal{S}}(\mathcal{G})_f^\varepsilon} \|f - g\|_\infty, \tag{5.6}$$

that is, approximate the signal f by a signal g that eliminates all persistence intervals with persistence less than ε, and reduces or preserves the persistence of all remaining intervals.

In the remainder of this work, we will show that the problem above always admits a solution, and we will provide an explicit construction of and element of $g \in \tilde{\mathcal{S}}(\mathcal{G})_f^\varepsilon$, such that $\|f - g\|_\infty < \varepsilon$.

Summary

In this chapter, we explored the intersection of Graph Signal Processing and Topological Data Analysis with the goal of developing novel filtering techniques for graph signals based on persistent homology.

We began by introducing the fundamentals of Graph Signal Processing, with particular emphasis on spectral filtering techniques. We then presented the concept of persistent homology from Topological Data Analysis as a means of quantifying what we term *topological noise*—features in the sublevel filtration of a signal that exhibit low persistence, where "low" is defined relative to a chosen threshold.

Using a concrete example, we demonstrated that conventional spectral filters and standard denoising techniques are generally inadequate for removing topological noise from graph signals. In particular, complete removal of topological noise is only achieved when the signal is reduced to the constant eigenfunction. This limitation motivates the need for alternative filtering approaches that specifically target low-persistence features.

To address this, we introduced a general structural framework for the class of signals under consideration and formulated an ideal version of the topological filtering problem. We showed, via a counterexample, that this ideal formulation does not admit a solution in general, due to the "entanglement" of persistent classes in different homology dimensions.

Given the restrictive nature of the ideal problem, we proposed a relaxed formulation. This version aims to eliminate features with persistence below a given threshold while preserving high-persistence features as faithfully as possible. Importantly, it allows for minor attenuation of persistent features but strictly forbids their amplification.

The remainder of this work is devoted to analyzing this relaxed problem, establishing the existence of solutions, and constructing approximate solutions explicitly.

Chapter III

Persistent homology based filters

In this chapter, we develop approximate solutions (ϵ-approximations) to Problems 5.1 and 5.4. More specifically, we show that in the one-dimensional case (i.e., graphs without 2-dimensional cells), it is possible to obtain an approximate solution to the more strict problem Problem 5.1. In contrast, for the more general two-dimensional case, we can only construct an approximate solution to Problem 5.4, due to the previously discussed challenge of simultaneously filtering both 0- and 1-dimensional homological features.

For the more complex 2 dimensional case, we construct the solution based on the notion of the *induced graph* (see Definition 4.7). This concept allows us to decompose the original problem—defined over a general graph with faces—into two simpler subproblems, each associated with a distinct graph. Specifically, the induced graph construction enables us to separate the treatment of 0-dimensional and 1-dimensional homology. By analyzing sublevel filtrations on these two associated graphs, we can independently extract and filter topological features corresponding to connected components and cycles in the original signal

The construction proceeds through the following sequence of steps:

1. **Compute the induced graph.** Given a signal over a graph with faces, ($\mathcal{G} = (V, E, F),\ f : V \to \mathbb{R}$), and a fixed embedding $\mathcal{E} : \mathcal{G} \hookrightarrow S^2$, we construct the *induced graph* $G[\mathcal{E}]$, together with the induced signal $f_{G[\mathcal{E}]} : V' \to \mathbb{R}$ defined over the vertex set V' of $G[\mathcal{E}]$. This induced structure captures both the original signal and the topological configuration of the graph.

2. **Filter 0-dimensional topological noise.** We then approximate the induced signal $f_{G[\mathcal{E}]}$ by a new signal g_0 such that all 0-dimensional homology classes with persistence below the chosen threshold ε are eliminated. This step corresponds to suppressing connected components that are deemed topologically irrelevant under the persistence criterion.

3. **Filter 1-dimensional topological noise via duality.** Next, we approximate $-g_0$ by $-g_1$, again targeting the removal of 0-dimensional homology classes with persistence below ε. By Theorem 4.8, this second filtering step is equivalent to eliminating 1-dimensional homology classes from the original signal f,

due to the duality induced by negating the signal.

4. **Project back to the original domain.** Finally, we restrict the filtered signal g_1 back to the original graph domain by setting $g = g_1|_V$. This yields a signal defined on the original vertex set V that satisfies the relaxed filtering criteria: low-persistence features are removed, and higher-persistence features are preserved or attenuated, but never amplified.

It is important to note that, due to the interplay between 0- and 1-dimensional homology under dual filtering, steps 2 and 3 may need to be applied iteratively. However, as we will show, this process converges in a finite number of iterations to a signal that satisfies the desired topological constraints.

SECTION 6

Basin Hierarchy Tree

Before introducing the actual procedure for filtering low-persistence classes, we first present a data structure that will aid in this task. We refer to this structure as the *Basin Hierarchy Tree* (BHT), which is specifically designed to encode information derived from persistent homology. The BHT is an adaptation of the classical Union-Find data structure, which we briefly review below.

The Union Find Algorithm

The *Union-Find* algorithm, also known as the *Disjoint Set Union* (DSU) data structure, is a fundamental algorithmic technique used to efficiently track a partition of a set into disjoint subsets. It supports two primary operations: `find`, which determines the representative (or root) of the set containing a particular element, and `union`, which merges two disjoint sets into a single set.

Formally, let X be a finite set. A disjoint-set data structure maintains a collection $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ such that:

$$\bigcup_{i=1}^{k} S_i = X, \quad S_i \cap S_j = \emptyset \quad \text{for } i \neq j.$$

Each S_i represents a disjoint subset of X. The `find` operation takes an element $x \in X$ and returns a unique identifier (typically the root of a tree representing the subset) of the set S_i such that $x \in S_i$. The `union` operation takes two elements $x, y \in X$ and merges the sets containing x and y, i.e., replaces $\{S_i, S_j\}$ with $S_i \cup S_j$ if $x \in S_i$, $y \in S_j$, and $i \neq j$, in practice this is done by setting the parent of the root of one of sets to be the root of the other set.

Let $parent[x]$ denote the parent of element x in a rooted tree representation of the set. Then the `find` operation with path compression can be implemented recursively as:

$$\texttt{find}(x) = \begin{cases} x & \text{if } parent[x] = x, \\ parent[x] := \texttt{find}(parent[x]) & \text{otherwise.} \end{cases}$$

The `union` operation merges two sets by connecting their roots:

$$\texttt{union}(x, y) = \texttt{link}(\texttt{find}(x), \texttt{find}(y)),$$

where `link` selects the root based on rank or size.

Efficient implementations of the Union-Find algorithm utilize two key heuristics:

- **Path compression**: During the `find` operation, each visited node on the path from a node to the root is updated to point directly to the root. This dramatically flattens the structure of the tree and accelerates subsequent operations by reducing the effective depth.
- **Union by rank or size**: When merging two sets, the root of the smaller or shallower tree is made a child of the root of the larger or deeper tree. Here the word *rank* is used instead of *height*, because when path compression is used together with union by rank, the rank may no longer represent the exact height of the tree. Rather, it serves as an upper bound or a rough estimate, maintained solely to ensure logarithmic merging behavior.

These two heuristics, when used together, guarantee that any sequence of m operations (both `find` and `union`) on n elements runs in nearly-linear time $O(\alpha(n).m) \approx O(m)$. More precisely, the time complexity per operation is $O(\alpha(n))$, where $\alpha(n)$ denotes the inverse Ackermann function. This function grows extremely slowly—so slowly that $\alpha(n) \leq 5$ for all practical input sizes encountered in real-world computations. Thus, the Union-Find data structure is effectively constant-time per operation in most applications.

The Union-Find algorithm is a fundamental tool in computer science, widely used in applications such as Kruskal's algorithm for computing minimum spanning trees, dynamic connectivity problems, image segmentation, and the tracking of connected components in graphs—an aspect particularly relevant to our context. Its conceptual simplicity, combined with the availability of highly efficient implementations, makes it a core data structure in a broad range of combinatorial and algorithmic problems.

Algorithm 1 presents the full pseudocode for the Union-Find algorithm, implemented with both path compression and union by rank heuristics. An instance of the Union-Find class represents a collection of disjoint sets. At initialization, each element forms its own singleton set, which is reflected in the constructor method by setting `parent`$[x] = x$ for all x. The class supports two primary operations: `Find`, which returns the representative (or root) of the set containing a given element, and `Union`, which merges the sets containing two given elements.

From Union-Find to Basin Hierarchy Tree

Motivated by the efficiency and structural clarity of the Union-Find data structure in maintaining disjoint sets under dynamic connectivity operations, we propose an adaptation tailored for topological persistence computations. This leads us to the introduction of a new data structure, termed the *Basin Hierarchy Tree* (BHT), which is designed to operate as an efficient structure for persistence-based filtration processes over graphs.

The BHT retains many of the structural advantages of the Union-Find data structure, including its simplicity and effectiveness in representing disjoint components and supporting merge operations. However, it cannot achieve the same level of efficiency—particularly the near-constant time complexity—due to the intrinsic limitations imposed by persistence tracking. Specifically, the use of rank-based union heuristics, which are central to the efficiency of Union-Find, is incompatible with the strict ordering constraints required in persistent filtrations. As a result, while the BHT still encodes relationships between critical cells, supports dynamic inclusion of new simplices or cells, and facilitates the identification of birth-death pairs in persistent homology, its operational complexity is inherently higher and more sensitive to the filtration order.

In this section, we formalize the definition of the Basin Hierarchy Tree (BHT) and demonstrate its main structural and algorithmic properties. We highlight how the BHT captures essential topological information

Algorithm 1: Union-Find algorithm with path compression and union by rank

```
1:  procedure CONTRUCTOR(list of elements: X)
2:      for x ∈ X do
3:          parent[x] ← x
4:          rank[x] ← 0
5:      end for
6:  end procedure
7:  procedure FIND(x)
8:      if parent[x] ≠ x then
9:          parent[x] ← FIND(parent[x])
10:         rank[x] ← 1
11:     end if
12:     return parent[x]
13: end procedure
14: procedure UNION(x, y)
15:     xRoot ← FIND(x)
16:     yRoot ← FIND(y)
17:     if xRoot = yRoot then
18:         return
19:     end if
20:     if rank[xRoot] < rank[yRoot] then
21:         parent[xRoot] ← yRoot
22:     else if rank[xRoot] > rank[yRoot] then
23:         parent[yRoot] ← xRoot
24:     else
25:         parent[yRoot] ← xRoot
26:         rank[xRoot] ← rank[xRoot] + 1
27:     end if
28: end procedure
```

and serves as a foundational tool for the development of persistence-aware filtering techniques.

Throughout this section, we consider signals defined over connected graphs. We do not employ the graph with faces formalism here, as the concepts introduced will later be applied specifically to the induced graph of a graph with faces, as described in the preceding sections. It is important to emphasize that, even if the underlying graph of a connected graph with faces is not connected, its induced graph will be connected. This arises from the construction itself: faces in the graph with faces become vertices in the induced graph, effectively linking regions that were originally disconnected in the underlying graph. Therefore, our assumption of graph connectivity in this section does not imply that the underlying graph must be connected, but rather that the graph with faces as a whole must be connected.

We adopt standard graph-theoretic terminology throughout. A **rooted tree** $T = (V, E_T)$ is a tree with a designated root vertex $r \in V$. For any non-root vertex $v \in V \setminus \{r\}$, its **parent** $p(v)$ is defined as the unique adjacent vertex on the path from v to the root r. Given a graph $G = (V, E)$, we denote by $C(v, G) \subseteq V$ the vertex set of the connected component of G that contains the vertex $v \in V$.

When discussing signals over graphs, all notations used for graph with faces are inherited, since graphs are a particular case when there are no faces. When $\boldsymbol{G} = ((V, E), f)$ is a signal over a graph and $\prec$ is a

$\boldsymbol{G}$-ordering, then if $\sigma_o \in V \cup E$ we define the subsets

$$\begin{aligned}\text{Cells}(\prec \sigma_o) &= \{\sigma \in V \cup E \mid \sigma \prec \sigma_o\},\\ \text{Cells}(\preceq \sigma_o) &= \{\sigma \in V \cup E \mid \sigma \preceq \sigma_o\}.\end{aligned}$$

These correspond to all cells strictly preceding, or preceding and including (respectively), σ_o in the $\boldsymbol{G}$-ordering. From these sets of cells, we define corresponding subgraphs of $G = (V, E)$ as:

$$\begin{aligned}G_{\prec\sigma_o} &= (V \cap \text{Cells}(\prec \sigma_o),\ E \cap \text{Cells}(\prec \sigma_o))\,,\\ G_{\preceq\sigma_o} &= (V \cap \text{Cells}(\preceq \sigma_o),\ E \cap \text{Cells}(\preceq \sigma_o))\,.\end{aligned}$$

In other words, $G_{\prec\sigma_o}$ is the subgraph containing all vertices and edges that come earlier than σ_o in the $\boldsymbol{G}$-ordering, and $G_{\preceq\sigma_o}$ is the same subgraph with the addition of σ_o itself.

Theorem 6.1. Let $\boldsymbol{G} = ((V, E), f)$ be a signal over a connected graph, and let $\prec$ be a $\boldsymbol{G}$-ordering. Then, there is a unique tree $T = (V, E_T)$ such that for each $v \in V \setminus \{\min V\}$, there exists $e \in E$ satisfying

$$\min C(v, G_{\prec e}) = v, \tag{6.1}$$
$$\min C(v, G_{\preceq e}) = p(v). \tag{6.2}$$

Proof. First, Algorithm 2, a modified version of the union-find algorithm, constructs a tree that satisfies these conditions, thereby establishing existence. The uniqueness of the edge e associated with each vertex v then guarantees the uniqueness of the resulting tree—specifically, (v, e) is a birth-death pair. □

Definition 6.2 . In the context of Theorem 6.1, we define the **linking vertex of** v, denoted by $\mathfrak{L}(v)$, as $\max V(e)$ for $v \neq \min V$, and $\mathfrak{L}(\min V) = \emptyset$. The **Basin Hierarchy Tree** (BHT) of $\boldsymbol{G}$ with respect to $\prec$ is the triple $\mathfrak{T} = (V, E_T, \mathfrak{L})$. The partial order on V induced by $\mathfrak{T}$ is denoted by $p(v) \prec_T v$. The set of all BHT's for $\boldsymbol{G}$ with varying $\boldsymbol{G}$-ordering is denoted by $\mathbf{BHT}(\boldsymbol{G}, \boldsymbol{f})$.

In short, the *Basin Hierarchy Tree* (BHT) is a combinatorial structure that encodes the hierarchical merging of connected components in a graph filtration induced by a scalar signal. More precisely, the BHT is a tree whose structure reflects the order in which vertices merge into expanding basins as the filtration progresses. It generalizes the Union-Find paradigm to a topologically constrained setting, offering a simple way to track the evolution of components and their persistence within signal-induced filtrations.

An auxiliary construct, the *linking vertex map* $\mathfrak{L}$, records for each vertex v the maximum (with respect to the signal) among the two endpoints of the edge that connects v to its parent in the tree. This additional information is crucial for persistence computations, as will be demonstrated soon.

The collection $\text{BHT}(\boldsymbol{G}, \boldsymbol{f})$ denotes the set of all BHTs arising from different $\boldsymbol{G}$-orderings that are compatible with the signal f. As we will show later, different $\boldsymbol{G}$-orderings may lead to distinct BHTs; however, the persistent homology extracted from them remains invariant. Nonetheless, the filtration technique based on BHTs, which we introduce shortly, can be influenced by the choice of $\boldsymbol{G}$-ordering, as it determines the specific structure of the tree and, consequently, the behavior of the filtering process.

Remark 6.3 . If $\mathfrak{T} = (T, \mathfrak{L})$ is a BHT of (G, f), then each tuple $(v, \mathfrak{L}(v), p(v))$ forms a merge triplet defined in [SM2020], and $\mathfrak{T}$ is closely related to a minimal normalized triplet representation obtained from those merge triplets.

Algorithm 2: Constructing the BHT. At intermediate stages in the for loop, we have a forest of rooted trees, and root(v) denotes the root of the tree containing v in this forest.

Require: $\boldsymbol{G} = ((V, E), f)$ with a $\boldsymbol{G}$-ordering on $V \cup E$
Initialize array: parent$[v] \leftarrow v$ for all $v \in V$
Initialize linking_vertex$[v] \leftarrow$ None for all $v \in V$
function FIND(v)
    while parent$[v] \neq v$ **do**
        $v \leftarrow$ parent$[v]$
    end while
    return v
end function
for each edge $e = (u, v) \in E$, in increasing $\boldsymbol{G}$-order **do**
    $r_u \leftarrow$ FIND(u)
    $r_v \leftarrow$ FIND(v)
    if $r_u \neq r_v$ **then**
        $r_{\min} \leftarrow \min(r_u, r_v)$, $r_{\max} \leftarrow \max(r_u, r_v)$
        parent$[r_{\max}] \leftarrow r_{\min}$ ▷ Union without path compression
        linking_vertex$[r_{\max}] \leftarrow \max(u, v)$ ▷ Record merge initiator
    end if
end for
return structure $\mathfrak{T} =$ (parent, linking_vertex)

Algorithm 2 presents a naive construction of the Basin Hierarchy Tree (BHT), which results in a time complexity of $O(mn)$, where n is the number of vertices and m is the number of edges. Unlike the classical Union-Find algorithm, this construction imposes stricter constraints: the merge order is entirely dictated by the $\boldsymbol{G}$-ordering, preventing the use of rank-based union heuristics. Additionally, to preserve the explicit tree structure of the BHT, path compression cannot be applied directly.

However, we can introduce a parallel data structure that mirrors the parent relationships while allowing path compression, as in the standard Union-Find algorithm (see Algorithm 3). This auxiliary structure improves the runtime without increasing the space complexity, which remains $O(n)$. By incorporating path compression in this auxiliary structure, the overall time complexity is reduced to $O(m \log n)$.

In the following theorem, we show that the persistent 0-homology of (G, f) can be recovered from any $\mathfrak{T} \in \mathrm{BHT}(G, f)$, independently of the choice of $\boldsymbol{G}$-ordering.

Let $\mathfrak{T} = (V, E_T, \mathfrak{L}) \in \mathrm{BHT}(G, f)$ be a Basin Hierarchy Tree. The **persistence** of a vertex $v \in V$ in $\mathfrak{T}$ is defined as

$$\mathrm{pers}_{\mathfrak{T}}(v) = f(\mathfrak{L}(v)) - f(v). \tag{6.3}$$

If $\mathrm{pers}_{\mathfrak{T}}(v) = 0$, we say that v is a **trivial vertex**.

Algorithm 3: Constructing the BHT (Basin Hierarchy Tree) with Union-Find bookkeeping.

Require: $\boldsymbol{G} = ((V, E), f)$ with a $\boldsymbol{G}$-ordering on E

Initialize arrays: parent$[v] \leftarrow v$, compressed_parent$[v] \leftarrow v$ for all $v \in V$
Initialize linking_vertex$[v] \leftarrow$ None for all $v \in V$
function FIND(v)
    if compressed_parent$[v] \neq v$ **then**
        compressed_parent$[v] \leftarrow$ FIND(compressed_parent$[v]$) ▷ Path compression
    end if
    return compressed_parent$[v]$
end function
for each edge $e = (u, v) \in E$, in increasing $\boldsymbol{G}$-order **do**
    $r_u \leftarrow$ FIND(u)
    $r_v \leftarrow$ FIND(v)
    if $r_u \neq r_v$ **then**
        $r_{\min} \leftarrow \min(r_u, r_v)$, $r_{\max} \leftarrow \max(r_u, r_v)$
        compressed_parent$[r_{\max}] \leftarrow r_{\min}$ ▷ Efficient Union
        parent$[r_{\max}] \leftarrow r_{\min}$ ▷ BHT (no compression)
        linking_vertex$[r_{\max}] \leftarrow \max(u, v)$ ▷ Record merge initiator
    end if
end for
return structure $\mathfrak{T}$ = (parent, compressed_parent, linking_vertex)

Theorem 6.4. For any $\mathfrak{T} \in \mathrm{BHT}(G, f)$, the following identity of persistence diagrams holds:

$$\mathrm{PD}_0(G, f) \approx \left\{ \left(f(v), f(\mathfrak{L}(v)) \right) \,\middle|\, v \in V \right\}.$$

Here, both sides are considered as elements of $\mathcal{D}_\Delta$ (persistence diagrams modulo diagonal equivalence). On the right-hand side, we use a slight abuse of notation: the set

$$\left\{ \left(f(v), f(\mathfrak{L}(v)) \right) \,\middle|\, v \in V \right\}$$

denotes the multiset $\mu \in \mathcal{D}_\Delta$ such that

$$\mu(x, y) = \#\left\{ v \in V \,\middle|\, (f(v), f(\mathfrak{L}(v))) = (x, y) \right\}.$$

Proof. By construction, it is easy to see that (v, e) in Theorem 6.1 forms a persistence pair for all $v \in V$, whose persistence interval is $[f(v), f(e)) = [f(v), f(\mathfrak{L}(v))$, since e is defined exactly as the edge that connects v to an older connected component, one with a minimum lower than v. □

The BHT encodes the persistent 0-homology in decreasing order of persistence, in the sense that each child vertex in the tree is associated with a persistence interval that is lower than or equal to that of its parent. This hierarchical structure reflects the temporal evolution of connected components in the filtration, where components born later (and thus with lower persistence) are merged into older, more persistent ones.

Lemma 6.5 . Let $\mathfrak{T} = (V, E_T, \mathfrak{L})$ be a BHT, and let $v \in V \setminus \{\min V\}$ and $u \prec_T v$. Then the following

properties hold:

$$f\big(\mathfrak{L}(v)\big) \leq f\big(\mathfrak{L}(u)\big), \tag{6.4}$$

$$\mathrm{pers}_{\mathfrak{T}}(v) \leq \mathrm{pers}_{\mathfrak{T}}(u). \tag{6.5}$$

Proof. Let $\mathfrak{T} = (V_T, E_T, \mathfrak{L})$ be the BHT of (G, f) with respect to $\prec$ and let $v \in V$. If $p(v)$ is the root of $\mathfrak{T}$ then $f(\mathfrak{L}(p(v))) = \infty$, and the result is trivial. If $p(v)$ is not the root, then let $e, e' \in E$ be the edges satisfying $f(\mathfrak{L}(v)) = f(e)$ and $f(\mathfrak{L}(p(v))) = f(e')$, as in the definition of BHT. By definition, we have

$$v = \min C(v, G_{\prec e}) \succ \min C(v, G_{\preceq e}) = p(v),$$

and

$$p(v) = \min C(p(v), G_{\prec e'}) \succ \min C(p(v), G_{\preceq e'}) = p(p(v)).$$

From the two inequalities above we get

$$p(v) = \min C(v, G_{\preceq e}) \succ \min C(p(v), G_{\preceq e'}) = p(p(v))$$

Hence, $C(v, G_{\preceq e}) = C(p(v), G_{\preceq e}) \subset C(p(v), G_{\preceq e'})$. Therefore, $e \prec e'$, which implies

$$f\Big(\mathfrak{L}(v)\Big) = f(e) \leq f(e') = f\Big(\mathfrak{L}(p(v))\Big).$$

If $u \prec_T v$, there exists a natural number m such that $u = p^m(v) = p \circ p \circ \cdots \circ p(v)$, and we have $f\Big(\mathfrak{L}(v)\Big) \leq f\Big(\mathfrak{L}(u)\Big)$. Moreover, we have

$$u \prec_T v \implies u \prec v \implies f(u) \leq f(v).$$

In conclusion

$$\mathrm{pers}_{\mathfrak{T}}(v) = f\big(\mathfrak{L}(v)\big) - f(v) \leq f\big(\mathfrak{L}(u)\big) - f(u) = \mathrm{pers}_{\mathfrak{T}}(u).$$

□

Lemma 6.5 establishes that persistence decreases along the branches of a BHT as we move from parent to child. This monotonicity property will play a key role later in enabling the efficient computation of the persistence-based filter.

Proposition 6.6 . Any BHT $\mathfrak{T} = (T, \mathfrak{L})$ satisfies the following properties:

1. If $v \preceq_T u$, then $f(u) \leq f(\mathfrak{L}(v))$.
2. If $v \prec_T u$ and $C(w, G_t) = C(u, G_t)$ for some $t < f(\mathfrak{L}(v))$, then $v \preceq_T w$.
3. For any $v \in V \setminus \{\min V\}$, there exists a path from $\mathfrak{L}(v)$ to v in T such that all signal values along the path lie in the interval $[f(v), f(\mathfrak{L}(v))]$.

Proof. **(i) 1:** Since $v \preceq_T u$, it means that u merges into the component containing v at a point when v is still the minimum vertex of that component. There are two cases to consider:

- If v is the root of the tree, then $\mathfrak{L}(v) = \emptyset$, and we conventionally set $f(\emptyset) = \infty$, so the inequality $f(u) \leq f(\mathfrak{L}(v))$ holds trivially.

- If v is not the root, then there exists an edge e that merges v into an older component. By construction, $f(e) = f(\mathfrak{L}(v))$. Since u joins v's component before this merging occurs, we must have $u \prec e$ in the $\boldsymbol{G}$-ordering, and hence $f(u) \leq f(e) = f(\mathfrak{L}(v))$.

(ii) 2: Suppose $v \prec_T u$ and $C(w, G_t) = C(u, G_t)$ for some $t < f(\mathfrak{L}(v))$. Then at time t, and for all later times, w lies in the same connected component as u. Since $v \prec_T u$, there exists a time $t_o \leq f(\mathfrak{L}(v))$ at which u merges into the component of v.

Now, consider the moment when w merges into the component containing u (and hence, eventually v):

- If w merges into u's component before u merges into v's, then when u merges into v's component, w joins as well. Since v is the minimum in the component at that point, it follows that $v \prec_T w$.

- Alternatively, if u merges into v's component before w joins u, then w merges directly into the component where v is still the minimum. Again, this implies $v \prec_T w$.

In both cases, we conclude that $v \preceq_T w$ (accounting for the case where $v = w$ which we disregarded in the proof).

(iii) 3: From the definition of the BHT, let e be the edge such that

$$v = \min C(v, G_{\prec e}) \neq \min C(v, G_{\preceq e}) = p(v).$$

By definition, $\mathfrak{L}(v) = \max V(e)$. If $\mathfrak{L}(v) \in C(v, G_{\prec e})$, then $\mathfrak{L}(v)$ is already in the same component as v just before e is added, and the assertion follows trivially since all function values in $C(v, G_{\prec e})$ lie within $[f(v), f(\mathfrak{L}(v))]$.

If $\mathfrak{L}(v) \notin C(v, G_{\prec e})$, then let γ be a path within $C(v, G_{\prec e})$ from v to $\min V(e)$. Concatenating the edge e to γ yields a path from v to $\mathfrak{L}(v)$. All vertices along this path, including the endpoints of e, are active at level $f(\mathfrak{L}(v))$ and must have values in the interval $[f(v), f(\mathfrak{L}(v))]$, completing the proof. □

Example 6.7 . We consider a graph with six vertices and a signal over it, as depicted in Figure 6.1 (left). Its BHT, computed with respect to the unique $\boldsymbol{G}$-ordering satisfying $wx \prec vx$ and $vy \prec zy$, is given in the middle of the figure. A related but different notion is the merge tree [MBW2013], which captures information about the merging of connected components as the threshold increases. The merge tree for this example is illustrated on the right of the figure.

Definition 6.8 (Compatible Ordering)**.** Let $\prec$ be a $(\mathcal{G}, f)$-ordering for a signal f on a graph with faces $\mathcal{G}$. Then, for any other signal g on the same graph with faces, there exists a unique $(\mathcal{G}, g)$-ordering $\prec'$ such that

$$\sigma_1 \prec \sigma_2 \iff \sigma_1 \prec' \sigma_2 \quad \text{whenever } g(\sigma_1) = g(\sigma_2). \tag{6.6}$$

That is, $\prec'$ refines the partial ordering induced by g into a total ordering by breaking ties according to the original ordering $\prec$. This construction ensures that $\prec'$ is consistent with g, while preserving the relative order of elements that are indistinguishable under g alone.

In this case, we say that the $(\mathcal{G}, g)$-ordering $\prec'$ is *compatible* with $\prec$.

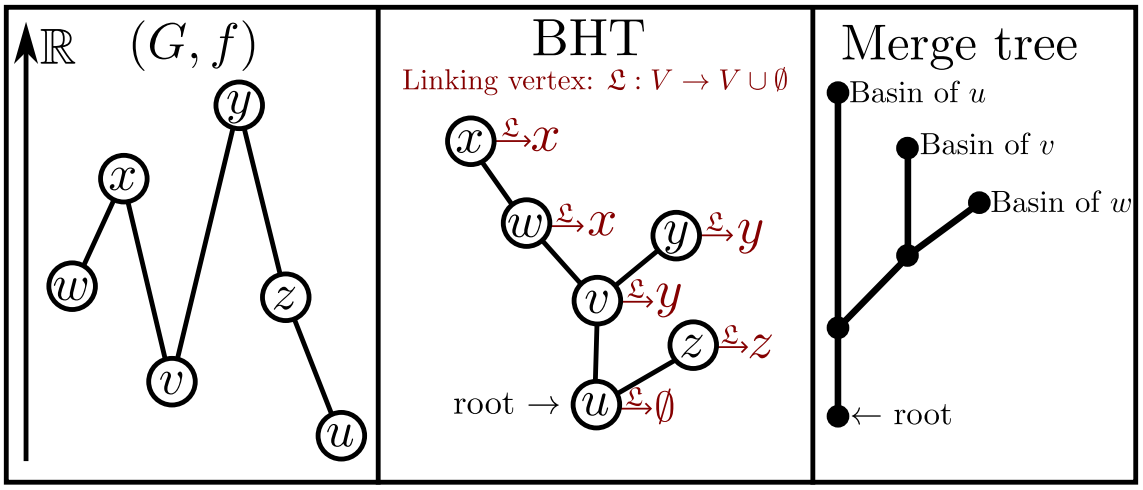


Figure 6.1: A signal over a graph (left) with its corresponding BHT (middle) and merge tree (right).

Proposition 6.9 . Let f and g be two signals over the same graph $G = (V, E)$ such that

$$f(u) \leq f(v) \Rightarrow g(u) \leq g(v), \quad \text{for all edges } uv \in E.$$

Let $\prec$ be a (G, f)-ordering, and $\prec'$ a (G, g)-ordering compatible with $\prec$. If both $\prec$ and $\prec'$ induce the same Basin Hierarchy Tree (BHT), then the sets of birth-death pairs coincide: $\mathrm{BD}(G, \prec) = \mathrm{BD}(G, \prec')$.

Proof. Let $\mathfrak{T} = (T, \mathfrak{L})$ denote the common BHT induced by both orderings. For any vertex $v \in V$, v represents the birth of a 0-homology class, and its corresponding death occurs when it merges with an older component at the linking vertex $\mathfrak{L}(v)$.

Let (v, e) be the birth-death pair for v in the (G, f)-ordering, and (v, e') the pair in the (G, g)-ordering. Since $\mathfrak{L}(v)$ is the same in both trees, both e and e' must be edges connecting $\mathfrak{L}(v)$ to another node. Specifically, write $e = u\mathfrak{L}(v)$ and $e' = u'\mathfrak{L}(v)$.

By assumption, the edge-wise value comparisons are preserved between f and g, so we have $f(u) \leq f(\mathfrak{L}(v)) \Rightarrow g(u) \leq g(\mathfrak{L}(v))$. And, by definition, we have $g(u') \leq g(\mathfrak{L}(v))$, since $g(e') = g(\mathfrak{L}(v))$. It follows that $g(e) = g(e') = g(\mathfrak{L}(v))$.

Now, because $\prec'$ is compatible with $\prec$ (see Definition 6.8), it preserves the relative ordering of elements with equal function values according to $\prec$. Therefore, if $g(e) = g(e')$, but $e \prec e'$ while $e' \prec' e$, this would violate compatibility. Hence, we must have $e = e'$, and thus the birth-death pairs for v under both orderings are identical.

Since this argument holds for every vertex $v \in V$, we conclude that $\mathrm{BD}_0(G, \prec) = \mathrm{BD}_0(G, \prec')$.

It is now clear that $\mathrm{BD}_1(G, \prec) = \mathrm{BD}_1(G, \prec')$, since for any edge e under either ordering, it can be responsible for either the death of a 0-homology class or the birth of a 1-homology class. From the preceding argument, we know that the set of edges responsible for deaths is identical under both $\prec$ and $\prec'$. Therefore, the remaining edges must correspond to the births of 1-homology classes. Moreover, since we are considering the case without faces, all such classes are permanent cycles. Hence, each edge e that is not a death element appears in a birth-death pair of the form $(e, \emptyset)$, which concludes the proof. □

SECTION 7

The Low Persistence Filter

In this section, we introduce the framework of the *low-persistence filter*, designed for the analysis and processing of signals supported on graphs with faces. The goal of this filter is to attenuate topological noise while preserving meaningful homological features, thereby providing a tool for topological signal processing.

We begin by examining the base case of graphs without any face structure. In this setting, we make no assumptions about planarity, allowing us to consider arbitrary graphs. This case serves as a conceptual and notational foundation for the more involved scenarios that follow, and illustrates how persistent homology can be used to guide signal filtering in a simple case only, in which only 0-homology plays a role.

Building on this, we extend our attention to planar graphs with faces. In this case, we make use of the construction of the *induced graph* (Definition 4.7), which provides a dual representation that enables the application of persistent homology to both 0- and 1-dimensional features. This generalization allows us to design filters capable of targeting not only connected components but also 1-cycles.

A key challenge we address is the simultaneous filtering of both 0- and 1-dimensional homology in a coherent and topologically meaningful manner. This task is nontrivial due to the interactions between the two types of features and the constraints imposed by the planar structure, as we described in Section 5. We present a solution tailored to this case, culminating in our main theoretical contribution, formalized in Theorem 7.10, which characterizes the properties of the Low Persistence Filter capable of simultaneous filtering of 0- and 1-homology features.

7.1 One dimensional case

As previously mentioned, the absence of faces implies that we are currently considering signals defined on graphs of the form $(G = (V, E), f : V \to \mathbb{R})$. In this setting, the only homology classes with finite persistence arise in dimension zero, making the filtering process relatively straightforward. Specifically, 0-homology classes corresponding to short-lived features in the persistence diagram represent transient connected components in the sublevel set filtration—essentially, local minima that rapidly merge with older, more persistent regions.

These features are naturally represented by the Basin Hierarchy tree, denoted $\mathfrak{T} = (V, E_T)$. Each node $v \in V$ with low persistence corresponds to a short-lived basin in the sublevel set filtration. The children of v in the tree represent the constituent components of this basin—regions that merged into v during the filtration process.

Intuitively, by increasing the signal values within all vertices belonging to the basin associated with such a node v, up to a suitable threshold, we can effectively suppress the corresponding low-persistence interval in the persistence diagram.

In the following, we formalize this intuitive idea and define the corresponding filtering operation in precise terms.

Definition 7.1 . For $\boldsymbol{G} = ((V, E), f)$ a signal over a graph, let $\mathfrak{T} = (T, \mathfrak{L}) \in \mathrm{BHT}(\boldsymbol{G})$. We define the **0-Low**

Persistence Filter (0-LPF) by

$$ {}^{\mathfrak{T}}\mathcal{L}_0^{\epsilon} f(v) = \max\left(\left\{ f(\mathfrak{L}(u)) \,\middle|\, u \in \mathrm{A}_v^{\epsilon} \right\} \cup \left\{ f(v) \right\}\right), \tag{7.1} $$

where

$$ \mathrm{A}_v^{\epsilon} = \left\{ u \in V \,\middle|\, \begin{array}{l} u \preceq_T v \\ \mathrm{pers}_{\mathfrak{T}}(u) < \epsilon \end{array} \right\} \tag{7.2} $$

is the set of ancestors of $v \in V$ in $\mathfrak{T}$ with persistence less than ϵ. When it is clear from the context, we omit $\mathfrak{T}$ in the notation and write $\mathcal{L}_0^{\epsilon}$ for ${}^{\mathfrak{T}}\mathcal{L}_0^{\epsilon}$.

From the definition, we see

- if $0 < \mathrm{pers}_{\mathfrak{T}}(u) < \varepsilon$, then $f(u) < {}^{\mathfrak{T}}\mathcal{L}_0^{\epsilon} f(u) = {}^{\mathfrak{T}}\mathcal{L}_0^{\epsilon} f(\mathfrak{L}(u))$;
- if $\mathrm{pers}_{\mathfrak{T}}(u) > \varepsilon$, then $f(u) = {}^{\mathfrak{T}}\mathcal{L}_0^{\epsilon} f(u)$ and $f(\mathfrak{L}(u)) = {}^{\mathfrak{T}}\mathcal{L}_0^{\epsilon} f(\mathfrak{L}(u))$.

Definition 7.1 operates as follows. For each vertex $v \in V$, we identify all basins in the Basin Hierarchy tree in which v participates and whose persistence is less than a given threshold ε. Among these candidate basins, we select the one whose linking vertex has the highest function value. The signal value at v is then raised to match the value of this linking vertex. This ensures that the corresponding low-persistence feature—represented by the selected basin—is effectively eliminated from the persistence diagram, while also avoiding unnecessary alteration of other topological features.

While the definition characterizes the behavior of the filter at the level of individual vertices, the filtering process can be understood more structurally through the hierarchy of the tree. Specifically, as established in Lemma 6.5, the persistence of nodes is non-decreasing when traversing the Basin Hierarchy tree from any node toward the root. This observation allows us to simplify the filtering procedure: it suffices to consider only those nodes whose persistence is less than the threshold ε and whose parent has persistence greater than or equal to ε.

By applying the filter to such nodes—raising the signal values within their corresponding basins—we not only eliminate the targeted low-persistence features, but also implicitly suppress all of their descendant basins, which may themselves have nonzero but still sub-threshold persistence. This hierarchical process ensures that the filtering process is both efficient and structurally coherent (see Algorithm 4).

Although Algorithm 4 does not directly implement the formula specified in Definition 7.1, it yields the same filtered function, provided that the Basin Hierarchy Tree $\mathfrak{T}$ is the same. This equivalence follows from the fact that both approaches identify and modify the same set of basins with persistence below the threshold ε, applying the same update rule to the corresponding vertices.

While Algorithm 4 offers a more intuitive and operational perspective on the filtering process, Definition 7.1 is particularly useful in a theoretical context, as it formulates the filter at the vertex level. This vertex-wise definition facilitates formal reasoning and is well-suited for use in subsequent proofs and analytical arguments.

Lemma 7.2 . Let $\boldsymbol{G} = ((V, E), f)$ be a signal over a graph, and let $\mathfrak{T} = (T, \mathfrak{L})$ be its BHT with respect to a

Algorithm 4: 0-Low Persistence Filter

Require: Graph signal $\boldsymbol{G} = ((V, E), f)$ with a $\boldsymbol{G}$-ordering $\prec$, and threshold ε

```
Initialize Basin Hierarchy Tree: 𝔗 = (T, 𝔏) ← basin_hierarchy_tree(G, ≺)
Initialize filtered function: g ← f
Initialize stack with children of the root: stack ← stack[children(root)]
while stack is not empty do
    v ← stack.pop()
    if persistence(v) = 0 then
        continue
    else if persistence(v) < ε then
        g({u ∈ V | v ⪯_T u}) ← f(𝔏(v))
    else
        stack.push(children(v))
    end if
end while
return filtered function g
```

$\boldsymbol{G}$-ordering $\prec$. Let $\varepsilon > 0$ be a real number. For any function $g : V \to \overline{\mathbb{R}}$ and $v \in V$ define the transformation

$$g_v^{\mathfrak{T}}(u) = \begin{cases} g(\mathfrak{L}(v)), & \text{if } v \preceq_T u \\ g(u), & \text{otherwise} \end{cases}. \tag{7.3}$$

When it is clear from the context we omit the $\mathfrak{T}$ and write just g_v.

If $V_\varepsilon = \{v_1, \ldots, v_m\} \subset V$ is the set of all vertices such that

$$f(\mathfrak{L}(v_i)) - f(v_i) < \varepsilon,$$

then $\mathcal{L}_0^\varepsilon(f) = f_m$, in which $f_i = (f_{i-1})_{v_i}^{\mathfrak{T}}$ with $f_0 = f$. In other words,

$$\mathcal{L}_0^\varepsilon(f) = (\ldots(f_{v_1})_{v_2}\ldots)_{v_m}$$

independent of the ordering of V_ε.

In other words, Lemma 7.2 is showing that we can compute the low-persistence filter by eliminating one low persistence feature at a time.

Proof. Given $u \in V$, we will prove $\mathcal{L}_0^\varepsilon f(u) = f_m(u)$. Let $\mathrm{A}_u^\varepsilon \subset V_\varepsilon$ be as in (7.2).

If $\mathrm{A}_u^\varepsilon = \emptyset$, then it is trivial to see $f_m(u) = f(u) = \mathcal{L}_0^\varepsilon f(u)$. Otherwise, by Item 1 of Proposition 6.6 and (7.1), we have

$$\mathcal{L}_0^\varepsilon f(u) = \max_{v \in \mathrm{A}_u^\varepsilon} f(\mathfrak{L}(v)).$$

Let $v_j \in \mathrm{A}_u^\varepsilon$ be the vertex such that $v_j \preceq_T w$ for any $w \in \mathrm{A}_u^\varepsilon$. By Lemma 6.5, we have that

$$f(\mathfrak{L}(v_j)) = \max_{v \in \mathrm{A}_u^\varepsilon} f(\mathfrak{L}(v)). \tag{7.4}$$

Furthermore, if $f(\mathfrak{L}(w)) < f(\mathfrak{L}(v_j))$ for some $w \in \mathrm{A}_u^\varepsilon$, then $v_j \prec_T w$ and $C(w, G_{f(\mathfrak{L}(w))}) = C(\mathfrak{L}(w), G_{f(\mathfrak{L}(w))})$,

hence, by Item 2 of Proposition 6.6 we have $v_j \prec_T \mathfrak{L}(w)$.

We have that $f_i(\mathfrak{L}(v_j)) = f(\mathfrak{L}(v_j))$ for all $i = 1, \dots, m$, since there are no ancestors of v_j with persistence less than ε. Furthermore, we have $f_j(v_i) = f_j(\mathfrak{L}(v_i)) = f(\mathfrak{L}(v_j))$ for all $i = 1, \dots, m$, since all v_i and $\mathfrak{L}(v_i)$ are descendants of v_j. Consequently, we have $f_k(v_i) = f_k(\mathfrak{L}(v_i)) = f(\mathfrak{L}(v_j))$ for any $k \geq j$. And finally, we conclude that $f_m(u) = f(\mathfrak{L}(v_j)) = \max_{v \in \mathrm{A}_u^\varepsilon} f(\mathfrak{L}(v)) = \mathcal{L}_0^\varepsilon f(u)$. □

The following proposition establishes that the Basin Hierarchy Tree (BHT) remains unaltered after the application of the Low Persistence Filter. This stability property is crucial for controlling persistent 0-homology and plays a central role in the proof of Theorem 7.5.

Theorem 7.3. Let $\varepsilon \geq 0$, and let $\mathfrak{T}$ denote the BHT with respect to the (G, f)-ordering $\prec$. Define $g = {}^{\mathfrak{T}}\mathcal{L}_0^\varepsilon f$ as the result of applying the 0-Low Persistence Filter to f, and let $\prec'$ be the (G, g)-ordering compatible with $\prec$ (see Definition 6.8). Then, $\mathfrak{T}$ is also a BHT with respect to the ordering $\prec'$.

Proof. For a signal $\boldsymbol{G} = (G, f)$, let $\mathfrak{T} = (T, \mathfrak{L})$ be the BHT with respect to a $\boldsymbol{G}$-ordering $\prec$. Fix $\varepsilon \geq 0$ and $v \in V$, and define f_v as in Equation (7.3). From Lemma 7.2, we see that it is sufficient to prove that $\mathfrak{T}$ is a BHT of (G, f_v).

Let $U = \{\sigma \in V \cup E \mid f_v(\sigma) = f(\mathfrak{L}(v))\}$, and let $\prec'$ be the (G, f_v)-ordering defined as in (6.6).

We prove $\mathfrak{T}$ is the BHT of (G, f_v) with respect to $\prec'$ by showing that for a pair $(w, e) \in V \times E$ satisfying

$$\min C(w, G_{\preceq e}) = p(w) \prec w = \min C(w, G_{\prec e}) \tag{7.5}$$

with the ordering $\prec$, it holds that

$$\min C(w, G_{\preceq' e}) = p(w) \prec' w = \min C(w, G_{\prec' e}) \tag{7.6}$$

with the ordering $\prec'$.

If $f(\mathfrak{L}(w)) = f(e) \geq f(\mathfrak{L}(v))$, then the result is trivial since $C(w, G_{\prec e}) = C(w, G_{\prec' e})$ $C(w, G_{\preceq e}) = C(w, G_{\preceq' e})$. If $f(\mathfrak{L}(w)) < f(\mathfrak{L}(v))$, we have $w = \min C(w, G_{\prec' e})$ since $C(w, G_{\prec' e}) \subset C(w, G_{\prec e})$. We prove $\min C(w, G_{\preceq' e}) = p(w)$ by considering the two following cases:

(i) $w \in V \setminus U$: We must have $p(w) \in V \setminus U$, otherwise we would have $w \prec_T v \preceq_T p(w)$. We know from Item 3 of Proposition 6.6 that there is a path from w to $p(w)$ with all vertices being lower than $\mathfrak{L}(v)$ with respect to $\prec$. This path cannot have any vertex bigger than v with respect to $\prec_T$; otherwise, $v \prec_T w$ since $C(v, G_{\prec \mathfrak{L}(v)}) = C(w, G_{\prec \mathfrak{L}(v)})$. This contradicts the fact that $w \notin U$. With the same argument we can see that $\mathfrak{L}(v)$ connects to $p(w)$ via a path that does not intersect U. By concatenating the two paths, we get a path connecting w to $p(w)$ that only has vertices of $V \setminus U$. This path is contained in $G_{\preceq' e}$ since only the elements of U can become greater in the new ordering $\prec'$.

(ii) $w \in U$: Vertices of U can only be connected to vertices $x \in V$ with $f(x) \leq f(\mathfrak{L}(v))$ through edges with $f_v(e') = f(\mathfrak{L}(v))$, and these edges preserve their relative ordering in the new ordering. Hence, w and $p(w)$ are in the same component in $G_{\preceq' e}$.

□

From the theorem above, we observe that different levels of filtering can be applied efficiently, as the BHT is preserved at each stage.

Theorem 7.4. Let $f : V \to \mathbb{R}$ be a signal over a graph $G = (V, E)$, and $\prec$ a (G, f)-ordering, then for any $(G, \mathcal{L}_0^\varepsilon f)$-ordering $\prec'$ compatible with $\prec$ we have that the birth-death pairs of the two orderings are the same $\mathrm{BD}(G, \prec) = \mathrm{BD}(G, \prec')$.

Proof. The proof is basically a consequence of Proposition 6.9. Next we confirm that in this case we have all the assumptions made in the proposition.

In the proposition notation let f be our signal in $G = (V, E)$ and $g = \mathcal{L}_0^\varepsilon f$, in which the filter is computed based on the BHT $\mathfrak{T} = (T, \mathfrak{L})$.

Let $e = uv$ be an edge in E with $f(u) \leq f(v)$, since $\mathcal{L}_0^\varepsilon$ can only lift values we only need to worry about the case where the value of u is lifted to confirm it cannot be lifted more than the value of v. If u is lifted it means it is part of a basin with persistence less than ε. Let $w \prec_T u$ be the root of this basin. If $w \prec_T v$, then we have $g(u) = g(v) = f(\mathfrak{L}(w))$. And if $w \not\prec_T v$, then $g(u) = f(\mathfrak{L}(w)) \leq f(v) \leq g(v)$.

Since $\prec'$ is compatible with $\prec$, by Theorem 7.3 we have that they share the same BHT $\mathfrak{T}$. We conclude by Proposition 6.9 that $\mathrm{BD}(G, \prec) = \mathrm{BD}(G, \prec')$. □

With all the necessary components in place, we now state and prove the main theorem in this simplified case without faces.

Theorem 7.5. Given a signal over a graph (G, f), for any $\mathfrak{T} \in BHT(G, f)$ we have that

$$\mathrm{PD}_0(G, {}^{\mathfrak{T}}\mathcal{L}_0^\varepsilon f) \approx \mathrm{PD}_0(G, f)_{\geq \varepsilon}, \tag{7.7}$$

more specifically ${}^{\mathfrak{T}}\mathcal{L}_0^\varepsilon f \in \mathcal{S}(G)_f^\varepsilon$. Furthermore,

$$0 \leq {}^{\mathfrak{T}}\mathcal{L}_0^\varepsilon f(v) - f(v) < \max_{I \in \mathrm{PD}_0(\mathcal{G}, f)_{<\varepsilon}} \mathrm{pers}(I) \leq \varepsilon, \tag{7.8}$$

for all $v \in V$. In particular, we have that

$$|\hat{\mathcal{L}}_0^\varepsilon f - f|_\infty < \frac{\varepsilon}{2}, \tag{7.9}$$

where $\hat{\mathcal{L}}_0^\varepsilon f(v) = {}^{\mathfrak{T}}\mathcal{L}_0^\varepsilon f(v) - \frac{\varepsilon}{2}$.

Proof. Theorem 6.4, Lemma 6.5, and Theorem 7.3 establish that branches in $\mathfrak{T} \in BHT(G, f)$ with nontrivial persistence lower than ε are eliminated by $\mathcal{L}_0^\varepsilon$, while those with persistence greater than or equal to ε remain unchanged. This implies Equation (7.7). Furthermore, Theorem 7.4 guarantees that the birth-death pairs are preserved, hence ${}^{\mathfrak{T}}\mathcal{L}_0^\varepsilon f \in \mathcal{S}(G)_f^\varepsilon$.

Since $f(v) \leq f(\mathfrak{L}(u))$ for any $u \prec_T v$ by Lemma 6.5, we have $0 \leq {}^{\mathfrak{T}}\mathcal{L}_0^\varepsilon f(v) - f(v)$. Since $\mathcal{L}_0 f(v)$ is either $f(v)$ or $f(\mathfrak{L}(u))$ for some $u \prec_T v$ with $f(\mathfrak{L}(u)) - f(u) < \varepsilon$, we conclude that $f(\mathfrak{L}(u)) - f(v) < \varepsilon$, and therefore $\mathcal{L}_0^\varepsilon f(v) - f(v) < \varepsilon$.

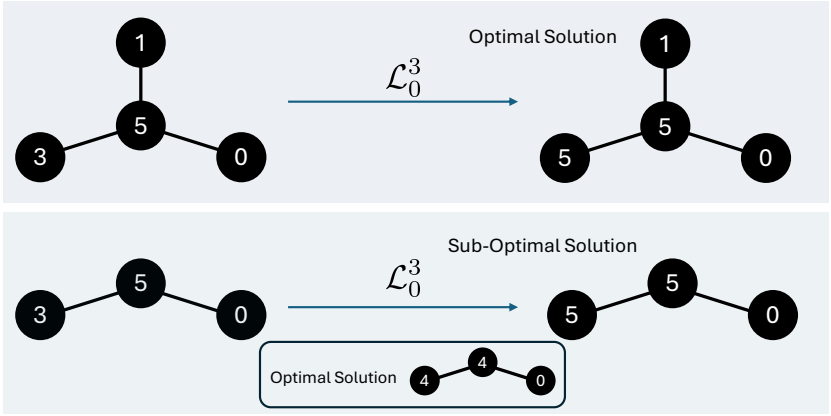


Figure 7.1: An example demonstrating that the optimal solution of Problem 5.1 is not always attained by the 0-Low Persistence Filter. However, if ε is considered negligible (representing the size of noisy features in the persistence diagrams), then the solution produced by the Low Persistence Filter serves as a sufficiently accurate approximation.

Now, let $I_{\max} \in \mathrm{PD}_0(\mathcal{G}, f)$ denote the interval of largest persistence among those with $\mathrm{pers}(I) < \varepsilon$. Then for any ε' satisfying $\mathrm{pers}(I_{\max}) < \varepsilon' < \varepsilon$, we have $\mathcal{L}_0^{\varepsilon'} f = \mathcal{L}_0^{\varepsilon} f$, and by the previous part of the proof, this yields Equation (7.8).

The last part follows easily, since Equation (7.8) implies

$$-\frac{\varepsilon}{2} \leq {}^{\mathfrak{T}}\mathcal{L}_0^{\varepsilon} f(v) - \frac{\varepsilon}{2} - f(v) < \frac{\varepsilon}{2},$$

□

The previous theorem shows that for a graph with no faces, there always exists a solution to Problem 5.1, since $\mathcal{L}_0^{\varepsilon} f \in \mathcal{S}(G)_f^{\varepsilon}$. It is important to emphasize that $\mathcal{L}_0^{\varepsilon} f$ may not be the exact solution to Problem 5.1; that is, there may exist some $g \in \mathcal{S}(G)_f^{\varepsilon}$ such that $|g - f|_\infty < |\mathcal{L}_0^{\varepsilon} f - f|_\infty$. However, finding this exact solution can be highly complex, depending on both the signal and the graph structure (see Figure 7.1). The key result of Theorem 7.5 is that $\mathcal{L}_0^{\varepsilon} f$ provides a good approximation, as it guarantees that $|\mathcal{L}_0^{\varepsilon} f - f|_\infty < \varepsilon$.

7.2 Two dimensional case

We now build upon the previous definition, together with the concept of the induced graph of a graph with faces (see Definition 4.7), to extend the filter to a more general setting.

Definition 7.6 . For a signal over a planar graph with faces $\boldsymbol{\mathcal{G}} = (\mathcal{G}, f)$, define:

$$\mathcal{L}_0^{\varepsilon} f = \left(\mathcal{L}_0^{\varepsilon} f_{G[\mathcal{E}]}\right)\Big|_V, \tag{7.10}$$

$$\mathcal{L}_1^{\varepsilon} f = -\left(\mathcal{L}_0^{\varepsilon}(-f_{G[\mathcal{E}]})\right)\Big|_V. \tag{7.11}$$

Note that, although we omit explicit references to the BHTs, each of the definitions above implicitly depends on a choice of ordering. Specifically, the construction of $\mathcal{L}_0^{\varepsilon}$ requires a $(G[\mathcal{E}], f_{G[\mathcal{E}]})$-ordering, while the construction of $\mathcal{L}_1^{\varepsilon}$ requires a $(G[\mathcal{E}], -f_{G[\mathcal{E}]})$-ordering. Each of these orderings induces a distinct BHT, which in turn determines the corresponding low-persistence filter.

The idea behind the above definition is that the 0- and 1-persistent homology of a graph with faces is determined by the signal and its dual on the induced graph. This observation allows us to construct filters that remove low-persistence 0-homology classes in both cases, which correspond to filtering low-persistence 0- and 1-homology classes in the original signal, as we will prove next.

The low-persistence filter in Equation (7.10) (respectively Equation (7.11)) removes 0-homology (respectively 1-homology) classes with persistence less than ε. As a consequence of Theorems 4.8 and 7.5, we obtain the following corollary:

Corollary 7.7 . For $i = 0, 1$, we have:

$$\mathrm{PD}_i(\mathcal{G}, \mathcal{L}_i^\varepsilon f) \approx \mathrm{PD}_i(\mathcal{G}, f)_{\geq \varepsilon}$$

and

$$\|\mathcal{L}_i^\varepsilon f - f\|_\infty < \varepsilon.$$

Although the persistence intervals in $\mathrm{PD}_0(\mathcal{G}, f)_{\geq \varepsilon}$ (respectively $\mathrm{PD}_1(\mathcal{G}, f)_{\geq \varepsilon}$) are preserved under the application of $\mathcal{L}_0^\varepsilon$ (respectively $\mathcal{L}_1^\varepsilon$), the intervals in $\mathrm{PD}_1(\mathcal{G}, f)_{\geq \varepsilon}$ (respectively $\mathrm{PD}_0(\mathcal{G}, f)_{\geq \varepsilon}$) may still be affected.

Note that solving Problem 5.4 requires more than simply applying the composition $\mathcal{L}_1^\varepsilon \mathcal{L}_0^\varepsilon$. As illustrated in Figure 7.2, we have $\mathcal{L}_1^\varepsilon \mathcal{L}_0^\varepsilon f \notin \tilde{S}(\mathcal{G})_f^\varepsilon$, since some elements with persistence less than ε remain in the resulting persistence diagram.

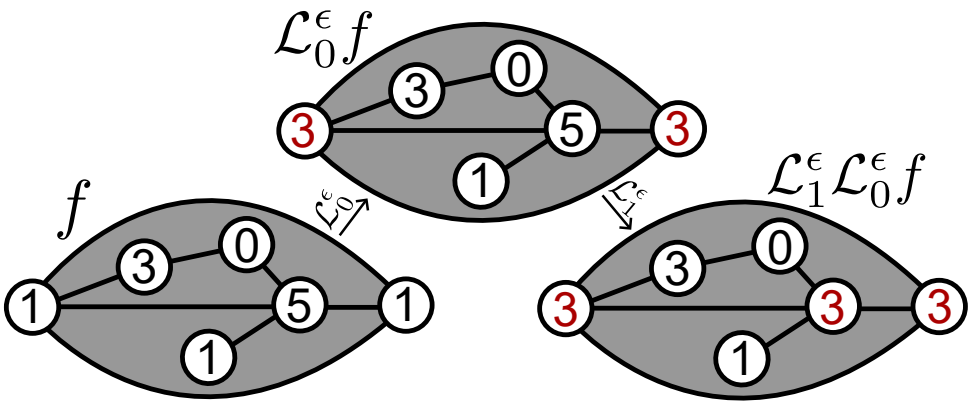


Figure 7.2: Signal over a planar graph with faces $(\mathcal{G}, f)$ in which $\mathrm{PD}_0(\mathcal{L}_1^\varepsilon \mathcal{L}_0^\varepsilon f)_{<\varepsilon} = \{[1, 3)\} \neq \emptyset$. Considering $2 < \varepsilon < 4$.

To address this problem, we consider an iterative application of $\mathcal{L}_0^\varepsilon$ and $\mathcal{L}_1^\varepsilon$:

$$\mathcal{L}_*^{\varepsilon,k} \triangleq \underbrace{\ldots \mathcal{L}_0^\varepsilon \mathcal{L}_1^\varepsilon \mathcal{L}_0^\varepsilon}_{k \text{ times}}. \tag{7.12}$$

Lemma 7.9 ensures that this process stabilizes after finitely many steps, and Theorem 7.10 shows that it yields a solution to Problem 5.4. Before stating and proving those two results we need the following propositon showing that applying $\mathcal{L}_1^\varepsilon$ to a function f can only decrease the persistence of a 0-homology class generated by any vertex.

Proposition 7.8 . Let f a signal over a graph $G = (V, E)$, and let $\mathfrak{T}_0 = (V, E_{T_0}, \mathfrak{L}_0)$ be the BHT with respect to a (G, f)-ordering $\prec$, and $\mathfrak{T}_1 = (V, E_{T_1}, \mathfrak{L}_1)$ be the BHT with respect to a $(G, -f)$-ordering $\prec_1$. Then, there exists $\mathfrak{T}_0' = (V, E_{T_0'}, \mathfrak{L}_0') \in \mathrm{BHT}(G, -^{\mathfrak{T}_1}\mathcal{L}_0^{\varepsilon}(-f))$ such that $\mathrm{pers}_{\mathfrak{T}_0'}(u) \leq \mathrm{pers}_{\mathfrak{T}_0}(u)$ for all $u \in V$ and

$$w \prec_{T_o'} u \implies f(w) \leq f(u). \tag{7.13}$$

Proof. From Lemma 7.2, it is sufficient to prove the assertion for $-(-f)_v^{\mathfrak{T}_1}$ in place of $-^{\mathfrak{T}_1}\mathcal{L}_0^{\varepsilon}(-f)$ for each $v \in V$.

For ease of notation, define

$$f_{-v}(u) = -(-f)_v^{\mathfrak{T}_1}(u) = \begin{cases} f(\mathfrak{L}_1(v)) & \text{if } v \prec_{T_1} u, \\ f(u) & \text{otherwise.} \end{cases}$$

Let $U = \{\sigma \in V \cup E \mid f_{-v}(\sigma) = f(\mathfrak{L}_1(v))\}$ and let $\prec_0'$ be the (G, f_{-v})-ordering defined as in (6.6). Notice that only vertices and edges with f values above $f(\mathfrak{L}_1(v))$ can have their values altered in f_{-v} by being pushed down to $f(\mathfrak{L}_1(v))$. If $w \in V$ is such that:

1. $f(\mathfrak{L}_0(w)) \leq f(\mathfrak{L}_1(v))$, then the edge e such that

   $$\min C(w, G_{\preceq_0 e}) = p(w) \prec_0 w = \min C(w, G_{\prec_0 e}),$$

   also satisfies $f(e) \leq f(\mathfrak{L}_1(v))$. Since the vertices and edges up to that point are exactly the same and in the same order we have

   $$\min C(w, G_{\preceq_0' e}) = p(w) \prec_0' w = \min C(w, G_{\prec_0' e}),$$

   hence $\mathrm{pers}_{\mathfrak{T}_0'}(w) = \mathrm{pers}_{\mathfrak{T}_0}(w)$.

2. $f(\mathfrak{L}_0(w)) > f(\mathfrak{L}_1(v))$, then there is a path connecting w and $p_0(w)$ passing through $\mathfrak{L}_0(w)$ with values in $[f(p_0(w)), f(\mathfrak{L}_0(w))]$. Then at most some vertices in this path are pushed down in f_{-v}, making w and $p(w)$ connect before or at $f(\mathfrak{L}_0(w))$. Furthermore, by definition of $\prec_0'$, we have that w keeps its relative ordering to its neighbouring vertices, so it is still a local minimum for f_{-v}. Now, we have two cases:

   (a) If $f(\mathfrak{L}_1(v)) < f(p(w))$ and $v \prec_{T_1} w$, then the path connecting w and $p(w)$, described above, implies that $v \prec_{T_1} p(w)$ (Item 2 from Proposition 6.6), hence $w, p(w) \in U$ (and the whole path connecting them is also in U), and by definition of $\prec_0'$, we have $p(w) \prec_0' w$.

   (b) If either $f(\mathfrak{L}_1(v)) < f(p(w))$ or $v \prec_{T_1} w$ is not true, then it is trivial that $p(w) \prec_0' w$.

   Hence, if $f_{-v}(w) = f(w)$, the connected component of w merges with the component of $p(w)$ at or before $f(\mathfrak{L}_0(w))$ (since this is the maximum value of f_{-v} in the path described above), and if $f_{-v}(w) < f(w)$, we have $f_{-v}(w') \leq f_{-v}(w)$, for all w' in the path between w and $p(w)$. So we conclude $\mathrm{pers}_{\mathfrak{T}_0'}(w) \leq \mathrm{pers}_{\mathfrak{T}_0}(w)$.

□

Notice that Proposition 7.8 not only implies that applying $\mathcal{L}_1^{\varepsilon}$ to f can only decrease the persistence of

0-homology classes, but also that applying $\mathcal{L}_0^\varepsilon$ to f can only decrease the persistence of 1-homology classes, due to the duality between the two definitions.

Lemma 7.9 . For any $\mathcal{G} = (\mathcal{G}, f)$ and $n \in \{0, 1\}$, we have $\mathcal{L}_n^\varepsilon f \lesssim f$. Moreover,

$$\mathcal{L}_n^\varepsilon f \neq f \iff |\mathrm{PD}_n(\mathcal{G}, \mathcal{L}_n^\varepsilon f)_{>0}| < |\mathrm{PD}_n(\mathcal{G}, f)_{>0}|.$$

In other words, applying $\mathcal{L}_n^\varepsilon$ strictly decreases the cardinality of the multiset $\mathrm{PD}_n(\mathcal{G}, f)_{>0}$—counting interval multiplicities—if and only if the function is altered.

Proof. Recall that, by definition

$$\mathcal{L}_0^\varepsilon f = \left(\mathcal{L}_0^\varepsilon f_{G[\mathcal{E}]}\right)\Big|_V.$$

Thus, by Theorems 4.8 and 7.5, we have

$$\mathrm{PD}_0(\mathcal{G}, \mathcal{L}_0^\varepsilon f) = \mathrm{PD}_0(\mathcal{G}, f)_{>\varepsilon}.$$

Moreover, by Proposition 7.8 and (4.7), we see that applying $\mathcal{L}_0^\varepsilon$ to f does not increase the persistence of intervals in $\mathrm{PD}_1(\mathcal{G}, f)$. Hence, $\mathcal{L}_0 f \lesssim f$. The proof for $\mathcal{L}_1^\varepsilon$ follows by an analogous argument. □

Lemma 7.9 is crucial for establishing that iteratively applying $\mathcal{L}_0^\varepsilon$ and $\mathcal{L}_1^\varepsilon$ in succession will always stabilize, as we are working with graphs with faces consisting of finitely many vertices, edges, and faces.

We now state and prove the main theorem of this thesis. This result provides a method for obtaining a good approximation to a solution of Problem 5.4, which, in many cases, coincides with an exact solution.

Theorem 7.10. For any $\mathcal{G} = (\mathcal{G}, f)$, there exists $k \geq 0$ such that $\mathcal{L}_*^{\varepsilon,k} f = \mathcal{L}_*^{\varepsilon,k+1} f$. Denote by k_o the smallest such k. We have $\mathcal{L}_*^{\varepsilon,k_o} f = \mathcal{L}_*^{\varepsilon,k} f$ for any $k \geq k_o$. We define

$$\mathcal{L}_*^\varepsilon f := \mathcal{L}_*^{\varepsilon,k_o} f. \tag{7.14}$$

Then, we have $|f - \mathcal{L}_*^\varepsilon f|_\infty < \varepsilon$, $\mathrm{PD}_n(\mathcal{L}_*^\varepsilon f)_{\leq\varepsilon} = \emptyset$ for $n = 0, 1$, and $\mathcal{L}_*^\varepsilon f \lesssim f$. Consequently we have

$$\|f - \mathcal{L}_*^\varepsilon\|_\infty < \max_{I \in \mathrm{PD}_*(\mathcal{G}, f)_{<\varepsilon}} \mathrm{pers}(I) \leq \varepsilon$$

Proof. The stability, specifically the existence of k_o, is a consequence of Lemma 7.9 since $|\mathrm{PD}_n(\mathcal{G}, f)_{>0}|$ is finite. To see that $|\mathcal{L}_*^\varepsilon f - f|_\infty < \varepsilon$, we show that $\mathcal{L}_*^\varepsilon f(u) < f(u) + \varepsilon$, for any $u \in V$. The proof that $f(u) - \varepsilon < \mathcal{L}_*^\varepsilon f(u)$ is analogous.

Let

$$\mathfrak{T}_0^k = (V', E_{T_0^k}, \mathfrak{L}_0^k) \in \mathrm{BHT}(G[\mathcal{E}], (\mathcal{L}_*^{\varepsilon,k} f)_{G[\mathcal{E}]})$$

and

$$\mathfrak{T}_1^k = (V', E_{T_1^k}, \mathfrak{L}_1^k) \in \mathrm{BHT}(G[\mathcal{E}], -(\mathcal{L}_*^{\varepsilon,k} f)_{G[\mathcal{E}]},)$$

for $k \in \mathbb{Z}_{\geq 0}$, where V' are the vertices of $G[\mathcal{E}]$. By Proposition 7.8, we can demand these BHTs to satisfy

$$\operatorname{pers}_{\mathfrak{T}_n^{k+1}}(u) \leq \operatorname{pers}_{\mathfrak{T}_n^{k}}(u) \tag{7.15}$$

for $n = 0, 1$ and $u \in V'$. Furthermore, if

$$\mathcal{L}_*^{\varepsilon,k+1} f(u) > \mathcal{L}_*^{\varepsilon,k} f(u)$$

then $\mathcal{L}_*^{\varepsilon,k+1} = \mathcal{L}_0^{\varepsilon} \mathcal{L}_*^{\varepsilon,k}$, and the value of u is pushed up resulting in $\operatorname{pers}_{\mathfrak{T}_0^{k+1}}(u) = 0$. This, together with (7.15), implies $\operatorname{pers}_{\mathfrak{T}_0^{j}}(u) = 0$ for any $j > k$. Hence, for any $u \in V'$ we have

$$0 < \operatorname{pers}_{\mathfrak{T}_0^{k}}(u) \implies \mathcal{L}_*^{\varepsilon,k} f(u) \leq f(u). \tag{7.16}$$

Let ℓ be the highest index for which $\mathcal{L}_*^{\varepsilon,\ell+1} f(u) > \mathcal{L}_*^{\varepsilon,\ell} f(u)$ so that we have $\mathcal{L}_*^{\varepsilon} f(u) \leq \mathcal{L}_*^{\varepsilon,\ell+1} f(u)$. Moreover, there is $w \in V'$ such that $w \prec_{T_0^\ell} u$ and $0 < \operatorname{pers}_{T_0^\ell}(w) < \varepsilon$. By Equation (7.16) we conclude that $\mathcal{L}_*^{\varepsilon,\ell} f(w) \leq f(w)$, and from Equation (7.13) we get $f(w) \leq f(u)$. Since $\operatorname{pers}_{T_0^\ell}(w) < \varepsilon$ we have $\mathcal{L}_*^{\varepsilon,\ell+1} f(w) < \mathcal{L}_*^{\varepsilon,\ell} f(w) + \varepsilon$. All together, we have

$$\begin{aligned} \mathcal{L}_*^{\varepsilon} f(u) &\leq \mathcal{L}_*^{\varepsilon,\ell+1} f(u) = \mathcal{L}_*^{\varepsilon,\ell+1} f(w) \\ &< \mathcal{L}_*^{\varepsilon,\ell} f(w) + \varepsilon \leq f(w) + \varepsilon \\ &\leq f(u) + \varepsilon. \end{aligned}$$

□

Summary

In this chapter, we introduced and developed tools for constructing Low Persistence Filters, with a focus on hierarchical and topological structures. We began by presenting the *Basin Hierarchy Tree* (BHT), a data structure specifically designed to encode information derived from persistent homology. The BHT builds upon the classical Union-Find algorithm, which we briefly reviewed—emphasizing its optimized variant that incorporates rank-based union and path compression. However, due to the hierarchical constraints inherent in the topological structure of components, only path compression could be effectively employed in our adaptation of the BHT.

After formally defining the BHT, we constructed a Low Persistence Filter for the one-dimensional case, corresponding to signals on graphs. We showed that, in this restricted setting, the filter yields an approximate solution to the original, stricter formulation of our problem (Problem 5.1). This result highlights that the main challenges in Low Persistence Filtering emerge when addressing multiple homological dimensions simultaneously.

To tackle this broader setting, we developed a more general filtering method capable of handling both 0- and 1-dimensional homological features concurrently. Unlike the previous case, this filter provides a an approximate solution only to a relaxed formulation of the problem (Problem 5.4), reflecting the increased complexity arising from the interplay between different homological dimensions.

Chapter IV

Applications and Extensions

Now that we have developed the Low Persistence Filter (LPF), we turn our attention to a series of examples and applications that illustrate its behavior and potential. To begin, we introduce the open-source software package we developed, which provides efficient implementations for computing Basin Hierarchy Trees (BHTs) and applying the Low Persistence Filter.

We then proceed to a collection of illustrative examples designed to demonstrate the qualitative and quantitative effects of the LPF in a variety of settings. These examples include applications to line graphs, natural images, surfaces, and geometric meshes. By examining how the LPF performs in these contexts, we gain insight into its versatility and its limitations. These concrete cases showcase the filter's ability to simplify data while preserving essential structure.

In Section 10, we extend the LPF framework with a technique we refer to as *size-aware filtering*. The main idea here is to incorporate component size as an additional criterion in the filtering process, in contrast to relying solely on persistence. This modification is motivated by the observation that persistence alone may not fully capture the significance of certain features, particularly in spatial or image-based data. The Basin Hierarchy Tree structure is especially well-suited for this adaptation, as it naturally encodes both topological and geometric information, making the implementation of size-aware strategies conceptually and practically straightforward.

Next, in Section 11, we explore the extraction of summary statistics from the Basin Hierarchy Tree. These statistics can serve as compact, informative descriptors of the underlying data and may be particularly valuable in downstream tasks such as feature engineering and machine learning. By summarizing key structural aspects of the data in a form that is easy to compute and interpret, the BHT provides a lens for data analysis that goes beyond traditional descriptors.

Overall, this chapter provides a multifaceted view of the Low Persistence Filter and the Basin Hierarchy Tree, showcasing their utility through software, examples, theoretical extensions, and analytical perspectives.

SECTION 8

Low persistence filter implemention in Python/C++

The Low Persistence Filter, as developed and described in detail in Chapter III, has been implemented as part of an open-source software package, which is available at `https://github.com/mvlier/topapprox`. This implementation serves both as a practical demonstration of the theoretical framework introduced in this thesis and as a tool for further experimentation and evaluation. The core functionality is written in Python to facilitate rapid development and ease of integration with existing scientific computing libraries. To address performance bottlenecks associated with computationally intensive tasks—most notably the construction of the Basin Hierarchy Tree—we employed C++ for critical subroutines. These components are seamlessly integrated with the Python code using appropriate interfacing tools to ensure both computational efficiency and usability. This hybrid approach allows for scalable experimentation while preserving code clarity and modularity.

SECTION 9

Applying the Low Persistence Filter

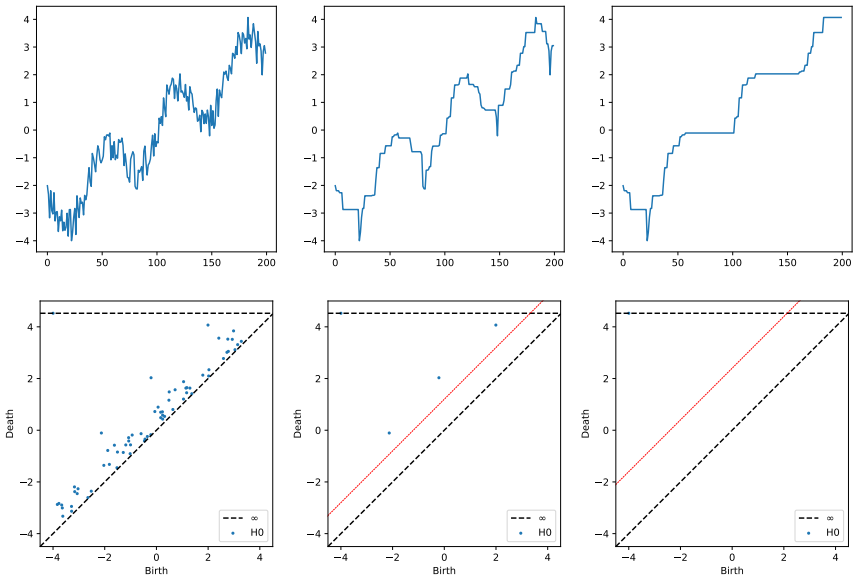


Figure 9.1: A 1D signal over a line graph (left) is filtered using the LPF with different thresholds (middle, right). The corresponding persistence diagrams are shown in the lower row.

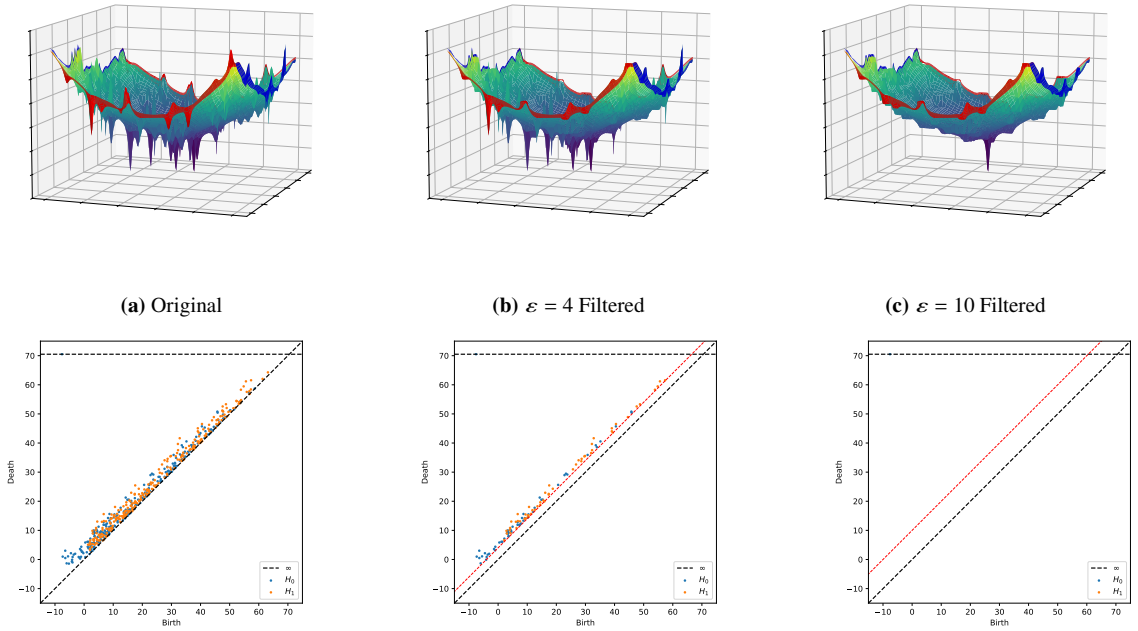
Example 9.1 . We begin with a simple example to illustrate the net effect of applying the Low Persistence Filter (LPF). Specifically, we consider a one-dimensional signal, such as a time series, which can be represented as a function over a line graph (Figure 9.1, left).

In this 1D setting, the only relevant topological features arise in 0-homology (connected components), allowing us to focus on the simpler version of our filter—the 0-LPF.

As shown in Figure 9.1, the original signal exhibits high-frequency fluctuations. Applying the LPF with various thresholds progressively "smooths" the signal, attenuating these fluctuations (middle and right panels).

The corresponding persistence diagrams are displayed in the bottom row of the figure. Each point in the diagram represents a local minimum of the signal, paired with its persistence value (i.e., the difference in function value relative to an adjacent local maximum).

This example underscores the importance of selecting an appropriate LPF threshold. An excessively high threshold (right) may remove significant features, whereas a suitably chosen threshold (middle) preserves the underlying trend while effectively suppressing noise.



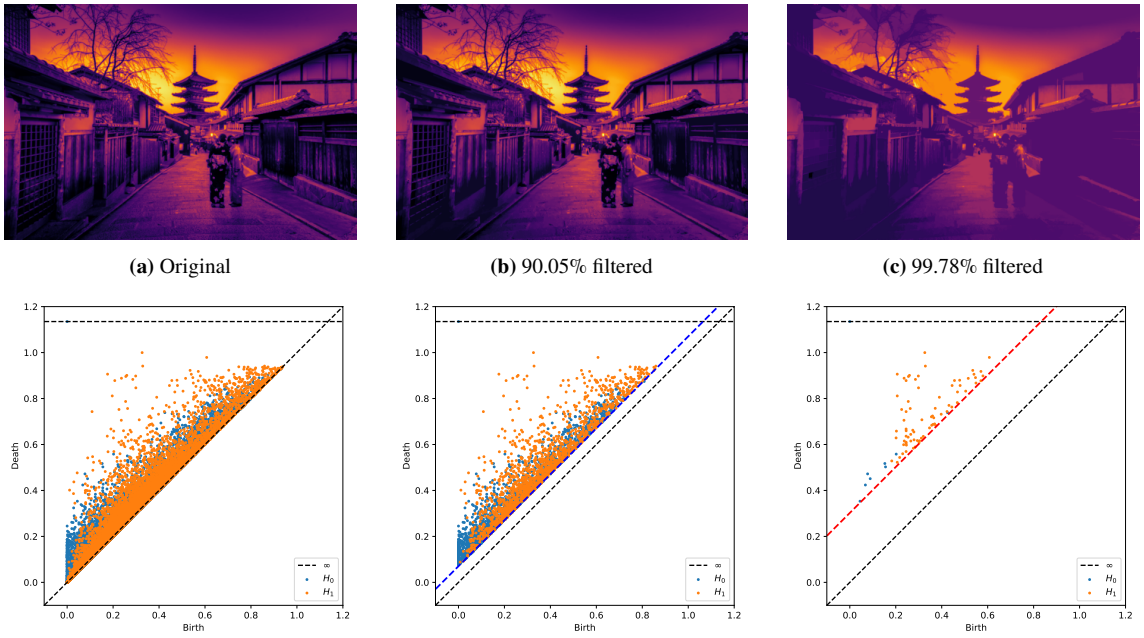
(a) Original **(b)** $\varepsilon = 4$ Filtered **(c)** $\varepsilon = 10$ Filtered

(d) Original PD **(e)** $\varepsilon = 4$ Filtered PD **(f)** $\varepsilon = 10$ Filtered PD

Figure 9.2: A modified Shekel function (left column) and its filtered versions using the Low Persistence Filter. Each plot's corresponding persistence diagram is shown below. The threshold in the rightmost case eliminates all finite intervals, leaving only the global minimum.

Example 9.2 . A two-dimensional function can be interpreted as a scalar field defined over a grid graph with faces, where each node corresponds to a sample point and the edges and faces encode neighborhood relationships. Figure 9.2 illustrates a modified version of the Shekel function, a classical test function in global optimization known for its densely distributed local extrema. These characteristics make it a particularly challenging case for optimization algorithms and an ideal candidate for demonstrating the behavior of the Low Persistence Filter (LPF).

When the LPF is applied with a sufficiently large threshold, it effectively suppresses local extrema with low persistence, thereby eliminating excessive topological features and revealing the global structure of the function. In the figure, we present two levels of filtering: the first corresponds to a moderate threshold that selectively removes low-persistence features while retaining prominent structures, and the second applies a higher threshold that completely eliminates all finite persistence intervals. This progressive simplification highlights the LPF's ability to act as a topological denoising tool, emphasizing persistent features that reflect the dominant geometric or functional trends in the data.



(a) Original **(b)** 90.05% filtered **(c)** 99.78% filtered

(d) Original PD **(e)** 90.05% filtered PD **(f)** 99.78% filtered PD

Figure 9.3: A natural image and its filtered versions using the Low Persistence Filter. The plot below each image shows its corresponding persistence diagram.

Example 9.3 . An image can be interpreted as a signal defined on a grid graph with square faces, where pixel intensities represent the signal values. Figure 9.3 illustrates the application of the LPF to a natural image.

The original image and its corresponding persistence diagram are shown in the left column. In the middle and right columns, the blue and red dashed lines in the persistence diagrams indicate the thresholds corresponding to the removal of 90.51% and 99.78% of the persistence intervals, respectively. The images resulting from applying these thresholds are shown above each diagram.

Filtering 90.51% of the persistence intervals produces an image that is visually indistinguishable from the original, whereas filtering 99.78% preserves only the most prominent features. This highlights the LPF's capability to extract essential image structures by eliminating topological noise.

For comparison, Figure 9.4 shows the effects of increasing levels of filtering using a wavelet filter applied to the same image and its persistence diagram. As discussed in Section 3, filters based on GSP and DSP (such as the wavelet filter) do not fully remove topological noise. The evolution of the image under increasing wavelet filtering differs immensely from the evolution under LPF thresholding, further emphasizing the distinctive behavior and effectiveness of the LPF.

Example 9.4 . A function defined on a 3D mesh can be interpreted as a signal on a graph whose domain consists of triangular faces. In this context, the mesh vertices form the nodes of the graph, and the function values—such as height, curvature, or other scalar quantities—constitute the signal values. Figure 9.5 demonstrates the application of the LPF to such a mesh-based signal.

The left column presents the original signal alongside its corresponding persistence diagram, capturing both 0-dimensional and 1-dimensional topological features. In the middle column, the LPF is applied to the

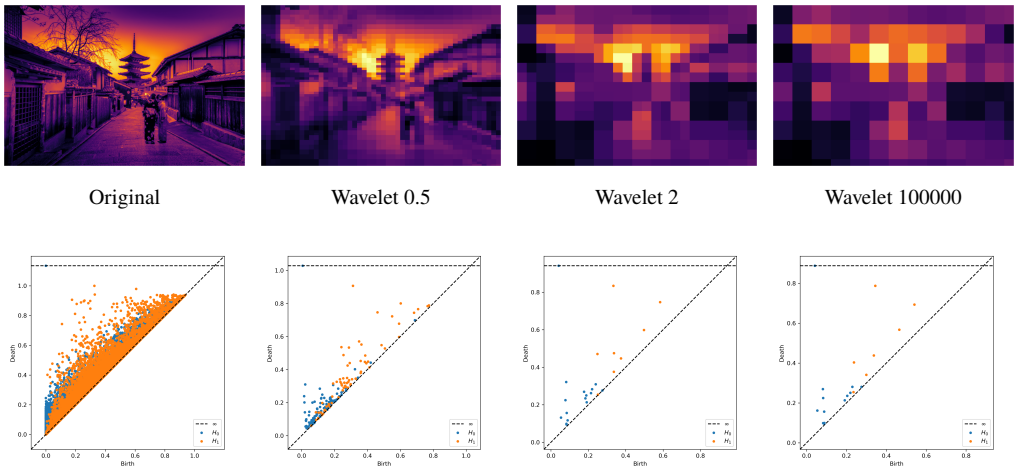


Figure 9.4: Top row: images after applying the wavelet filter with different parameter σ. Bottom row: corresponding persistence diagrams.

0-dimensional persistence, which primarily reflects local minima or "dents" in the signal. These features, often manifested as darker regions on the mesh, are effectively smoothed out by thresholding persistence intervals below $\varepsilon = 0.2$.

The right column shows the effect of subsequently applying the LPF to the 1-dimensional persistence, which corresponds to topological loops or "bumps" in the signal. These features, typically indicated by brighter elevated regions, are similarly filtered using the same threshold $\varepsilon = 0.2$.

Together, these two stages of filtering demonstrate the LPF's capacity to selectively remove topological noise across dimensions, preserving only the most significant geometric and topological structures of the original signal. This layered approach emphasizes the flexibility and interpretability of LPF-based processing in geometric and graph-based signal domains.

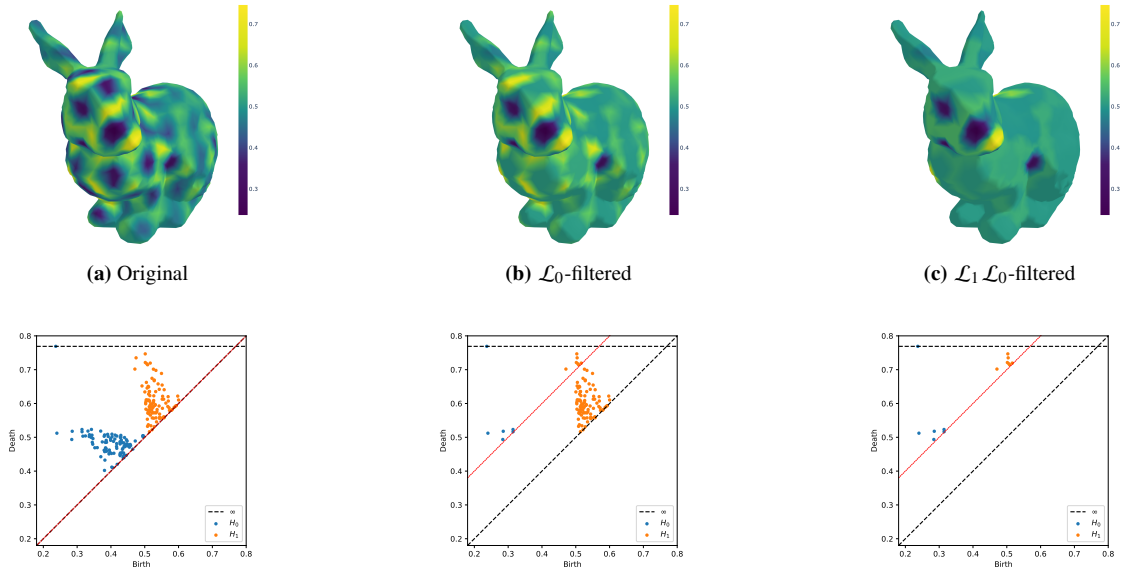
SECTION 10

Size Aware Low Persistence Filter

The construction of the low-persistence filter based on the Basin Hierarchy Tree (BHT) naturally extends to what we refer to as a *size-aware filter*. This enhanced version incorporates an additional structural constraint, enabling more refined control over the filtering process. Specifically, we introduce two independent thresholds: the persistence threshold $\varepsilon > 0$, as in the standard LPF, and a size threshold $\eta > 0$. Under this extended scheme, all homology classes whose persistence is less than ε *and* whose associated size is less than η are eliminated from the signal.

This dual-criterion filtering allows us to remove not only topological features with low persistence—which are typically considered topological noise—but also features that are spatially small, even if they have relatively high persistence. This is particularly valuable in applications where both geometric scale and topological significance play a role in determining the importance of a feature.

Intuitively, the notion of *size* depends on the homological dimension. For 0-dimensional homology, size refers to the number of vertices in a connected component, effectively measuring the extent of spatial spread



(a) Original **(b)** $\mathcal{L}_0$-filtered **(c)** $\mathcal{L}_1\mathcal{L}_0$-filtered

(d) Original PD **(e)** $\mathcal{L}_0$-filtered PD **(f)** $\mathcal{L}_1\mathcal{L}_0$-filtered PD

Figure 9.5: A signal on a 3D mesh (left column) with its corresponding persistence diagram (bottom row) and filtered versions after applying $\mathcal{L}_0^{\varepsilon}$ (middle column) and $\mathcal{L}_1^{\varepsilon}\mathcal{L}_0^{\varepsilon}$ (right column), with respective persistence diagrams on the second row ($\varepsilon = 0.2$).

in the graph domain. For 1-dimensional homology, size corresponds to the length or area of a representative 1-cycle, which may be defined in terms of the number vertices inside the finite region bounded by the cycle.

The inclusion of this size-sensitive criterion offers a powerful extension of the LPF, enabling the selective preservation of meaningful structures based not only on their persistence in the filtration, but also on their spatial scale. A formal definition of the size function, suitable for both 0- and 1-dimensional homology, is provided below.

Definition 10.1 . Given a signal $f : V \to \mathbb{R}$ over a graph with faces $\mathcal{G} = (V, E, F)$, with $G[\mathcal{E}]$ denoting an induced graph and $\mathfrak{T}$ the Basin Hierarchy Tree (BHT) of the induced signal $f_{G[\mathcal{E}]}$, we define the *size of the basin* of a vertex $v \in V_T$ as the cardinality of the set of its descendants in T,

$$\text{size}_{\mathfrak{T}}(v) = |\{u \in V_T \mid v \preceq_T u\}|.$$

If v corresponds to the birth of a homology class with interval $I = [f(v), f(\mathfrak{L}(v)))$ in the persistence diagram, we also define $\text{size}_{\mathfrak{T}}(I) = \text{size}_{\mathfrak{T}}(v)$.

Note that the subscript $\mathfrak{T}$ in $\text{size}_{\mathfrak{T}}$ is significant, as small differences between BHTs—due to variations in edge ordering—may lead to slight differences in basin sizes, particularly near basin boundaries where elements may be assigned to neighboring basins.

Definition 10.2 . Let $f : V \to \mathbb{R}$ be a signal over a graph with faces $\mathcal{G} = (V, E, F)$, with induced graph $G[\mathcal{E}]$ and $\mathfrak{T}$ a Basin Hierarchy Tree (BHT) of the induced signal $f_{G[\mathcal{E}]}$.

We define the **0-Size Aware Low Persistence Filter (0-SALPF)** as

$$ {}^{\mathfrak{T}}\mathcal{F}_0^{\varepsilon,\eta} f = \left({}^{\mathfrak{T}}\mathcal{F}_0^{\varepsilon,\eta} f_{G[\mathcal{E}]} \right) \big|_V, $$

where

$$ {}^{\mathfrak{T}}\mathcal{F}_0^{\varepsilon,\eta} f_{G[\mathcal{E}]}(v) = \max \left(\left\{ f_{G[\mathcal{E}]}(\mathfrak{L}(u)) \,\middle|\, u \in \tilde{A}_v^{\varepsilon} \right\} \cup \left\{ f_{G[\mathcal{E}]}(v) \right\} \right), \tag{10.1} $$

and

$$ \tilde{A}_v^{\varepsilon} = \left\{ u \in V \,\middle|\, \begin{array}{l} u \preceq_T v,\ \mathrm{pers}_{\mathfrak{T}}(u) < \varepsilon, \\ \mathrm{size}_{\mathfrak{T}}(u) < \eta \end{array} \right\} \tag{10.2} $$

is the set of ancestors of $v \in V$ in $\mathfrak{T}$ with persistence less than ε and size less than η.

When clear from context, we omit the dependence on $\mathfrak{T}$ and write $\mathcal{F}_0^{\varepsilon,\eta}$ in place of ${}^{\mathfrak{T}}\mathcal{F}_0^{\varepsilon,\eta}$.

In a completely analogous manner, we define the **1-Size Aware Low Persistence Filter (1-SALPF)** by replacing $f_{G[\mathcal{E}]}$ with $-f_{G[\mathcal{E}]}$ in the construction of the 0-SALPF.

Similar to the case of the Low Persistence Filter we define the general Size Aware Low Persistence Filter, which can filter both 0- and 1-homology simultaneously.

First we define the following filter

$$ \mathcal{F}_*^{\varepsilon,\eta,k} \triangleq \underbrace{\ldots \mathcal{F}_0^{\varepsilon,\eta} \mathcal{F}_1^{\varepsilon,\eta} \mathcal{F}_0^{\varepsilon,\eta}}_{k \text{ times}}. \tag{10.3} $$

Then, similar to the LPF case we have the following result.

Theorem 10.3. For any $\mathcal{G} = (\mathcal{G}, f)$, there exists $k \geq 0$ such that $\mathcal{F}_*^{\varepsilon,\eta,k} f = \mathcal{F}_*^{\varepsilon,\eta,k+1} f$. Denote by k_o the smallest such k. We have $\mathcal{F}_*^{\varepsilon,\eta,k_o} f = \mathcal{F}_*^{\varepsilon,\eta,k} f$ for any $k \geq k_o$. We define

$$ \mathcal{F}_*^{\varepsilon,\eta} f := \mathcal{F}_*^{\varepsilon,\eta,k_o} f. \tag{10.4} $$

Then, we have $|f - \mathcal{F}_*^{\varepsilon,\eta} f|_\infty < \varepsilon$, $\mathcal{L}_*^{\varepsilon} f \lesssim f$ and $\mathcal{F}_*^{\varepsilon,\eta} f$ has no persistence interval with both size less than η and persistence less than ε.

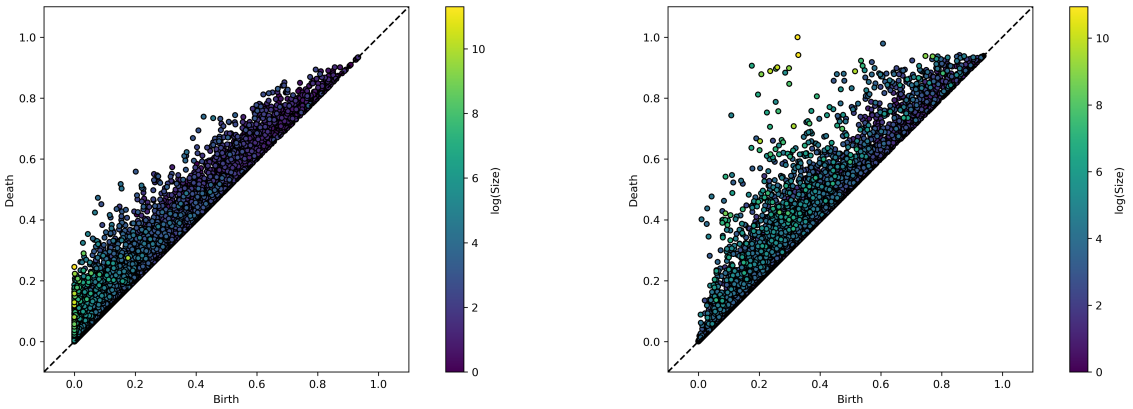
The proof of Theorem 10.3 is a analogous to Theorem 10.3.

We illustrate the behavior of the Size Aware Low Persistence Filter through the following examples.

Example 10.4 . In this first example, we examine the case of a natural image. Specifically, we revisit the pagoda image previously introduced in Figure 9.3. This image has dimensions 534×800, corresponding to over 400,000 pixels—each of which is treated as a vertex in our graph with faces. Consequently, the possible sizes of basins range from 1 to approximately 4×10^5, spanning several orders of magnitude.

Figure 10.1 presents the 0-dimensional persistence diagram (left) and the 1-dimensional persistence diagram (right), where each persistence interval is colored according to the size of the corresponding homological feature. Although there is a mild visual tendency for smaller features to appear closer to the diagonal—particularly in the 0-dimensional case—this pattern is far from systematic or consistent.

To further investigate this observation, Figure 10.2 quantifies the relationship between persistence and size. As the scatter plot reveals, there is very weak correlation between these two quantities across both homological dimensions. This empirical evidence highlights the importance of developing a filtering method that can account for both persistence and size independently. A filter based solely on persistence may overlook large but short-lived structures, while a purely size-based filter may ignore small but topologically significant features. The size-sensitive low-persistence filter addresses this limitation by allowing simultaneous control over both dimensions of feature relevance.

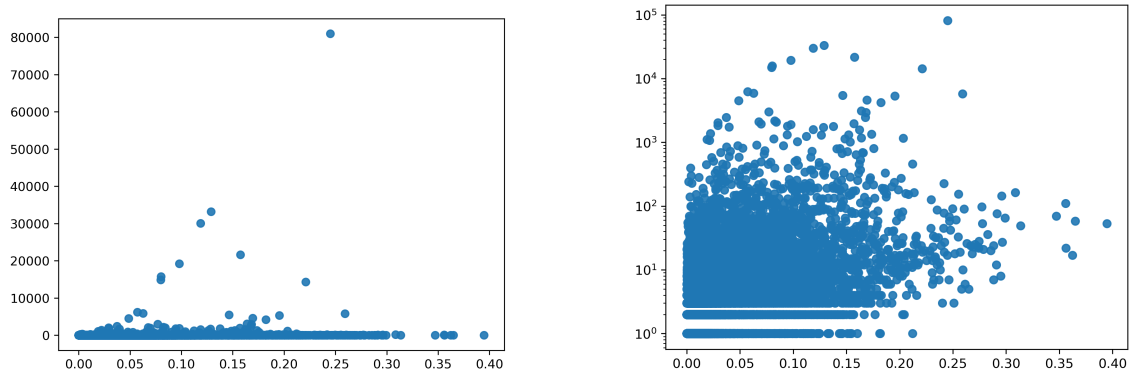


(a) PD_0 **(b)** Original PD

Figure 10.1: Persistence diagrams with feature size information for the natural image of the Pagoda.

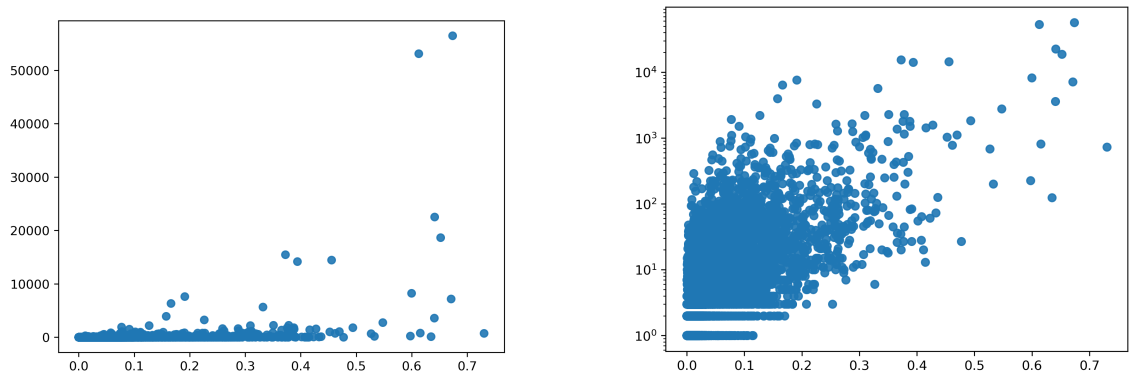
In Figure 10.3, we present the distribution of basin sizes corresponding to persistence intervals, plotted on a log-log scale. The plot reveals an apparent negative linear relationship between the frequency of occurrence and the basin size. Since this relation is linear in log-log scale, it indicates a power-law distribution, where the frequency of occurrence is inversely proportional to a power of the basin size.

Figure 10.6 illustrates the effect of the Size-Aware Low Persistence Filter on a synthetic image under different threshold configurations. The figure is organized as a grid, where each column corresponds to an increasing persistence threshold while the size threshold is held constant, and each row corresponds to an increasing size threshold with a fixed persistence threshold. This layout enables a clear visual comparison of how each parameter independently influences the filtering results. As the persistence threshold increases (left to right), smaller topological features are progressively removed, leading to a simplification of the image structure. Conversely, increasing the size threshold (top to bottom) suppresses components based on their spatial extent, further reducing noise and highlighting larger regions of interest. The figure demonstrates the complementary roles of persistence and size in controlling the granularity and selectivity of the filtering process.



(a) Linear scale (H_0)

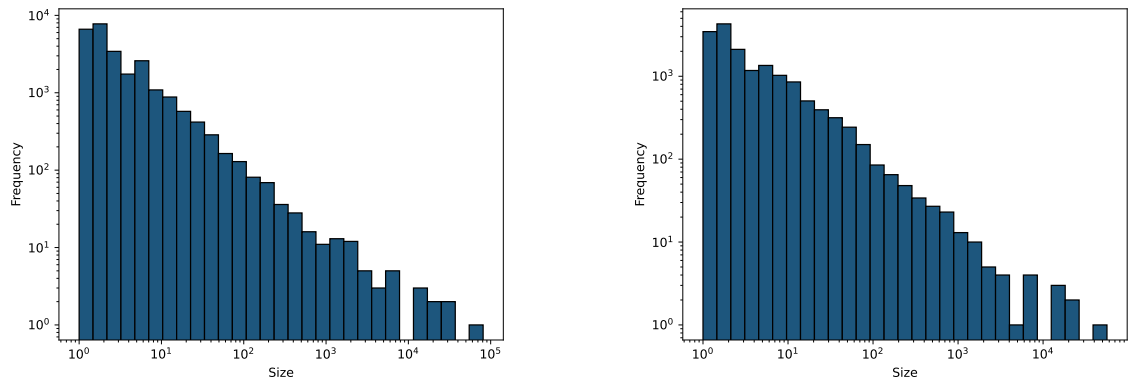
(b) Log scale (H_0)



(c) Linear scale (H_1)

(d) Log scale (H_1)

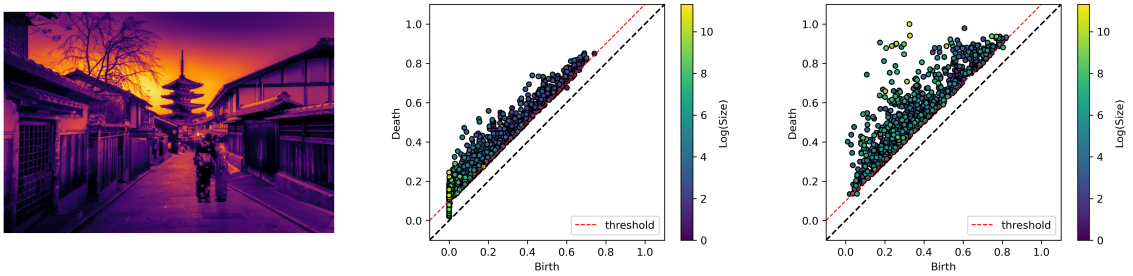
Figure 10.2: Persistence agains size in both linear scale (fig. 10.2a) and logarithmic scale (fig. 10.2b). Pearson correlation in H_0 is equal to 0.087, so there is a very weak linear relation, while for H_1 the measure is 0.269, showing a bit stronger linera correlation as we can see in the plot.



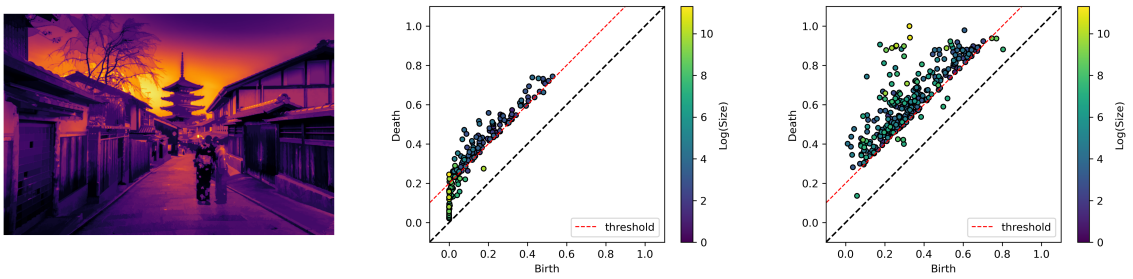
(a) H_0

(b) H_1

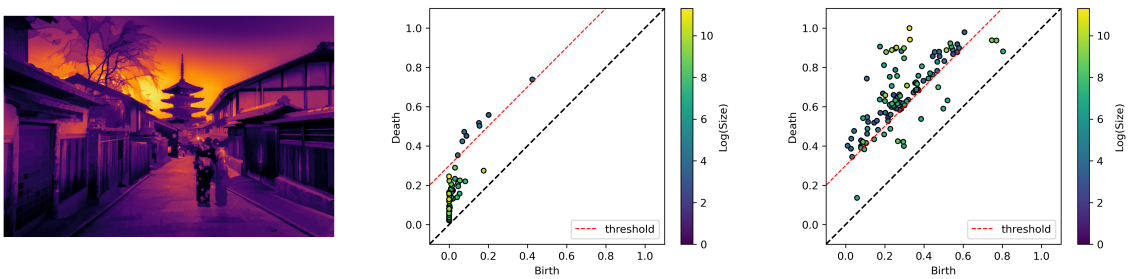
Figure 10.3: Histograms of sizes with both axes in logarithmic scale. **(a)** 0-homology; **(b)** 1-homology.



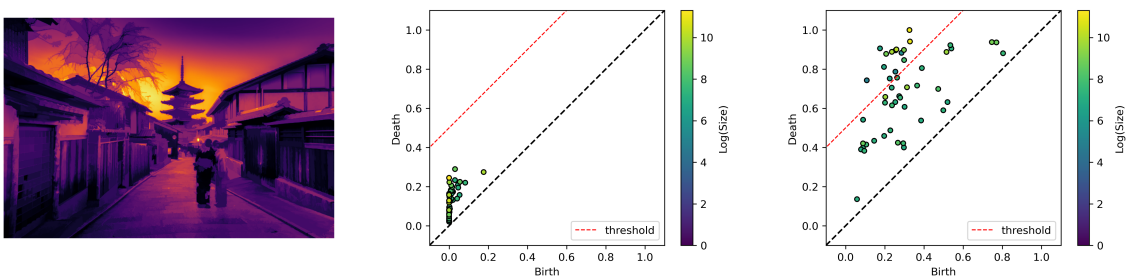
(a) $\eta = 1000,\ \varepsilon = 0.1$



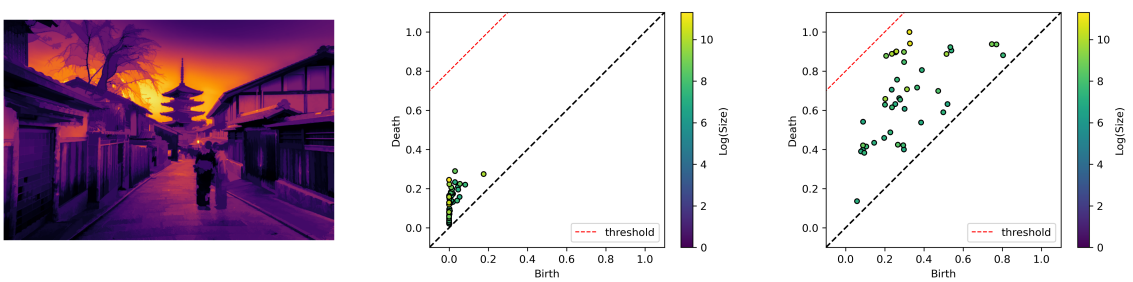
(b) $\eta = 1000,\ \varepsilon = 0.2$



(c) $\eta = 1000,\ \varepsilon = 0.3$

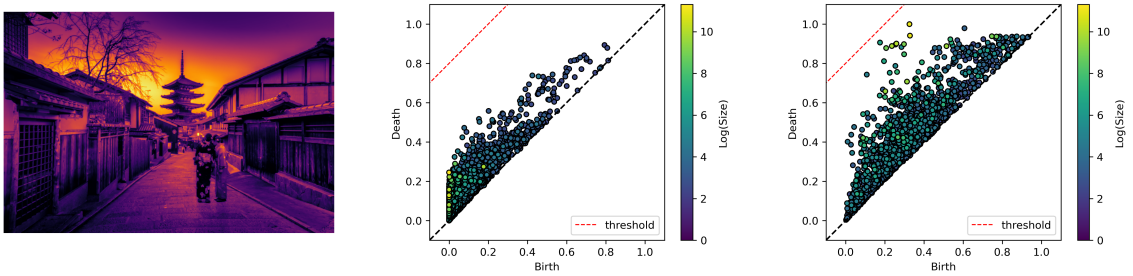


(d) $\eta = 1000,\ \varepsilon = 0.5$

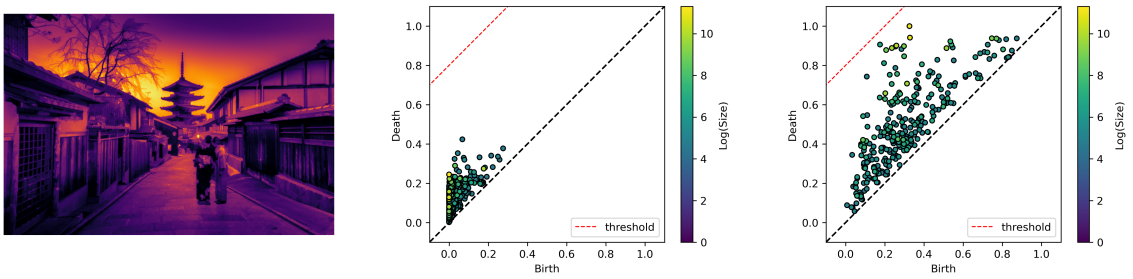


(e) $\eta = 1000,\ \varepsilon = 0.8$

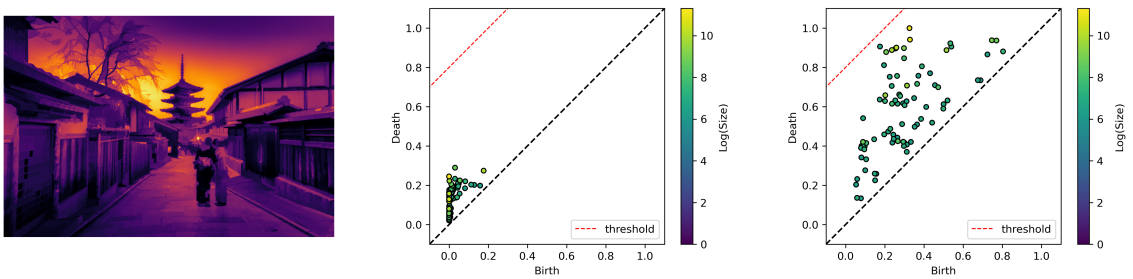
Figure 10.4: Size-Aware Low-Pass Filter applied to a natural image with a fixed size threshold and varying persistence thresholds.



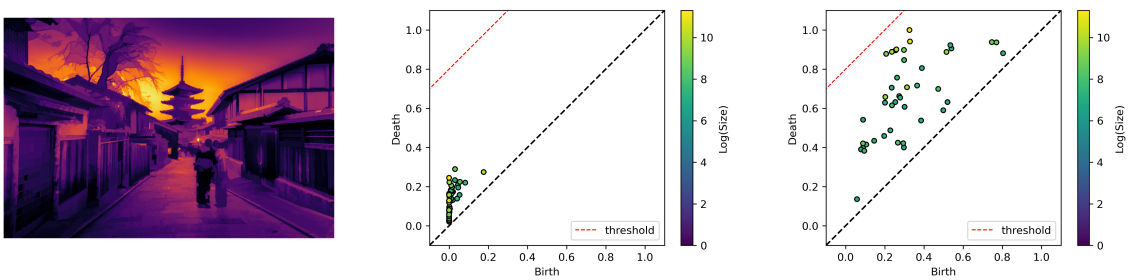
(a) $\eta = 10,\ \varepsilon = \infty$



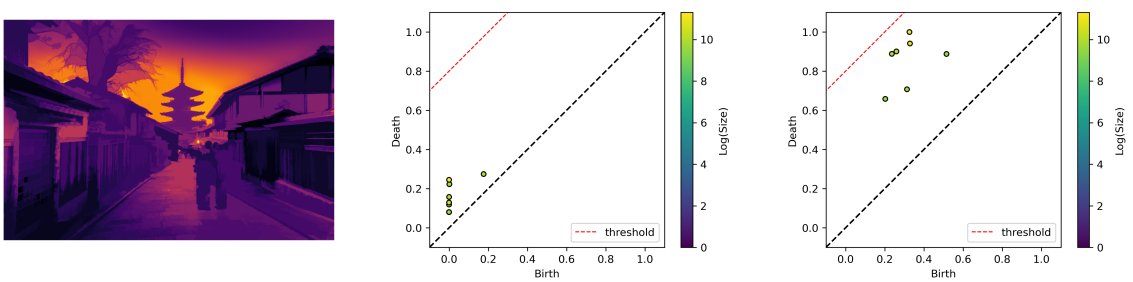
(b) $\eta = 100,\ \varepsilon = \infty$



(c) $\eta = 500,\ \varepsilon = \infty$



(d) $\eta = 1000,\ \varepsilon = \infty$



(e) $\eta = 10000,\ \varepsilon = \infty$

Figure 10.5: Size-Aware Low-Pass Filter applied to a natural image with a fixed maximum persistence threshold and varying size thresholds.

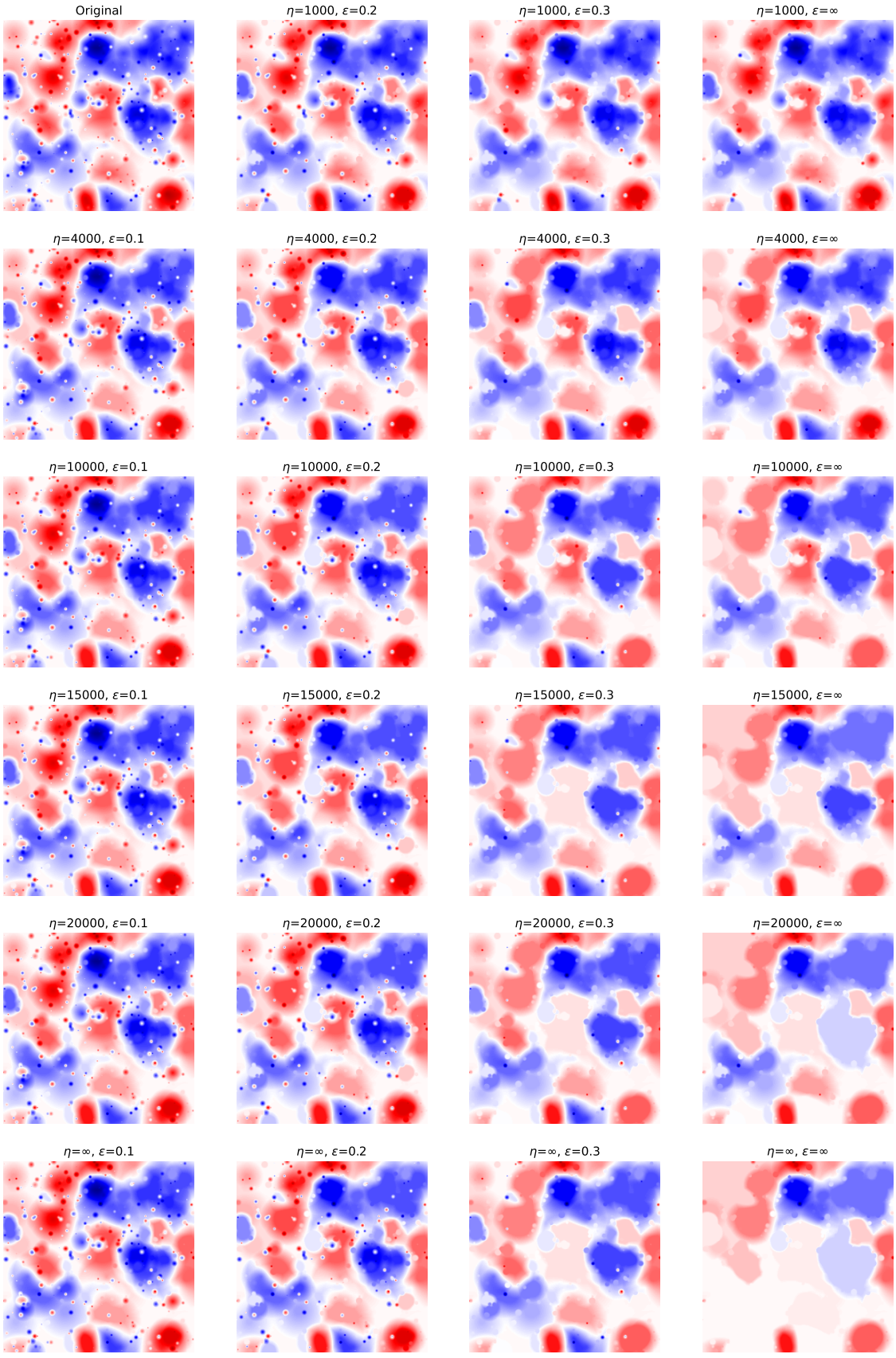


Figure 10.6: Results of the Size-Aware Low Persistence Filter applied to a synthetic image under varying threshold values. From left to right, the persistence threshold increases while the size threshold remains fixed. From top to bottom, the size threshold increases while the persistence threshold remains fixed. Blue regions indicate lower values, with increased intensity corresponding to lower magnitudes. Conversely, red regions denote higher values, with greater intensity representing higher magnitudes. White regions represent intermediate values between these extremes.

SECTION 11

Basin Hierarchy Tree Summary Statistics

In this section, we introduce summary statistics that can be extracted from the Basin Hierarchy Tree (BHT). These statistics can be used as feature vectors for machine learning algorithms, as demonstrated in Section 11.2.

The summaries introduced here are based on two structural measures in rooted trees: the **depth** of a node—defined as the number of edges in the path from the root to the node—and the **height** of a node—defined as the number of edges in the longest path from the node to a leaf.

11.1 BHT Depth and Height Distributions

We begin by defining depth and height distributions in the general setting of rooted trees.

Definition 11.1 (Depth Distribution)**.** Let $T = (V, E)$ be a rooted tree with root vertex $r \in V$. The *depth* of a vertex $v \in V$ is the number of edges in the unique path from r to v. The *depth distribution* of T is the sequence $(d_0, d_1, \ldots, d_H)$, where d_k denotes the number of vertices at depth k, and H is the height of the tree (i.e., the maximum depth). This distribution provides a level-wise profile of the tree structure when traversed from the root.

Definition 11.2 (Height Distribution)**.** Let $T = (V, E)$ be a rooted tree. The *height* of a vertex $v \in V$ is the number of edges in the longest path from v to a leaf in the subtree rooted at v. The *height distribution* of T is the sequence $(h_0, h_1, \ldots, h_H)$, where h_k denotes the number of vertices with height k, and H is again the height of the tree. This distribution summarizes how subtree sizes are distributed across the hierarchy.

We now extend these concepts to the setting of Basin Hierarchy Trees, beginning with a key subtree.

Definition 11.3 (Positive Persistence Subtree)**.** Let $\mathfrak{T} = (T, \mathfrak{L})$ be a Basin Hierarchy Tree. We know that each node in $T = (V, E)$ has persistence less than or equal to that of its parent (see Lemma 6.5). Define

$$V^+ = \{v \in V \mid \mathrm{pers}(v) > 0\}, \quad E^+ = \{uv \in E \mid u, v \in V^+\}.$$

Then $T^+ = (V^+, E^+)$ is a rooted subtree of T with the same root. We refer to T^+ as the *Positive Persistence Subtree* of the BHT.

Definition 11.4 (BHT Depth Distribution)**.** The *BHT Depth Distribution* of a BHT $\mathfrak{T} = (T, \mathfrak{L})$ is defined as the depth distribution of its Positive Persistence Subtree T^+.

Definition 11.5 (BHT Height Distribution)**.** The *BHT Height Distribution* of a BHT $\mathfrak{T} = (T, \mathfrak{L})$ is defined as the height distribution of its Positive Persistence Subtree T^+.

These two distributions summarize the hierarchical structure of the non-trivial (i.e., positive persistence)

nodes in a BHT—information that is entirely lost in the traditional persistence diagram. In the following section, we provide an example to illustrate how the BHT is a richer structure in terms of the topological and geometric information it captures from the data.

Before presenting this example, we define a class of alternative, weighted versions of the BHT depth and height distributions that incorporate additional persistent homology information.

Definition 11.6 (Weighted BHT Depth and Height Distributions)**.** In the standard definitions above, each node at depth or height k contributes a unit count:

$$d_k = \sum_{\text{depth}(v)=k} 1, \qquad h_k = \sum_{\text{height}(v)=k} 1.$$

Alternatively, one can define a weight w_v for each vertex v, leading to the weighted versions:

$$d_k = \sum_{\text{depth}(v)=k} w_v, \qquad h_k = \sum_{\text{height}(v)=k} w_v.$$

This allows the distributions to reflect the "importance" of nodes based on their attributes.

Natural choices for weights arise from the BHT itself. Since each node in the Positive Persistence Subtree T^+ has both positive persistence and a basin size, several possible weight functions include:

$$\begin{aligned}
&w_v = \text{pers}(v), && w_v = \text{size}(v), && w_v = \text{pers}(v) \cdot \text{size}(v),\\
&w_v = (\text{pers}(v))^{\text{depth}(v)}, && w_v = (\text{size}(v))^{\text{depth}(v)}, && w_v = \alpha \cdot \text{pers}(v) + \beta \cdot \text{size}(v),\\
&w_v = (\text{pers}(v))^{\text{height}(v)}, && w_v = (\text{size}(v))^{\text{height}(v)}, && w_v = (\text{pers}(v))^{\text{size}(v)}.
\end{aligned}$$

In short, many informative weight schemes can be designed using combinations of persistence, basin size, depth, and height of each node.

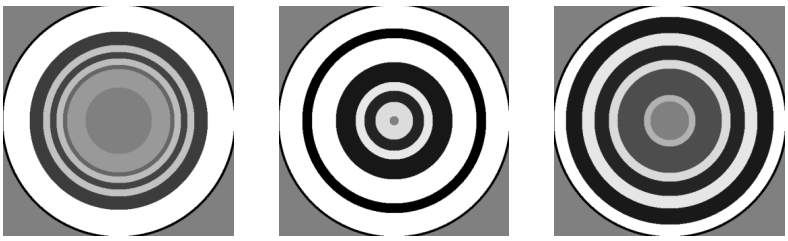
Figure 11.1: Visual representation of a pattern with a deeply nested, hierarchical structure. The patterns are artificially generated by successively merging annular regions arranged in a radial configuration resembling a squared sine curve multiplied by a monotonically increasing function. Each pattern is characterized by the number of local minima it contains; in the figure, from left to right, this number is 10, 30, and 100, respectively.

11.2 Binary classification with BHT features

The complete set of experiments related to this section can be accessed in the public GitHub repository maintained by the author: `github.com/mvlier/Persistence_Based_Methods`.

In this section, we present a simple synthetic example to demonstrate the effectiveness of BHT summary statistics as feature vectors for training a Random Forest Classifier in a binary classification problem. We start

(a) Pattern A ($n = 3$)



(b) Pattern B ($n = 4$)

Figure 11.2: Two patterns with distinct number of rings.

by describing a particular pattern that results in rich structures in the BHT, which are lost in the persistence diagram.

In Figure 11.2 we show the two types of patterns that we will use for generating a dataset of images. The three images on the top row all fall in the class of $n = 3$, n being the number of local minima (more explicitely the number of rings that are local minimum). The three images on the bottom row fall in the class of $n = 4$. We call this two pattern A and pattern B, respectively.

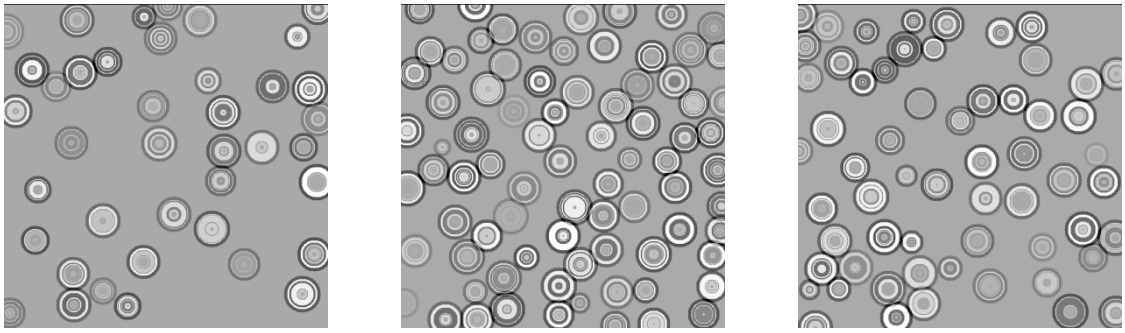
Experiment Setup. For our binary classification task, we generate datasets denoted as EN-M, where N and M range from 0 to 100 (e.g., E70-40). To clarify the significance of N and M in the dataset name, we describe the construction of each image in both classes. Each image in Class 1 of a dataset EN-M consists of a collection of non-overlapping patterns of types A and B, as defined previously. Specifically, N% of the patterns are of type A, and the remaining $(100-N)$% are of type B. Likewise, each image in Class 2 contains M% type A patterns and $(100-M)$% type B patterns. The smaller the difference between N and M, the more similar the two classes become, thus increasing the difficulty of the classification task. Figure 11.3 shows examples from the E70-40 dataset: images in the top row belong to Class 1, while those in the bottom row correspond to Class 2.

Note that the density and size of the patterns vary across images, as only the proportion between the two types is relevant.

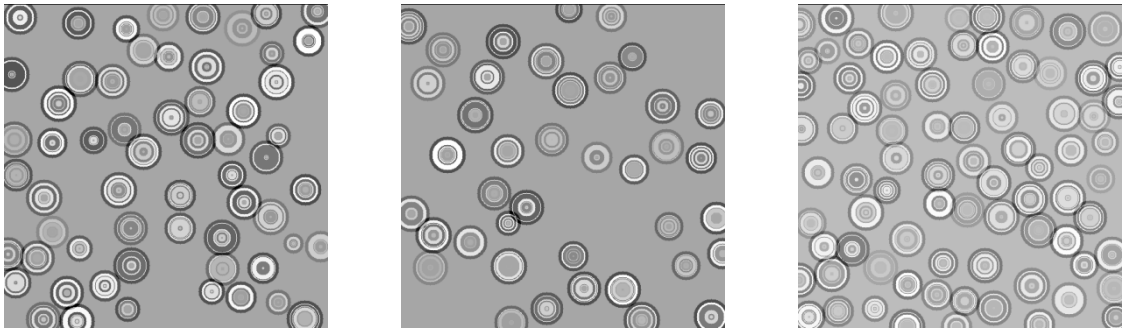
In all experiments, we use 512×512 pixel images, with the number of patterns in each image ranging from 40 to 80. The outer radius of each pattern is capped at 25 pixels.

Using these rules, we generate the datasets E60-50, E60-40, E70-40, E80-50, E70-30, E80-20, E90-10, and E100-0. The earlier datasets (e.g., E60-50 and E60-40) are particularly challenging to classify due to the high similarity between the two classes, whereas the later datasets (e.g., E90-10 and E100-0) are significantly easier to classify, owing to a greater distinction between the classes, as we discussed before.

Each dataset consists of 300 images, with 150 samples per class. For all models, we employ a stratified



(a) Class 1 **(b)** Class 1 **(c)** Class 1



(d) Class 2 **(e)** Class 2 **(f)** Class 2

Figure 11.3: Examples of the two classes in E70-40. On the top line we see three examples of pattern A and on the bottom line we see three examples of pattern B.

60%/40% train-test split, resulting in 180 training images and 120 testing images.

Except for a deep learning model described in the next section, all models used in this study are random forest classifiers. These models differ solely in the feature vectors utilized. We categorize these features into two groups: BHT-based features (derived from BHT summary statistics) and non-BHT features (computed directly from the persistence diagrams). The objective is to assess whether incorporating the additional hierarchical information captured by BHT—which is not retained in persistence diagrams—leads to improved performance. Given that the generated datasets contain numerous patterns with hierarchical structures, it is reasonable to hypothesize that BHT-based features may offer a performance advantage.

BHT based Features

Depth Distribution. The depth distribution may vary in length depending on the input. To standardize it, we truncate the distribution to the first six values (corresponding to depths 0 through 5). The sixth entry (index 5) aggregates all remaining values, i.e., it holds the sum of the original distribution from index 5 to the end. This process is applied separately to the 0- and 1-dimensional homology depth distributions. The resulting vectors are then concatenated into a single feature vector of size 12.

Height Distribution. This feature is constructed in the same manner as the Depth Distribution. We truncate each height distribution at index 5 and aggregate the remaining values into the sixth entry. Concatenating the 0- and 1-dimensional homology vectors yields a 12-dimensional feature vector.

Depth-Height Distribution. This feature vector is the concatenation of the Depth Distribution and the Height Distribution described above, resulting in a 24-dimensional vector.

PE + Method. Each of the three previously described methods is also combined with Persistent Entropy (PE)—explained below—by concatenating the corresponding feature vector with the 2-dimensional PE vector.

Non-BHT Based Features

Persistence Entropy. The Persistence Entropy [RCMP2016, CGGD+2015] in our setting is represented as a two-dimensional vector. The first component corresponds to the entropy of the 0-dimensional persistence diagram, and the second to that of the 1-dimensional diagram. Formally,

$$\texttt{PE}(\mathrm{PD}) = (\texttt{entropy}(\mathrm{PD}_0), \texttt{entropy}(\mathrm{PD}_1)),$$

where

$$\texttt{entropy}(\mathrm{PD}_i) = -\sum_{I \in \mathrm{PD}_i} p_I \log(p_I), \quad \text{with} \quad p_I = \frac{\mathrm{pers}(I)}{\sum_{J \in \mathrm{PD}_i} \mathrm{pers}(J)}.$$

Here, we consider each PD_i as the restriction of the persistence diagram to only finite persistence intervals. We slightly abuse notation by writing sums over $I \in \mathrm{PD}_i$, implicitly accounting for multiplicities. That is, I and J are counted according to their multiplicities in the multiset PD_i, and the two sums above should formally be expressed as

$$-\sum_{I \in \mathbb{R}^2} \mathrm{PD}_i(I).p_I.\log(p_I) \quad \text{and} \quad \sum_{J \in \mathbb{R}^2} \mathrm{PD}_i(J).\mathrm{pers}(J),$$

in which $\mathrm{PD}_i(I)$ is the multiplicity of I in PD_i.

Persistence Image. The *persistence image* [AEK+2017] is a vectorized representation of a persistence diagram, enabling its use in machine learning pipelines. Given a persistence diagram $\mathrm{PD} = \{(b_i, d_i)\}_{i=1}^N$, each point is mapped to birth-persistence coordinates (b_i, p_i), where $p_i = d_i - b_i$. A weighted sum of Gaussians centered at these coordinates defines a continuous function $\rho : \mathbb{R}^2 \to \mathbb{R}$:

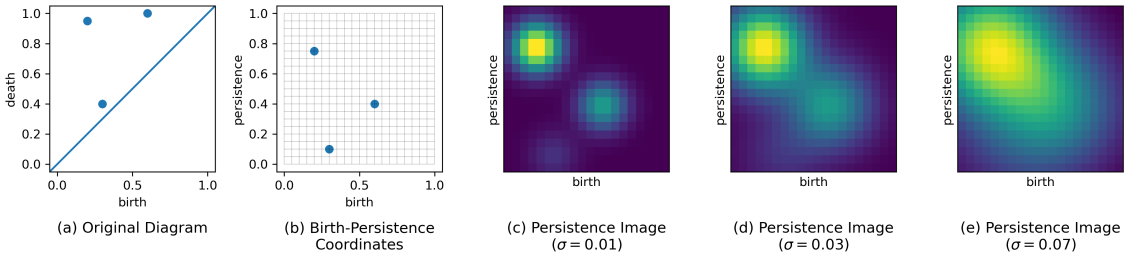
$$\rho(x, y) = \sum_{i=1}^{N} w(b_i, p_i) \cdot \exp\left(-\frac{(x - b_i)^2 + (y - p_i)^2}{2\sigma^2}\right),$$

where $w : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ is a weight function, typically chosen to vanish near the diagonal and increase with persistence. This emphasizes robustness to noise, as features near the diagonal are more likely to appear randomly due to small fluctuations in the data, so if the weighting function was not introduced the function ρ would be very unstable with respect to PD.

To obtain the persistence image, the domain of ρ is discretized into a grid of rectangular cells $\{C_{ij}\}_{i,j}$ in the birth-persistence plane. The value assigned to each pixel (i, j) is given by:

$$I_{ij} = \iint_{C_{ij}} \rho(x, y)\, dx\, dy.$$

The resulting matrix (I_{ij}) forms a fixed-dimensional numerical representation suitable for statistical and machine learning applications.



(a) Original Diagram (b) Birth-Persistence Coordinates (c) Persistence Image ($\sigma = 0.01$) (d) Persistence Image ($\sigma = 0.03$) (e) Persistence Image ($\sigma = 0.07$)

Figure 11.4: Illustration of the persistent image of a simple persistence diagram (a) with just 3 intervals. In (b) we see the diagram in birth-persistence coordinates. In (c), (d), and (e) we see the persistence image with increasing values for the standard deviation $\sigma = 0.01,\ 0.03$, and 0.07, respectively.

There are three main choices involved in generating persistence images: the distribution, the weighting function, and the image resolution. In our experiments, we use the `Persim`[1] package from `Scikit-TDA`. We retain the default settings: a Gaussian distribution and a linear weighting function proportional to the persistence, given by $w(b, p) = p$.

We experiment with four different image resolutions—2×2, 4×4, 10×10, and 20×20—as well as three different values of the Gaussian parameter σ (0.01, 0.03, and 0.07). For each experiment, we report results using the resolution and σ that yield the highest classification accuracy. In all cases, we concatenate the 0-dimensional and 1-dimensional persistence images into a single flattened vector of size $2d^2$, where $d \in \{2, 4, 10, 20\}$ denotes the resolution dimension.

In Figure 11.4, we illustrate the computation of the persistence image for a simple persistence diagram containing just three intervals. Panel (a) shows the original persistence diagram, which is first transformed into birth–persistence coordinates as shown in (b). Panels (c), (d), and (e) display the resulting persistence images computed with $\sigma = 0.01$, $\sigma = 0.03$, and $\sigma = 0.07$, respectively. As σ increases, the Gaussian distribution at each point becomes more spread out, and the image becomes more uniform. This highlights σ as a critical hyperparameter, which we tune across these three values in our experiments.

We also observe the effect of the linear persistence-based weighting $w(b, p) = p$, where features with higher persistence (toward the top of the diagram) contribute more prominently to the image intensity.

Deep Learning Model (ResNet)

Apart from all the Random Forest Classifiers with various feature vectors, as described above, we also train a deep learning model for this binary classification task. The model used is the state-of-the-art ResNet model [HZRS2016], more details are described next.

Model Architecture. We adopt a convolutional neural network from the `timm` library, using a pretrained `resnet18` and `resnet50` architecture as our base model. The final classification layer is modified to match the number of output classes (two classes) in the dataset. The model is fine-tuned end-to-end on our task.

Data Preparation and Augmentation. During training, we apply several data augmentation techniques to increase generalization:

- Resize to 230×230 pixels

[1] https://persim.scikit-tda.org/en/latest/index.html

- Random crop to 224×224
- Random horizontal and vertical flips
- Random rotation within $\pm 15°$
- Normalization with mean 0.63 and standard deviation 0.15

Validation images are resized to 230×230 and then center-cropped to 224×224, followed by the same normalization.

Training Configuration. The model is trained for 80 epochs using the NAdam optimizer with a learning rate of 1e−4. The batch size is set to 32. A step learning rate scheduler is applied, reducing the learning rate by a factor of 0.1 at 50 epochs. The loss function used is the binary cross-entropy.

Experiments Results

In Table 11.1, we observe that the highest classification accuracy across all datasets is achieved using a Random Forest classifier trained on BHT-based features. As the distinction between the two classes becomes more pronounced—going to the columns on the right—the accuracy of the BHT-based classifier approaches perfection. It exceeds 85% in all but two datasets (E60-40 and E60-50), and reaches perfect accuracy in the E100-0 case.

Even in this latter scenario, where the class distinction is trivial when using BHT summaries, the non-BHT methods—Persistence Entropy and Persistence Image—underperform, achieving only 85.83% and 90.83% accuracy, respectively.

Notably, when the two classes are highly similar—as in the E60-50 and E60-40 datasets—the non-BHT methods perform at near-random levels (close to 50% accuracy), while the BHT-based methods maintain significantly better performance, achieving 68.33% on E60-50 and 75.83% on E60-40. This is particularly remarkable given the minimal differences between classes in these datasets, and highlights the capacity of BHT-based summaries to detect subtle variations in texture patterns.

In all cases, the BHT-based methods outperform the non-BHT alternatives by a margin exceeding 11% in accuracy. These results highlight the fact that BHTs encode structural information that is not captured by persistence diagrams alone.

Table 11.1: Accuracy (%) of different features for classification across multiple datasets.

Method	E60-50	E60-40	E70-40	E80-50	E70-30	E80-20	E90-10	E100-0
BHT-based								
Depth	65.00	66.67	79.17	87.50	90.00	97.50	98.33	**100.00**
Height	52.50	69.17	75.00	81.67	90.00	95.83	98.33	99.17
Depth + Height	57.50	69.17	79.17	**90.00**	**94.17**	**97.50**	98.33	**100.00**
PE + Depth	57.50	67.50	82.50	89.17	91.67	**97.50**	98.33	**100.00**
PE + Height	50.00	64.17	80.83	79.17	90.00	98.33	**99.17**	**100.00**
PE + Depth + Height	**58.33**	**72.50**	**86.67**	**90.00**	92.50	96.67	98.33	**100.00**
PD-Based								
Pers. Entropy	45.83	51.67	64.17	54.17	65.00	73.33	80.00	85.83
Pers. Image	**58.33**	63.33	73.33	67.50	81.67	82.50	90.83	90.83
Deep Learning								
ResNet50	50.00	53.33	55.00	70.83	81.67	90.00	96.67	98.33
ResNet18	53.33	48.33	60.83	59.17	75.00	87.50	91.67	99.17

Chapter V

Conclusion

Topological Data Analysis (TDA), and in particular persistent homology, has emerged as a powerful framework for extracting multi-scale topological features from complex data. This thesis builds on these foundations to address a central challenge in the field: how to design filtering methods that remove topological noise while preserving the essential topological structure of signals defined on non-Euclidean domains such as graphs and higher-dimensional complexes.

We began by demonstrating that conventional denoising techniques—such as spectral graph filtering and deep learning-based methods—while effective at reducing topological noise to some extent, are fundamentally limited in their ability to preserve high-persistence, topologically meaningful features. These methods often introduce unacceptable distortions to the original signal or diminish important topological structures, especially when applied to complex, structured domains.

To overcome these limitations, we formalized the problem of persistence-aware signal filtering under strict structural constraints, and showed that, in general, this problem is not solvable when attempting to simultaneously filter features across multiple homological dimensions. In response, we proposed a relaxed version of the problem, allowing small, controlled variations in high-persistence features to enable meaningful simplification while preserving global topological structure.

At the core of our solution is the *Basin Hierarchy Tree* (BHT), a novel data structure that hierarchically encodes persistence information and supports efficient, structured filtering. Using this structure, we introduced the *Low Persistence Filter*, an algorithm that selectively removes low-persistence features while preserving those above a specified threshold. We extended this filter from graphs to two-dimensional generalized cell complexes (referred to as graphs with faces), enabling simultaneous filtering in 0- and 1-dimensional homology. We provided theoretical guarantees on topological fidelity within a defined approximation bound, culminating in our main result (Theorem 7.10).

To demonstrate the practical utility of our method, we developed an open-source implementation and applied it to a series of illustrative examples. We further extended the framework to incorporate a second threshold

based on feature size, leading to the *Size-Aware Low Persistence Filter*. Additionally, we introduced a suite of summary statistics derived from the BHT, which we showed to be effective descriptors in a synthetic classification task. These statistics not only outperformed traditional persistent homology features but also surpassed the accuracy of state-of-the-art deep learning models, highlighting the representational power of persistence-guided methods.

Future Directions. Several promising research directions emerge from this work. First, although the proposed filtering methods are developed for functions defined on graphs and two-dimensional complexes, extending these techniques to higher-dimensional topological domains remains an open and challenging problem. A related direction involves adapting the framework to signals defined on higher-order cells—rather than on vertices—following the recent advances in cell complexes signal processing proposed by Barbarossa et al. [SBT2021].

Second, further integration of Basin Hierarchy Tree-derived features into machine learning pipelines presents an exciting avenue for interpretable AI. In particular, combining persistence-based and size-sensitive descriptors may enhance model performance in settings characterized by rich topological structure.

Finally, exploring hybrid filtering strategies that combine spectral techniques with persistence-aware filtering could lead to more robust and adaptive methods, making use of the strengths of both approaches to address noise and feature preservation in complementary ways.

In summary, this thesis contributes both theoretical insights and practical methodologies for topologically informed signal processing. By addressing fundamental limitations of conventional techniques and introducing a new class of persistence-aware filtering algorithms, we advance toward a more effective understanding of topological structure in real-world data.

Bibliography

[AEK+2017] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.

[AGH+2009] Dominique Attali, Marc Glisse, Samuel Hornus, Francis Lazarus, and Dmitriy Morozov. Persistence-sensitive simplication of functions on surfaces in linear time. In *TopoInVis' 09*, 2009.

[Ale1915] J. W. Alexander. A proof of the invariance of certain constants of analysis situs. *Transactions of the American Mathematical Society*, 16(2):148–154, 1915.

[ASSC2002] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[Bau2021] Ulrich Bauer. Ripser: efficient computation of vietoris–rips persistence barcodes. *Journal of Applied and Computational Topology*, 5:391–423, 2021.

[BS2020] Sergio Barbarossa and Stefania Sardellitti. Topological signal processing over simplicial complexes. *IEEE Transactions on Signal Processing*, 68:2992–3007, 2020.

[Can1883] Georg Cantor. Grundlagen einer allgemeinen mannigfaltigkeitslehre. ein mathematisch-philosophischer versuch in der lehre des unendlichen. *Teubner, Leipzig*, 1883. Originally presented in parts between 1879 and 1883.

[Car2009] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[CB2015] William Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and its Applications*, 14(05):1550066, 2015.

[CCSG+2009] Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Y. Oudot. Proximity of persistence modules and their diagrams. *Proceedings of the 25th Annual Symposium on Computational Geometry*, pages 237–246, 2009.

[CdSGO2016] Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Y. Oudot. *The Structure and Stability of Persistence Modules*. SpringerBriefs in Mathematics. Springer, 2016.

[CF2006] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys (CSUR)*, 38(1):2, 2006.

[CGGD+2015] Harish Chintakunta, Thanos Gentimis, Rocio Gonzalez-Diaz, Maria-Jose Jimenez, and Hamid Krim. An entropy-based persistence barcode. *Pattern Recognition*, 48(2):391–401, 2015.

[CGGG+2024] Mauricio Che, Fernando Galaz-García, Luis Guijarro, Ingrid Membrillo Solis, and Motiejus Valiunas. Basic metric geometry of the bottleneck distance. *Proceedings of the American Mathematical Society*, 152(08):3575–3591, 2024.

[CSEH2007] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[CSMK2014] Siheng Chen, Aliaksei Sandryhaila, José MF Moura, and Jelena Kovacevic. Signal denoising on graphs via graph filtering. In *2014 ieee global conference on signal and information processing (globalsip)*, pages 872–876. IEEE, 2014.

[EH2010a] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010.

[EH2010b] Herbert Edelsbrunner and John L. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.

[ELZ2002] Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & computational geometry*, 28:511–533, 2002.

[EMP2006] Herbert Edelsbrunner, Dmitriy Morozov, and Valerio Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 127–134, 2006.

[ES1952] Samuel Eilenberg and Norman Steenrod. *Foundations of Algebraic Topology*. Princeton University Press, Princeton, 1952.

[Eul1736] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736.

[Far2003] Michael Farber. Topological complexity of motion planning. *Discrete & Computational Geometry*, 29(2):211–221, 2003.

[Far2004] Michael Farber. Instabilities of robot motion. *Topology and its Applications*, 140(2-3):245–266, 2004.

[For2010] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[Fro1992] Patrizio Frosini. Measuring shapes by size functions. In David P. Casasent, editor, *Intelligent Robots and Computer Vision X: Algorithms and Techniques*, volume 1607, pages 122 – 133. International Society for Optics and Photonics, SPIE, 1992.

[GF1964] Bernard A Galler and Michael J Fisher. An improved equivalence algorithm. *Communications of the ACM*, 7(5):301–303, 1964.

[GP2010] Victor Guillemin and Alan Pollack. *Differential Topology*. American Mathematical Society, 2010. Classic text introducing differential topology concepts.

[GPCI2015] Chad Giusti, Eva Pastalkova, Carina Curto, and Vladimir Itskov. Clique topology reveals intrinsic geometric structure in neural correlations. *Proceedings of the National Academy of Sciences*, 112(44):13455–13460, 2015.

[GW2018] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson, 4 edition, 2018.

[Hat2002] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[Hau1914] Felix Hausdorff. *Grundzüge der Mengenlehre*. Veit & Comp., 1914. English translation: *Set Theory*, Chelsea Publishing Company, 1957.

[HJ2013] Danijela Horak and Jürgen Jost. Spectra of combinatorial laplace operators on simplicial complexes. *Advances in Mathematics*, 244:303–336, 2013.

[HNH+2016] Yasuaki Hiraoka, Takenobu Nakamura, Akihiko Hirata, Emerson G Escolar, Kaname Matsue, and Yasumasa Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016.

[Hop1931] Heinz Hopf. Über die abbildungen der dreidimensionalen sphäre auf die kugelfläche. *Mathematische Annalen*, 104(1):637–665, 1931.

[HZRS2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[Kle1926] Felix Klein. *Vorlesungen über die Entwicklung der Mathematik im 19. Jahrhundert*, volume 1–2 of *Die Grundlehren der mathematischen Wissenschaften*. Verlag von Julius Springer, Berlin, 1926.

[KSA2020] Shizuo Kaji, Takeki Sudo, and Kazushi Ahara. Cubical ripser: Software for computing persistent homology of image and volume data, 2020.

[LCS+2021] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 1833–1844, October 2021.

[Lim2020] Lek-Heng Lim. Hodge laplacians on graphs. *Siam Review*, 62(3):685–715, 2020.

[MBGY2014] Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, pages 167–174, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[MBW2013] Dmitriy Morozov, Kenes Beketayev, and Gunther Weber. Interleaving distance between merge trees, 2013.

[McC2001] John McCleary. *A user's guide to spectral sequences*. Number 58. Cambridge University Press, 2nd edition, 2001. Standard modern introduction with applications across topology.

[Mun2000] James R. Munkres. *Topology*. Prentice Hall, 2000.

[Nak2018] Mikio Nakahara. *Geometry, topology and physics*. CRC press, 2018. Covers fiber bundles, gauge theory, and topological concepts in modern physics.

[New2003] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[Noe1927] Emmy Noether. Abstrakter aufbau der idealtheorie in algebraischen zahl- und funktionenkörpern. *Mathematische Annalen*, 96(1):26–61, 1927. Noether's work on modules and ideals had profound influence on homological methods in topology.

[NS1988] Charles Nash and Siddhartha Sen. *Topology and geometry for physicists*. Elsevier, 1988. Standard reference connecting algebraic topology with gauge theories and general relativity.

[OFK+2018] Antonio Ortega, Pascal Frossard, Jelena Kovacevic, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[Oud2015] Steve Y. Oudot. *Persistence Theory: From Quiver Representations to Data Analysis*. American Mathematical Society, 2015.

[PL2022] Konstantinos Parashakis and Andreas Loukas. Simplicial complexes for graph signal processing. *IEEE Transactions on Signal Processing*, 70:3077–3092, 2022.

[Poi1895] Henri Poincaré. *Analysis Situs*. Journal de l'École Polytechnique, 1895.

[RCMP2016] Matteo Rucco, Filippo Castiglione, Emanuela Merelli, and Marco Pettini. Characterisation of the idiotypic immune network through persistent entropy. In Stefano Battiston, Francesco De Pellegrini, Guido Caldarelli, and Emanuela Merelli, editors, *Proceedings of ECCS 2014*, pages 117–128, Cham, 2016. Springer International Publishing.

[Rob1999] Vanessa Robins. Towards computing homology from finite approximations. In *Topology proceedings*, volume 24, pages 503–532, 1999.

[SB2024] Stefania Sardellitti and Sergio Barbarossa. Topological signal processing over generalized cell complexes. *IEEE Transactions on Signal Processing*, 72:687–700, 2024.

[SBT2021] Stefania Sardellitti, Sergio Barbarossa, and Lucia Testa. Topological signal processing over cell complexes. In *2021 55th Asilomar Conference on Signals, Systems, and Computers*, pages 1558–1562, 2021.

[SM2013] Aliaksei Sandryhaila and José M. F. Moura. Discrete signal processing on graphs. *IEEE Transactions on Signal Processing*, 61(7):1644–1656, 2013.

[SM2020] Dmitriy Smirnov and Dmitriy Morozov. Triplet merge trees. In Hamish Carr, Issei Fujishiro, Filip Sadlo, and Shigeo Takahashi, editors, *Topological Methods in Data Analysis and Visualization V*, pages 19–36, Cham, 2020. Springer International Publishing.

[SNF+2013] David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[SSF+2022] Michael T. Schaub, Jean-Baptiste Seby, Florian Frantzen, T. Mitchell Roddenberry, Yu Zhu, and Santiago Segarra. *Signal Processing on Simplicial Complexes*, pages 301–328. Springer International Publishing, Cham, 2022.

[Tau2000] G TaubinÝ. Geometric signal processing on polygonal meshes. In *Proceedings of EUROGRAPHICS*, 2000.

[Vie1927] Leopold Vietoris. Über den höheren zusammenhang kompakter räume und eine klasse von zusammenhangstreuen abbildungen. *Mathematische Annalen*, 97:454–472, 1927. One of the foundational papers leading to the concept of simplicial homology.

[vLZK2024] Matias de Jong van Lier, Sebastián Elías Graiff Zurita, and Shizuo Kaji. Topological filtering of a signal over a network. *arXiv preprint arXiv:2408.14109*, 2024.

[VUFF1993] Alessandro Verri, Claudio Uras, Patrizio Frosini, and Massimo Ferri. On the use of size functions for shape analysis. *Biological cybernetics*, 70(2):99–107, 1993.

[Wen2017] Xiao-Gang Wen. Colloquium: Zoo of quantum-topological phases of matter. *Rev. Mod. Phys.*, 89:041004, Dec 2017. Comprehensive review on the role of topology in classifying quantum phases.

[Whi2012] George W Whitehead. *Elements of homotopy theory*, volume 61. Springer Science & Business Media, 2012. Comprehensive treatment of classical and modern homotopy theory.

[XW2014] Kelin Xia and Guo-Wei Wei. Persistent homology analysis of protein structure, flexibility, and folding. *International Journal for Numerical Methods in Biomedical Engineering*, 30(8):814–844, 2014.

[ZLS+2023] Yi Zhang, Dasong Li, Xiaoyu Shi, Dailan He, Kangning Song, Xiaogang Wang, Honwei Qin, and Hongsheng Li. Kbnet: Kernel basis network for image restoration. *arXiv preprint arXiv:2303.02881*, 2023.

[ZLZ+2020] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. *arXiv preprint*, 2020.

[ZZC+2017] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.