PERSISTENT HOMOLOGY VIA QUOTIENT SPACES

TAM THANH TRUONG

A Master's Thesis in Topology

Department of Mathematics
Faculty of Mathematics and Natural Sciences
University of Bergen

20th November, 2017

Tam Thanh Truong: *Persistent homology via quotient spaces,* A Master's Thesis in Topology, © 20th November, 2017

SUPERVISORS:
Morten Brun
Nello Blaser

LOCATION:
Norway, Bergen

TIME FRAME:
20th November, 2017

ABSTRACT

A point cloud can be endowed with a topological structure by constructing a simplicial complex using the points as vertices. Instead of assigning a single simplicial complex, Topological Data Analysis (TDA) employs multiple simplicial complexes, each representing the point cloud at a different resolution. These combine to form a filtration: a nested sequence of simplicial complexes which gives rise to persistent homology, a useful tool able to extract topological information from the point cloud. The Vietoris-Rips filtration is a popular choice in TDA, mainly for its simplicity and easy implementation for high-dimensional point clouds. Unfortunately, this filtration is often too large to construct fully.

We introduce in this thesis a way of reducing a simplicial complex by identifying its vertices. Applying this technique to each simplicial complex in the Vietoris-Rips filtration results in a smaller filtration that can be shown to approximate the Vietoris-Rips filtration in terms of persistent homology.

Ohana means family.
Family means nobody gets left behind, or forgotten.

— Lilo & Stitch

ACKNOWLEDGMENTS

This thesis is the result of my journey in obtaining a Master's degree, whereby I have been accompanied and encouraged by many people. I would like to thank and dedicate this section to all those people.

My deepest and sincere gratitude goes to both my supervisors, Morten Brun and Nello Blaser, whose ideas and suggestions helped to make this thesis a reality. The last two years have been made much more enjoyable thanks to their encouragement and unending patience.

I would like to express a profound sense of gratitude to Stefano Piceghello, Erlend Raa Vågset and Fredrik Heimstad for spending their spare time to read and correct the countless mistakes in this thesis; an extra thanks to the first for refining the figures used in the thesis.

To my friends and fellow students at the University of Bergen, my wholehearted thanks for all the moments and laughs we have shared.

I am greatly indebeted to my parents for their blessings, love and affection throughout the years. To my two younger brothers, I thank them for their encouragement and unyielding moral support.

Last, but not least, I would like to thank Heidi, for finding me the light, whenever it was far away.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

INTRODUCTION

With the improvement in computing technology, we are able to produce various kinds of data at an incredible rate. The relatively new term "Big Data" reflects this very well; the data produced is much larger and more complex than in the past. This complexity often stems from the data being noisier, making it difficult to sort out the irrelevant parts, or the data being very high-dimensional, which limits any kind of visualization.

Topological Data Analysis (TDA) is a recent field which has proved successful in extracting information from metric datasets, i. e. point clouds. One of the main techniques in TDA, persistent homology, was introduced in a paper by Edelsbrunner et al. [12] (2002), which was followed by the works of Zomorodian and Carlsson [23] (2005). An increasing number of applications have appeared recently in fields such as social sciences [4, 18], biology [5], neurosciences [14, 21] and nanotechnologies [17].

The goal of TDA is to determine qualitative properties or structures of the point cloud by studying its shape. There are multiple ways of approaching this underlying shape, e. g. the *Mapper* algorithm introduced by Singh et al. [19], notable for its discovery of a new subtype of breast cancer cite:nicolau:2011. In this thesis, however, we choose to adopt the technique of persistent homology.

Persistent homology revolves around building a nested sequence of simplicial complexes, called a filtration, with the point cloud as the vertex set. Since any simplicial complex can be realized as a topological space, the filtration captures the desired notion of shape at various resolutions. The specific choice of filtration is important as it should be able to detect relevant properties of the point cloud. We can visualize the persistent homology of a filtration via the so-called persistence diagram.

A popular choice for the filtration is the Vietoris-Rips filtration, based on balls increasing in size around each point in the cloud. While easy to implement, the size of the filtration grows exponentially with the amount of points, which makes it computationally expensive.

In this thesis, we will approximate the persistent homology of the Vietoris-Rips filtration via a different filtration. Since the limiting factor of the Vietoris-Rips filtration is the size of the point cloud, the immediate aim would be to reduce this size by some means. By identifying points before constructing the Vietoris-Rips filtration, we obtain a smaller filtration, called the quotient filtration. In terms of topological spaces, this quotient filtration consists of quotient spaces of the original spaces in the Vietoris-Rips filtration. We will show that given certain restrictions there exists an interleaving between the two filtrations, which results in their associated persistence diagrams being similar. This, in other words, means that the quotient filtration approximates the Vietoris-Rips filtration in terms of persistent homology.

While this quotient filtration construction is, most likely, not original, we have not seen this used in the context of persistent homology. The fact that we get one of the two inclusions for free makes it rather simple to achieve an interleaving with the Vietoris-Rips filtration.

OVERVIEW

The thesis is structured as follows:

Chapter 1 introduces basic concepts within simplicial methods and algebraic topology that will be frequently used throughout the thesis.

Chapter 2 describes the fundamentals of persistent homology, which we will use to compare various datasets in Chapter 4. We also define the Čech and Vietoris-Rips filtrations, the latter of which we are approximating in this thesis.

Chapter 3 introduces two new simplicial complexes: the preimage complex and the quotient complex. We also state and prove in this chapter the main theoretical result of this thesis, namely that the complexes are homotopy equivalent whenever the map used to construct them is surjective. We also present a section on the Nerve Theorem, which is needed for the proof.

Chapter 4 revisits the quotient filtration and describes a restriction needed to make it interleaved with the Vietoris-Rips filtration. We also describe the algorithm we have implemented to make such a filtration. Finally we compare the persistence diagrams between the filtrations using a few synthetic datasets.

In Chapter 5 we discuss the results of the applications done in the previous chapter and introduce several ideas for future work.

Part I

SOME KIND OF MANUAL

1 PRELIMINARIES

The topics presented in this chapter forms a basis for the thesis. These preliminaries only include concepts and results from simplicial methods and algebraic topology and we therefore expect the reader to be familiar with various algebraic objects such as groups, quotient sets and vector spaces.

The chapter is included to make the thesis more self-contained and any experienced reader may skip directly to Chapter 2 and use this chapter as a reference. Most of the following expositions and definitions are largely based on various literature. See [10], [15] and [13].

1.1 SIMPLICIAL METHODS

We present here a minimalistic approach to simplicial methods, the main notions being the simplicial complex and its geometric realization.

Definition 1.1.1 (Abstract simplicial complex). A collection K of finite nonempty subsets of a set V is called a *(abstract) simplicial complex* if, for any σ in K and any nonempty set $\tau \subseteq \sigma$, τ also is a member of K.

We will refer to V as the *vertex set* of K and the members of K as *simplices*. More specifically, if σ is in K and the cardinality $\mathrm{card}(\sigma)$ of σ is equal to n, then we say its *dimension* is equal to $n-1$ or that it's a $(n-1)$-simplex.

Sometimes we will also denote the simplicial complex as (K, V) if we need to emphasize the vertex set.

Example 1.1.2. For the set $V = \{0, 1, 2, 3, 4\}$, we can define the simplicial complex $K = \{0, 1, 2, 3, 4, 01, 02, 12, 23, 24, 34, 012, 234\}$ (here e. g. 234 denotes the set $\{2, 3, 4\}$). By identifying the set V with the vertices in Figure 1 we can describe K as the depicted diagram; the singleton sets, the two-element sets and the three-element sets in K correspond to the points, the lines and the shaded triangles in Figure 1, respectively.

Proposition 1.1.3. *The intersection of a collection of simplicial complexes $\{K_i\}_{i \in I}$ is either empty or a simplicial complex.*

Proof. Assume $K := \cap_{i \in I} K_i$ is nonempty and let σ be a simplex of K. We want to show that if τ is contained in σ, then τ is also a simplex of K. This should however be obvious: if $\tau \subseteq \sigma$ and σ is a simplex of K_i for all $i \in I$, then it follows from definition that τ must also be

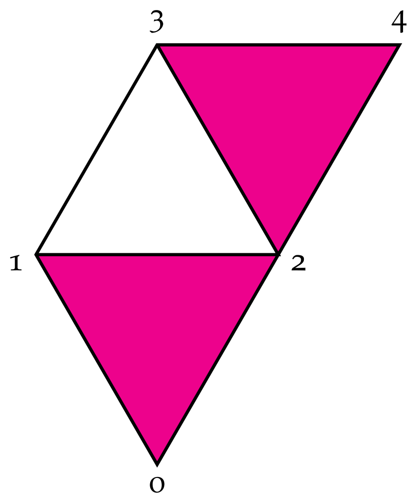

Figure 1: The simplicial complex K of Example 1.1.2

a simplex of K_i for all $i \in I$. Thus, τ is also in the intersection of all these, K. $\square$

Definition 1.1.4 (Closure). Let V be a set. We define the *closure* $\overline{V}$ of V to be the smallest simplicial complex containing V, i. e. the collection of all nonempty subsets of V.

Remark 1.1.5. A map $f\colon V \to W$ between the vertex sets of two simplicial complexes (K, V) and (L, W) also induces a map between the simplicial complexes; if σ is a simplex of K, then the image of σ under f is defined as $f(\sigma) := \{f(v) \mid v \in \sigma\}$.

Definition 1.1.6 (Simplicial map). Let (K, V) and (L, W) be simplicial complexes. A map $f\colon V \to W$ is called a *simplicial map* if it maps simplices to simplices, i. e. for σ in K we have that $f(\sigma)$ belongs to L.

Definition 1.1.7. The *standard n-simplex* is a subset of $\mathbb{R}^{n+1}$ given by

$$\Delta^n = \left\{ (t_0, t_1, \dots, t_n) \in \mathbb{R}^{n+1} \mid \sum_{i=0}^{n} t_i = 1, t_i \geqslant 0 \right\}.$$

We now describe a way to endow an abstract simplicial complex with a topology by embedding it into a euclidean space. The following definition only considers finite simplicial complexes, which is sufficient for the purpose of this thesis.

Definition 1.1.8 (Geometric realization). Let (K, V) be a finite simplicial complex, i. e. $|V| = n$ for some $n \in \mathbb{N}$. We define the *geometric realization*, or just *realization*, $|K|$ of K as a subspace of $\mathbb{R}^{n+1}$ by choosing an embedding f of V onto the vertices of the standard n-simplex Δ^n and identifying the simplices of K with the corresponding face/simplex spanned by the image of the vertices by f.

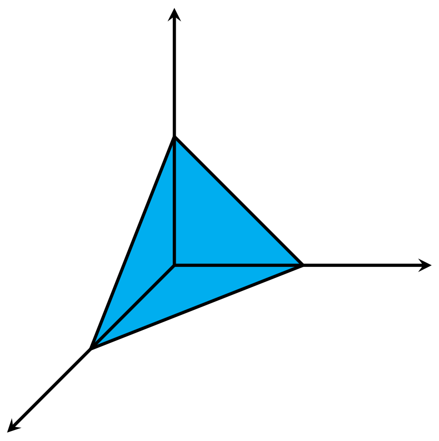

Figure 2: The standard 2-simplex

Remark 1.1.9. Throughout this thesis we will omit the notation $|-|$ when no ambiguity can arise. Whenever we refer to a simplicial complex K by its topological features, we mean the topological features of its realization $|K|$.

1.2 ALGEBRAIC TOPOLOGY

1.2.1 *Homotopy*

The important notions to take from this section will be the ones of homotopy equivalence and connectedness, both needed for the main theoretical result in Chapter 3.

Definition 1.2.1 (Homotopy). Let X and Y be two topological spaces and f and g be two functions from X to Y. A *homotopy* between f and g is a map $H\colon X \times [0,1] \to Y$ such that

- H is continuous,
- $H(x, 0) = f(x)$ and
- $H(x, 1) = g(x)$.

We say that two maps $f, g\colon X \to Y$ are *homotopic* if there exists a homotopy H between them; we denote this as $f \simeq g$.

Definition 1.2.2 (Homotopy equivalence). Let X and Y be two topological spaces. A map $f\colon X \to Y$ is said to be a *homotopy equivalence* if there exists a map $g\colon Y \to X$ such that $f \circ g \simeq \mathbb{1}_Y$ and $g \circ f \simeq \mathbb{1}_X$ (here $\mathbb{1}_X$ denotes the identity map on X).

We say that two spaces X and Y are *homotopy equivalent*, or have the same *homotopy type*, if there exists a homotopy equivalence f between them; we denote this as $X \simeq Y$. If a space X is homotopy equivalent to a point, then we say that X is *contractible*.

Definition 1.2.3 (Homotopy group). Let X be a topological space with $x_0 \in X$. Then the n-*th homotopy group* $\pi_n(X, x_0)$ of X is the set of homotopy classes $[f]$ of maps $f\colon (I^n, \partial I^n) \to (X, x_0)$ such that the homotopies satisfies $H((\partial I^n), t) = x_0$ for all t.

The group operation is defined as follows:

$$(f+g)\left((s_1, s_2, \dots, s_n)\right) = \begin{cases} f(2s_1, s_2, \dots, s_n), & s_1 \in \left[0, \frac{1}{2}\right] \\ g(2s_1 - 1, s_, \dots, s_n) & s_1 \in \left[\frac{1}{2}, 1\right] \end{cases}$$

Remark 1.2.4. We define below the concept of a topological space X being "path-connected". It can be shown that for such a space all the homotopy groups $\pi_n(X, x)$ coincide, i. e. they are isomorphic for all $x \in X$. Thus, we often shorten the notation to $\pi_n(X)$.

Definition 1.2.5 (Connectedness). Let X be a topological space. We say that X is *path-connected* if for any two points x, y in X there exists a path between them, i. e. a continuous function $\alpha\colon [0,1] \to X$ such that $\alpha(0) = x$ and $\alpha(1) = y$. More generally, X is said to be n-*connected* if X is path-connected and its first n homotopy groups vanish, i. e. $\pi_i(X) \cong 0$ for $1 \leqslant i \leqslant n$.

1.2.2 *Homology*

The main topics covered here are chain complexes and homology. More specifically, we will define the specific chain complex which gives rise to simplicial homology and we will assign this construction to filtrations in Chapter 2 to define persistent homology.

Definition 1.2.6 (Chain complex). We define a *chain complex* $\mathcal{C}$ of abelian groups to be a sequence of abelian groups $(C_k)_{k \in \mathbb{Z}}$ connected by maps $\partial_k\colon C_k \to C_{k-1}$ which satisfy $\partial_{k-1}\partial_k = 0$ for all $k \in \mathbb{Z}$. We refer to the ∂_k's as *boundary maps* and often denote a chain complex as follows:

$$\cdots \xrightarrow{\partial_{k+1}} C_k \xrightarrow{\partial_k} C_{k-1} \xrightarrow{\partial_{k-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} C_{-1} \xrightarrow{\partial_{-1}} \cdots$$

For a chain complex $\mathcal{C}$ and an integer k, we define the following subspaces:

$$Z_k(\mathcal{C}) := \ker(\partial_k), \text{ the } k\text{-}\textit{cycles},$$
$$B_k(\mathcal{C}) := \operatorname{im}(\partial_{k+1}), \text{ the } k\text{-}\textit{boundaries}.$$

Definition 1.2.7 (Homology group). The p-*th homology group*, $H_p(\mathcal{C})$ of $\mathcal{C}$ is defined as the quotient:

$$H_p(\mathcal{C}) := \frac{Z_p(\mathcal{C})}{B_p(\mathcal{C})}$$

Definition 1.2.8 (Orientation). An *orientation* of an n-simplex is an ordering of its vertices, i. e. $(v_0, v_1, \dots, v_{n-1})$, such that two orientations are considered equal if they only differ by an even permutation. A simplex with an orientation is said to be *oriented*.

Because there are only two orientations to consider, we denote the opposite orientation of a simplex as the negative of the simplex.

Example 1.2.9. $(v_0, v_1, v_2) = (v_2, v_0, v_1) = -(v_0, v_2, v_1) = -(v_2, v_1, v_0)$.

We will assume from now on that all our simplices are oriented.

Definition 1.2.10 (Group of (oriented) n-chains). Let (K, V) be a finite simplicial complex. The *group of (oriented) n-chains* $C_n(K)$ of K is the free abelian group generated by the n-simplices of K. An n-chain is therefore a finite sum $\sum_{i=1}^{N} a_i \sigma_i$ where the a_i are the coefficients and σ_i are the n-simplices.

Definition 1.2.11 (Boundary map). Let K be a simplicial complex and $(v_0, v_1, \dots, v_n)$ an $(n+1)$-simplex of K. We define the boundary map $\partial_{n+1} \colon C_{n+1}(K) \to C_n(K)$ as:

$$\partial_{n+1}\left((v_0, v_1, \dots, v_n)\right) = \sum_{i=0}^{n} (-1)^i (v_0, v_1, \dots, \widehat{v}_i, \dots, v_n).$$

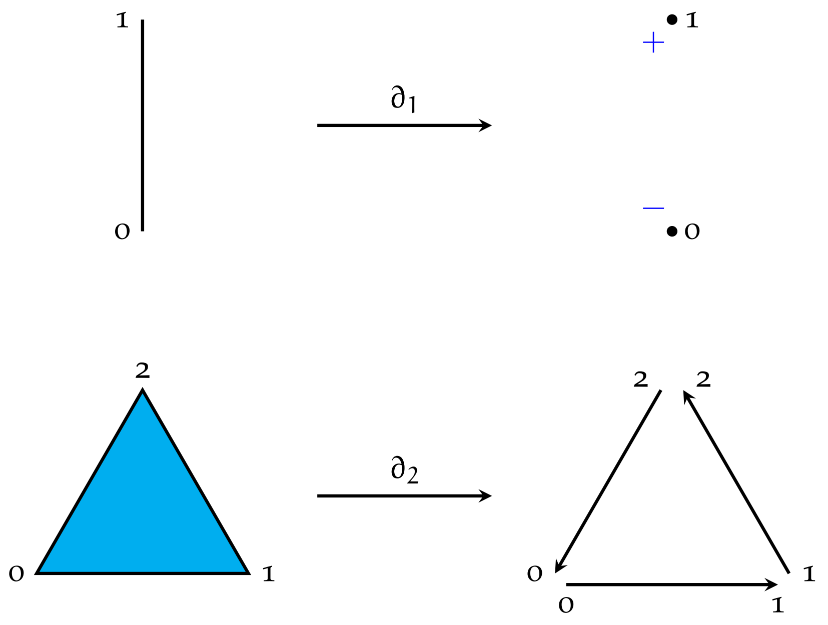

Figure 3: Boundary maps ∂_1 and ∂_2

Proposition 1.2.12. *For a natural number n, we have that $\partial_n \circ \partial_{n+1} = 0$.*

Proof. We show this by direct computation:

$$\begin{aligned}
&\partial_n \circ \partial_{n+1}((v_0, \dots, v_n)) \\
&\quad = \partial_n \left(\sum_{i=0}^{n+1} (-1)^i (v_0, \dots, \widehat{v}_i, \dots, v_n) \right) \\
&\quad = \sum_{i=0}^{n+1} (-1)^i \partial_n((v_0, \dots, \widehat{v}_i, \dots, v_n)) \\
&\quad = \sum_{i=0}^{n+1} (-1)^i \left(\sum_{j=0}^{i-1} (-1)^j ((v_+, \dots, \widehat{v}_j, \dots, \widehat{v}_i, \dots, v_n)) \right. \\
&\qquad\qquad \left. + \sum_{j=i}^{n} (-1)^j ((v_0, \dots, \widehat{v}_i, \dots, \widehat{v}_{j+1}, \dots, v_n)) \right) \\
&\quad = \sum_{i=0}^{n+1} \sum_{j=0}^{i-1} (-1)^{i+j} ((v_+, \dots, \widehat{v}_j, \dots, \widehat{v}_i, \dots, v_n)) \\
&\qquad\qquad + \sum_{i=0}^{n+1} \sum_{j=i}^{n} (-1)^{i+j} ((v_0, \dots, \widehat{v}_i, \dots, \widehat{v}_{j+1}, \dots, v_n)) \\
&\quad = 0
\end{aligned}$$

because the coefficients for a given simplex $(v_0, \dots, \widehat{v}_k, \dots, \widehat{v}_l, \dots, v_n)$ above are equal to $(-1)^{k+l} + (-1)^{k+l-1} = (-1)^{k+l-1}(-1+1) = 0$. $\square$

This shows that for a simplicial complex K, we have a chain complex $\mathcal{C}(K) = (C_n(K))_{n \in \mathcal{N}}$ associated to it; taking the homology of this chain complex results in *simplicial homology*.

Definition 1.2.13 (Simplicial homology). For a simplicial complex K, we define the p*-th simplicial homology group* $H_p(K)$ of K as

$$H_p(K) = H_p(\mathcal{C}(K))$$

where $\mathcal{C}(K)$ is the chain complex above.

Remark 1.2.14. Simplicial homology can be considered as functors $H_p(-)$ from the category of simplicial complexes to the category of abelian groups. In fact, if the coefficients are taken from a field k, then the simplicial homology $H_p(K)$ of a simplicial complex K becomes a k-vector space. Unless specified otherwise, we will assume the coefficients to be in the field $\mathbb{Z}/2\mathbb{Z}$ which makes the corresponding homology functors map to the category of vector spaces instead. This distinction for functors will become important for the next chapter. For more about categories and functors, see [16].

2 VIETORIS-RIPS AND PERSISTENT HOMOLOGY

Our data usually comes in form of finite subsets $P \subseteq \mathbb{R}^k$, and therefore talking about the topology is quite uninteresting as all the higher homology groups vanish. One can however remedy this by constructing a simplicial complex using P as a vertex set. Not only does this endow our dataset P with a topology, but, depending on the chosen construction, we also get the notion of *persistence*, i. e. topological features which persists over long periods of time. This will be made more clear in Section 2.2

We will start by defining and comparing the Čech and Vietoris-Rips complexes, two frequently used tools in TDA. Both of these give rise to filtrations which are relatively easy to implement.

2.1 ČECH, VIETORIS-RIPS AND FILTRATIONS

Assuming we have a finite dataset P, we proceed by creating balls of constant radius $t > 0$ around each point in P. The construction of the Čech (and Vietoris-Rips) complex considers the intersection of such balls and creates simplices accordingly.

Definition 2.1.1 (Čech complex). Let P be a finite subset of $\mathbb{R}^k$ and $B_{t/2}(x)$ be the open ball with radius $\frac{t}{2}$ around a point x in P. The *Čech complex* $C_t(P)$ is the simplicial complex with vertex set P given by all the subsets σ of P such that the intersection of balls around each point in σ is nonempty. That is,

$$C_t(P) = \{\sigma \subseteq P \mid \cap_{x \in \sigma} B_{t/2}(x) \neq \emptyset\}.$$

One might note that this construction is not limited to balls only. In fact, the more general concept is defined as follows:

Definition 2.1.2 (Nerve). Let $\mathcal{F} = \{U_i\}_{i \in I}$ be a family of sets. The *nerve* $\mathcal{N}(\mathcal{F})$ of $\mathcal{F}$ is the simplicial complex defined on the vertex set I such that a finite subset $\sigma \subseteq I$ is a simplex of $\mathcal{N}(\mathcal{F})$ if the intersection of the corresponding U_i's with $i \in \sigma$ is nonempty. That is,

$$\mathcal{N}(\mathcal{F}) = \{\sigma \subseteq I \mid \cap_{i \in \sigma} U_i \neq \emptyset\}.$$

The problem regarding the Čech complex $C_t(P)$ is that it is computationally expensive; the time required grows exponentially with the amount of points in P because finding out whether a high dimensional simplex is included in $C_t(P)$ requires to check all subsets of the same size. However, Theorem 3.1.6 shows that the Čech complex captures the topology of the original space up to homotopy.

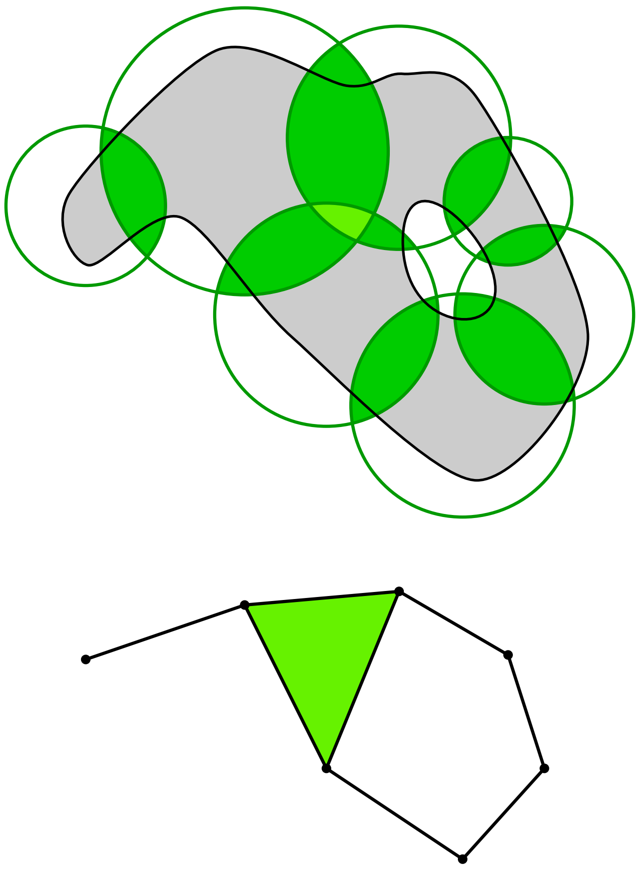

Figure 4: An example of the nerve of a covering. Here we let the family $\mathcal{F}$, consisting of open balls, be a covering of a topological space X. Each nonempty pairwise intersection of members of $\mathcal{F}$ results in a 1-simplex between the corresponding vertices, while the single nonempty triple intersection results in the 2-simplex, shown in light green.

The Vietoris-Rips complex is a simplicial complex built in a similar fashion:

Definition 2.1.3 (Vietoris-Rips complex). Let P be a finite subset of $\mathbb{R}^k$. For a given $t > 0$, the *Vietoris-Rips complex* $R_t(P)$ is the simplicial complex defined on the vertex set P with the following condition: a finite subset σ of P is a simplex of the Vietoris-Rips complex if all pairwise distances of points in σ are less than t, i. e.

$$R_t(P) = \{\sigma \subseteq P \mid d(x, y) < t \text{ for all } x, y \text{ in } \sigma\}.$$

The example in Figure 5 shows that, while both complexes can be constructed similarly using balls, they do not provide the same topology. While this is unfortunate, the main advantage of using the Vietoris-Rips complex is that it is a *clique complex*, i. e. it is uniquely determined by its 1-skeleton or 1-graph. This has resulted in more efficient algorithms by first constructing the 1-graph and expanding that into the complete Vietoris-Rips complex. See [22].

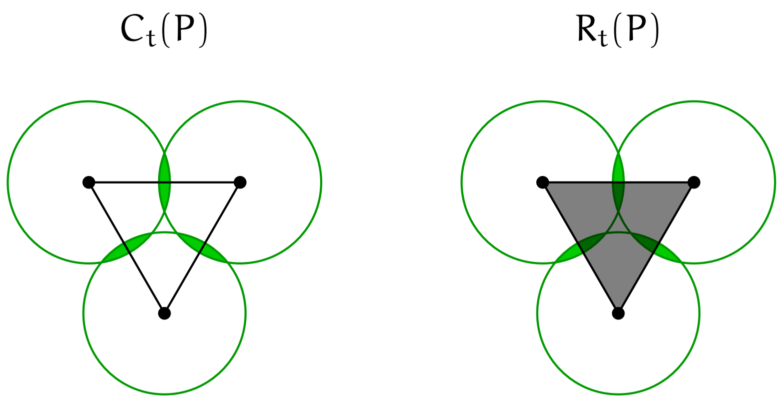

Figure 5: Comparison between the Čech and Vietoris-Rips complexes. Because the Vietoris-Rips complex is only concerned with pairwise intersections being nonempty, a 2-simplex is created, shown here in grey.

Definition 2.1.4 (Filtration). A filtration F is an indexed collection of subobjects F_i, where i runs over some ordered set, such that if $t \leqslant s$, then $F_t \subseteq F_s$.

Example 2.1.5 (Sublevelset filtration). Let X be a topological space and let $f\colon X \to \mathbb{R}$ be some function. Using f, we can define *sublevel sets* X_t as

$$X_t := \{x \in X \,|\, f(X) \leqslant t\}.$$

For $t \leqslant s$ we clearly have an inclusion $i_t^s\colon X_t \to X_s$ and thus, it is also a filtration. This is called the *sublevelset filtration* of (X, f) and we will denote it as $\mathbb{X}_{\text{sub}}$ or $\mathbb{X}_{\text{sub}}^f$.

We note that by varying the radius in the definition of the Čech complex we get a filtration of simplicial complexes, the *Čech filtration* $C(P) := (C_t(P))_{t \geqslant 0}$, indexed by the real numbers. Similarly, we define the *Vietoris-Rips filtration* to be $R(P) := (R_t(P))_{t \geqslant 0}$.

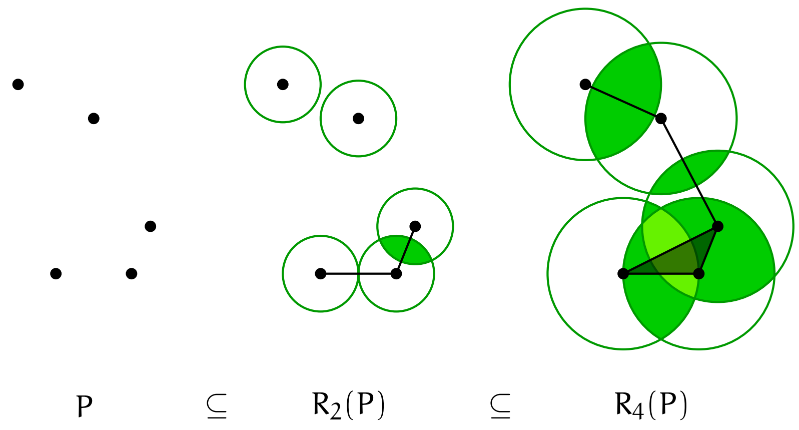

Figure 6: Example of a Vietoris-Rips filtration.

Proposition 2.1.6. *The Čech and Vietoris-Rips filtrations are* multiplicatively 2-interleaved, *i. e. for a set* P *there exist inclusions*

$$C_t(P) \subseteq R_t(P) \subseteq C_{2t}(P).$$

Proof. Let σ be a simplex of $C_t(P)$. We want to show that σ is also a simplex of $R_t(P)$. If σ is a simplex of $C_t(P)$, then the intersection of balls $\cap_{p\in\sigma} B_{t/2}(p)$ is nonempty. In particular, for any two points $p, q \in \sigma$ we have that the intersection $B_{t/2}(p) \cap B_{t/2}(q)$ is nonempty. Thus, there exists a point y in this intersection such that

$$d(p,q) \leqslant d(p,y) + d(y,q) \leqslant \frac{t}{2} + \frac{t}{2} = t.$$

We conclude that σ is also a simplex of $R_t(P)$.

For the second inclusion, let σ now be a simplex of $R_t(P)$. We want to show that the intersection of balls $\cap_{p\in\sigma} B_t(p)$ is nonempty. By definition we have that $d(p,q) < t$ for any two points $p, q \in \sigma$. In particular, for a fixed vertex p_0 in σ we have that $d(p_0, q) < t$ for q in σ. This shows that p_0 is in the intersection $\cap_{p\in\sigma} B_t(p)$. □

2.2 PERSISTENT HOMOLOGY

In this section we present the notions of persistence modules and its invariant, persistence diagrams.

2.2.1 *Barcodes and diagrams*

Definition 2.2.1 (Persistence module). A *persistence module* $\mathbb{V}$ over the real numbers is a functor from $(\mathbb{R}, \leqslant)$ (viewed as a category) to the category of vector spaces.

If s, t are members of $\mathbb{R}$ and $s \leqslant t$, i. e. there exists an unique morphism $s \to t$, we will often denote

$$V_t := \mathbb{V}(t), \text{ and}$$
$$v_s^t := \mathbb{V}(s \to t).$$

Definition 2.2.2 (Persistent homology). Let $X = (X_t)_{t\geqslant 0}$ be the a filtration of simplicial complexes and let H be the functor from simplicial complexes to vector spaces mentioned in Remark 1.2.14, which sends a simplicial complex K to its p-th simplicial homology group $H_p(K)$. We define the $p-$*th persistent homology* of X to be the persistence module $\mathbb{V} = H(X)$.

Explicitly, p-th persistent homology is just the diagram obtained by taking simplicial homology on the filtration X:

$$H_p(X_0) \to \cdots \to H_p(X_t) \to \cdots$$

We think intuitively of the real number t as a representation of time for X. In other words, the filtration represents a simplicial complex evolving over time; as we traverse along the filtration, more and more simplices are added, changing the structure of the complex. Persistent homology is therefore a way of representing this change in terms of homology.

An useful invariant of persistence modules is a multiset of intervals we call *persistence barcode* or *persistence diagram*. When applied to a filtered simplicial complex, the long bars of the barcode represent topological features which last over time, i. e. are "persistent", while the short bars represent noise and such in our data. There are multiple ways of defining these persistence barcodes; the construction below are based on [7] and [11].

For the purpose of this thesis, assume X to be the Vietoris-Rips filtration $R(P)$ with P a *finite* set. If we let $\mathbb{V}$ and H be as in Definition 2.2.2, then

- the vector spaces $H(X_t)$ are all finite-dimensional, and
- all but finitely many of the linear maps $H(i_t^s)\colon H(X_t) \to H(X_s)$ are isomorphisms.

Persistence modules which satisfy the first property are often referred to as *q-tame*, see [6].

The second property stems from there only being a finite amount of possible subcomplexes with vertex set P and thus, as t increases there are only finitely many "critical values" at which the subcomplex X_t changes, say $a_1 < a_2 < \ldots < a_n$. For $i = 1, 2, \ldots, n$, we are able to choose a basis B_i for each vector space V_{a_i}, such that the map $v_{a_i}^{a_{i+1}}$ is injective when restricted to B_i and

$$\operatorname{rank}(v_{a_i}^{a_{i+1}}) = \operatorname{card}\left(\operatorname{im}(v_{a_i}^{a_{i+1}}|_{B_i}) \cap B_{i+1}\right) \quad ([11], \text{Basis Lemma}).$$

We will refer to a persistence module which satifies both the properties above as *finite*.

Definition 2.2.3 (Persistence barcode). Let $\mathbb{V}$ be a finite persistence module and let $\{B_i\}_{i=1}^n$ be a choice of basis as above. Define the set

$$\mathcal{A} := \left\{(b, i) \,|\, b \in B_i, b \notin \operatorname{im}(v_{a_{i-1}}^{a_i}), i \in \{2, 3, \ldots, n\}\right\} \cup \left\{(b, 1) \,|\, b \in B_1\right\}$$

and the map $\alpha\colon \mathcal{A} \to \{1, 2, \ldots, n\}$:

$$\alpha\left((b, i)\right) := \max\left\{k \in \{1, 2, \ldots, n\} \,|\, (v_{a_{k-1}}^{a_k} \circ \cdots \circ v_{a_{i+1}}^{a_{i+2}} \circ v_{a_i}^{a_{i+1}})(b) \in B_k\right\}.$$

Given an element $(b, i) \in \mathcal{A}$, we refer to the integers i and $\alpha((b, i))$ as the *birth index* of basis element b and the *death index* of b, respectively.

Then the *persistence barcode* Bar($\mathbb{V}$) of $\mathbb{V}$ is defined as the multiset of intervals

$$\mathrm{Bar}(\mathbb{V}) := \left[[a_i, a_{j+1}) \mid \text{ there exists } (b, i) \text{ in } \mathcal{A} \text{ such that } a((b,i)) = j \right].$$

The multiplicity of an interval $[a_i, a_{j+1})$ in Bar($\mathbb{V}$) is the number of elements (b, i) in $\mathcal{A}$ such that $a((b,i)) = j$

We represent the barcode as a set of lines along one axis; this is the preferred visualization for the applications in Chapter 4. The alternative way is to plot it as a multiset of points lying on or above the diagonal in the real plane $\mathbb{R}^2$:

Definition 2.2.4 (Persistence diagram). Let $\mathbb{V}$ be a finite persistence module and Bar($\mathbb{V}$) be its persistence barcode. The *persistence diagram* Dgm($\mathbb{V}$) of $\mathbb{V}$ is defined as follows:

$$\mathrm{Dgm}(\mathbb{V}) := \left[(a_i, a_{j+1}) \in \mathbb{R}^2 \mid (a_i, a_{j+1}) \in \mathrm{Bar}(\mathbb{V}) \right].$$

The multiplicity of a point (a_i, a_{j+1}) in Dgm($\mathbb{V}$) is equal to the multiplicity of $[a_i, a_{j+1})$ in Bar($\mathbb{V}$).

2.2.2 *Distances and stability between persistence diagrams*

Recall the *supremum distance*: for points (p, q) and (p', q') in $\mathbb{R}^2$, we define $||(p,q) - (p', q')||_\infty := \max\{|p - p'|, |q - q'|\}$.

Definition 2.2.5 (Bottleneck distance). Given two persistence diagrams C and E, the *bottleneck distance* d_B between C and E is defined as follows:

$$d_B(C, E) := \inf \left\{ \sup_{c \in C} ||c - \phi(c)||_\infty \mid \phi \colon C \cup \Delta^\infty \to B \cup \Delta^\infty \text{ is a bijection} \right\}$$

Here Δ^∞ is the multiset containing all points on the diagonal with each of them having infinite multiplicity.

Definition 2.2.6 (ϵ-interleaving). Let $\mathbb{U}, \mathbb{V}$ be two finite persistence modules over the real numbers and $\epsilon \geqslant 0$. We say that $\mathbb{U}$ and $\mathbb{V}$ are *(additively) ϵ-interleaved* if there exists two families of linear maps

$$\{\phi_t \colon U_t \to V_{t+\epsilon}\}_{t \in \mathbb{R}}, \quad \{\psi_t \colon V_t \to U_{t+\epsilon}\}_{t \in \mathbb{R}}$$

such that the following four diagrams commute for $s \leqslant t \in \mathbb{R}$:

$$\begin{tikzcd}
\mathbb{U}_s \arrow[rr, "u_s^t"] \arrow[dr, "\phi_s"'] & & \mathbb{U}_t \arrow[rr, "u_t^{t+2\epsilon}"] \arrow[dr, "\phi_t"'] & & \mathbb{U}_{t+2\epsilon} . \\
& \mathbb{V}_{s+\epsilon} \arrow[rr, "v_{s+\epsilon}^{t+\epsilon}"'] & & \mathbb{V}_{t+\epsilon} \arrow[ur, "\psi_{t+\epsilon}"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
& \mathbb{U}_{s+\epsilon} \arrow[rr, "u_{s+\epsilon}^{t+\epsilon}"] & & \mathbb{U}_{t+\epsilon} \arrow[dr, "\phi_{t+\epsilon}"] & \\
\mathbb{V}_s \arrow[rr, "v_s^t"'] \arrow[ur, "\psi_s"] & & \mathbb{V}_t \arrow[rr, "v_t^{t+2\epsilon}"'] \arrow[ur, "\psi_t"] & & \mathbb{V}_{t+2\epsilon}
\end{tikzcd}$$

Remark 2.2.7. If $F = (F_t)_{t \geqslant 0}$ and $G = (G_t)_{t \geqslant 0}$ are two ϵ-interleaved filtrations of simplicial complexes, i. e.

$$F_{t-\epsilon} \subseteq G_t \subseteq F_{t+\epsilon},$$

then applying the p-th simplicial homology functor H to all such inclusion maps returns an ϵ-interleaving between the persistence modules $H(F_t)$ and $H(G_t)$ by definition.

Example 2.2.8. By changing to a log-scale in Example 2.1.6, i. e.

$$C_{\log(t)}(P) \subseteq R_{\log(t)} \subseteq C_{\log(t)+\log(2)},$$

we see by the previous remark that the Čech and Vietoris-Rips filtrations induces persistence modules which are $\log(2)$-interleaved on a log scale. We say the persistence modules are *log-interleaved* or *multiplicatively 2-interleaved.*

Definition 2.2.9 (Interleaving distance). Let $\mathbb{U}, \mathbb{V}$ be two finite persistence modules over the real numbers. The *interleaving distance* d_I between $\mathbb{U}$ and $\mathbb{V}$ is the pseudometric defined as:

$$d_I(\mathbb{U}, \mathbb{V}) := \inf\{\epsilon \geqslant 0 \mid \mathbb{U} \text{ and } \mathbb{V} \text{ are } \epsilon\text{-interleaved}\}.$$

Theorem 2.2.10 (Algebraic Stability Theorem). *Let $\mathbb{U}, \mathbb{V}$ be two finite persistence modules over the real numbers. Then*

$$d_B\left(\mathrm{Dgm}(\mathbb{U}), \mathrm{Dgm}(\mathbb{V})\right) \leqslant d_I(\mathbb{U}, \mathbb{V})$$

Proof. See [6], Stability theorem. □

Part II

THE SHOWCASE

3 REDUCING SIMPLICIAL COMPLEXES

3.1 NERVE THEOREM

The Nerve Theorem associates the homotopy type of a suitable topological space to the nerve of a good covering of that space. The following exposition is based on [2] and [20].

Definition 3.1.1 (Contractible carrier). Let K be a simplicial complex and X a topological space and suppose we have a map $C\colon K \to X$ which sends simplices of K to subspaces of X. We say C is a *contractible carrier* if

i) for every simplex σ in K, $C(\sigma)$ is contractible,

ii) C is order-preserving, i. e. if $\tau \subseteq \sigma$, then $C(\tau) \subseteq C(\sigma)$.

A continuous function $f\colon |K| \to X$ is *carried by* C if $f(|\sigma|) \subseteq C(\sigma)$ for all simplices σ in K.

Lemma 3.1.2 (Carrier Lemma). *Let K be a simplicial complex and X a topological space and suppose that C is a carrier from K to X. Then*

1) there exists a continuous function $f\colon |K| \to X$ carried by C.

2) if $f, g\colon |K| \to T$ are both carried by C, then $f \simeq g$.

Proof. See [20], Lemma 2.1. □

A *partially ordered set*, or a *poset*, can be assigned a simplicial complex structure.

Definition 3.1.3 (Order complex). Let P be a poset. The *order complex* $\mathcal{K}(P)$ is the simplicial complex having P as a vertex set and whose k-simplices are the k-chains $x_0 < x_1 < \ldots < x_k$ in P.

If Q is a poset, then for any q in Q we can define $Q_{\leqslant q}$ as the subposet $\{a \in Q \mid a \leqslant q\}$. For an order-preserving (or order-reversing) map $f\colon P \to Q$ between two posets P and Q, we say that $f^{-1}(Q_{\leqslant q})$ is a *fiber* of f.

Theorem 3.1.4 (Fiber Theorem). *Let $f\colon P \to Q$ be an order-preserving (or order-reversing) map. If all the fibers are contractible, then f induces a homotopy equivalence between $\mathcal{K}(P)$ and $\mathcal{K}(Q)$.*

Proof. Note that f induces a simplicial map on the order complexes, i. e. $f\colon \mathcal{K}(P) \to \mathcal{K}(Q)$. We want to show that f has a homotopy inverse map g.

Suppose that all the fibers are contractible; the map

$$\begin{aligned} C\colon \mathcal{K}(Q) &\longrightarrow |\mathcal{K}(P)| \\ \sigma &\longmapsto \left|f^{-1}(Q_{\leqslant \max \sigma})\right| \end{aligned}$$

is then a contractible carrier. By Lemma 3.1.2(a), there exists a continuous map $g\colon \mathcal{K}(P) \to \mathcal{K}(Q)$ carried by C. Explicitly, this means that $g(|\sigma|) \subseteq \left|f^{-1}(P_{\leqslant \max \sigma})\right|$ for every simplex in σ in $\mathcal{K}(Q)$.

We will show that g is a homotopy inverse; consider the maps

$$\begin{aligned} C'\colon \mathcal{K}(Q) &\longrightarrow |\mathcal{K}(Q)| \\ \sigma &\longmapsto |Q_{\leqslant \max \sigma}| \end{aligned}$$

and

$$\begin{aligned} C''\colon \mathcal{K}(P) &\longrightarrow |\mathcal{K}(P)| \\ \tau &\longmapsto \left|f^{-1}(Q_{\leqslant \max f(\tau)})\right| \end{aligned}$$

The first map C' is a contractible carrier and carries $f \circ g$ and $\mathbb{1}_{\mathcal{K}(Q)}$; for σ in $\mathcal{K}(Q)$ we have that

$$g(|\sigma|) \subseteq \left|f^{-1}(P_{\leqslant \max \sigma})\right| = f^{-1}(|P_{\leqslant \max \sigma}|),$$

because g is carried by C. Applying the map f results in the inclusion $(f \circ g)(|\sigma|) \subseteq C'(\sigma)$. Note that we also have the inclusion $|\sigma| \subseteq C'(\sigma)$.

Similarly, the second map C'' is a contractible carrier and carries $g \circ f$ and $\mathbb{1}_{\mathcal{K}(P)}$; for a simplex τ in $\mathcal{K}(P)$, the image $f(\tau)$ is a simplex of $\mathcal{K}(Q)$. Since g is carried by C, we have that

$$g(f(|\tau|)) \subseteq \left|f^{-1}(Q_{\leqslant \max f(\tau)})\right| = C''(\tau).$$

Note that we also have the inclusion $|\tau| \subseteq C''(\tau)$.

By Lemma 3.1.2(ii), we have that $f \circ g \simeq \mathbb{1}_{\mathcal{K}(P)}$ and $g \circ f \simeq \mathbb{1}_{\mathcal{K}(Q)}$. Thus, g is the desired homotopy inverse of f. ☐

Similar to the order complex, we can associate a partial order structure to a simplicial complex.

Definition 3.1.5 (Face poset). Let K be a simplicial complex. The *face poset* $\mathcal{P}(K) = (K, \subseteq)$ is the set of simplices of K ordered by inclusion.

We state the Nerve Theorem below; recall Definition 2.1.2 for the nerve $\mathcal{N}(\mathcal{U})$ of a covering $\mathcal{U}$.

We only prove the theorem for locally finite coverings; for the purpose of this thesis, this is adequate. See [3] for a proof of the general statement.

Theorem 3.1.6 (Nerve theorem). *Let (K, V) be a simplicial complex and $\mathcal{U} = \{K_i\}_{i \in I}$ be a good covering of K, i.e. any nonempty finite intersection of subcomplexes in $\mathcal{U}$ is contractible. Then K is homotopy equivalent to the nerve $\mathcal{N}(\mathcal{U})$.*

Proof. We assume that the covering $\mathcal{U}$ is locally finite, i. e. any vertex v in V is contained in only finitely many K_i's of $\mathcal{U}$. Let $P = \mathcal{P}(K)$ and $Q = \mathcal{P}(\mathcal{N}(\mathcal{U}))$ be the face posets of the simplicial complexes. Define the map

$$\begin{aligned} f\colon P &\longrightarrow Q \\ \sigma &\longmapsto \{i \in I \mid \sigma \in K_i\}. \end{aligned}$$

The map f is clearly order-reversing, i. e. $\tau \subseteq \sigma$ implies $f(\sigma) \subseteq f(\tau)$ for simplices τ, σ in P. The fiber $f^{-1}(Q_{\geqslant \rho})$ at ρ in Q is then equal to the intersection $\cap_{i\in\rho} K_i$ which is contractible by assumption. By Theorem 3.1.4, f induces a homotopy equivalence between $\mathcal{K}(Q)$ and $\mathcal{K}(P)$. □

3.2 QUOTIENT COMPLEX

The Rips filtration has been one of the preferred choices in topological data analysis. Despite its advantages, the filtration still grows exponentially with the amount of data points or vertices. Intuitively, the only way to reduce the size of the filtration is by limiting the vertex set, e. g. by removing points.

Another idea which might occur is to identify points, i. e. taking the quotient of the vertex set by some equivalence relation. By using a non-trivial relation the size will clearly be reduced, but the topology might change drastically. Similarly to the idea of the Vietoris-Rips complex, we are only allowed to relate points which are "close" enough. This idea will however be made more rigorous in the next chapter.

It turns out that this idea of a "quotient complex" works surprisingly well thanks to the Nerve theorem. We are able to show that its realization is homotopy equivalent to the realization of a larger simplicial complex, which is the main theoretical result of this thesis.

Definition 3.2.1 (Preimage of a simplicial complex). Let (L, W) be a simplicial complex with vertex set W and $f\colon V \to W$ be a map for some set V. We define the *preimage*, $f^{-1}(L) \subseteq \wp(V)$ of L to be the simplicial complex such that

$$\sigma \text{ is a simplex of } f^{-1}(L) \text{ iff } f(\sigma) \text{ is a simplex of } L.$$

The following lemma shows that intersection (Proposition 1.1.3), closure (Definition 1.1.4) and preimage (Definition 3.2.1) of simplicial complexes behave nicely (i. e. they commute) with each other.

Lemma 3.2.2. *Let (L, W) be a simplicial complex, $\{(L_i, W)\}_{i\in I}$ a collection of simplicial complexes with the same vertex set W and $f\colon V \to W$ a map for some set V. Then*

a) $\cap_{i\in I}\overline{L_i} = \overline{\cap_{i\in I} L_i}$

b) $f^{-1}(\cap_{i\in I}\overline{L}_i) = \cap_{i\in I} f^{-1}(\overline{L}_i)$

c) $f^{-1}(\overline{L}) = \overline{f^{-1}(L)}$

Proof. Recall by Definition 1.1.4, that the closure $\overline{L}$ of a set L is the simplicial complex consisting of all subsets of L. Thus a simplex of $\overline{L}$ is a subset of L.

a) We want to show that for a given subset $\tau \in \wp(W)$, τ is a simplex of the intersection $\cap\overline{L}_i$ if and only if τ is also a simplex of the closure $\overline{\cap L_i}$.

   If τ is a simplex of $\cap\overline{L}_i$, then it is also a simplex of each $\overline{L_i}$ for all i, i. e. $\tau \subseteq L_i$ for all i. Thus, τ is contained in the intersection $\cap L_i$. By definition of the closure, it follows that τ is a simplex of $\overline{\cap L_i}$.

   Conversely, if τ is a simplex of $\overline{\cap L_i}$, then it is also a subset of the intersection $\cap L_i$, i. e. $\tau \subseteq L_i$ for all i. Being a subset of each L_i means that τ is a simplex of the closure $\overline{L}_i$ for all i. It follows that τ is a simplex of $\cap\overline{L}_i$.

b) For a given subset $\sigma \in \wp(V)$ we want to show that σ is a simplex of the preimage $f^{-1}(\cap\overline{L}_i)$ if and only if σ is also a simplex of the intersection $\cap f^{-1}(\overline{L}_i)$.

   If σ is a simplex of $f^{-1}(\cap\overline{L}_i)$, then by Definitio 3.2.1 the image $f(\sigma)$ is a simplex of $\cap\overline{L}_i$, i. e. $f(\sigma) \subseteq L_i$ for all i. Thus, $f(\sigma)$ is a simplex of the closure $\overline{L}_i$ for all i and, by Definition 3.2.1, σ a simplex of $f^{-1}(\overline{L}_i)$ for all i. It follows that σ is a simplex of $\cap f^{-1}(\overline{L}_i)$.

   Conversely, if σ is a simplex of $\cap f^{-1}(\overline{L}_i)$, then it must be a simplex of $f^{-1}(\overline{L}_i)$ for all i. By Definition 3.2.1, $f(\tau)$ is a simplex of $\overline{L}_i$ for all i, i. e. $f(\tau) \subseteq L_i$ for all i. Thus, $f(\tau)$ is contained in $\cap L_i$, which, by Definition 3.2.1, implies that $\tau \in f^{-1}(\cap L_i)$.

c) We want to show that for a given subset $\delta \in \wp(V)$, δ is a simplex of the preimage $f^{-1}(\overline{L})$ if and only if δ is a simplex of the closure $\overline{f^{-1}(L)}$.

   If δ is a simplex of $f^{-1}(\overline{L})$, then by Definition 3.2.1 the image $f(\delta)$ is a simplex of $\overline{L}$. This implies that $f(\delta)$ is a subset of L which, by Definition 3.2.1, shows that δ is contained in $f^{-1}(L)$. It follows that δ is also a simplex of the closure $\overline{f^{-1}(L)}$.

   Conversely, if δ is a simplex of $\overline{f^{-1}(L)}$, then it is also a subset of $f^{-1}(L)$. By Definition 3.2.1 the image $f(\delta)$ is contained in L. Thus, it is also a simplex of $\overline{L}$. This implies, by Definition 3.2.1, that δ is a simplex of the preimage $f^{-1}(\overline{L})$.

$\square$

We now present the main result of the thesis.

Corollary 3.2.3. *Let* (L, W) *be a simplicial complex with vertex set* W *and* $f\colon V \to W$ *be a surjective map for some set* V*. Then* L *is homotopy equivalent to* $f^{-1}(L)$*.*

Proof. Consider the coverings $\mathcal{U} = \{\overline{\gamma}\}_{\gamma \in L}$ and $\mathcal{V} = \{f^{-1}(\overline{\gamma})\}_{\gamma \in L}$ of the simplicial complexes L and $f^{-1}(L)$, respectively. By surjectivity of f, the map $\overline{\gamma} \mapsto f^{-1}(\overline{\gamma})$ becomes a bijection between the covers.

The nerves of the coverings are isomorphic: indeed, if $\{\overline{\gamma}_i\}_{i=1}^n$ is a simplex of $\mathcal{N}(\mathcal{U})$, then by definition $\cap_{i=1}^n \overline{\gamma}_i$ must also be nonempty. Surjectivity of f implies that $f^{-1}(\cap_{i=1}^n \overline{\gamma}_i)$ is also nonempty. Moreover, we know that the preimage map "commutes" with the intersection of sets and thus $\cap_{i=1}^n f^{-1}(\gamma_i)$ must be nonempty too. It follows that $\{f^{-1}(\gamma_i)\}_{i=1}^n$ is a simplex of $\mathcal{N}(\mathcal{V})$.

In short:

$$\begin{aligned} \{\overline{\gamma}_1, \overline{\gamma}_2, \dots, \overline{\gamma}_n\} \in \mathcal{N}(\mathcal{U}) &\Leftrightarrow \cap_{i=1}^n \overline{\gamma}_i \neq \emptyset \\ &\Leftrightarrow f^{-1}(\cap_{i=1}^n \overline{\gamma}_i) \neq \emptyset \\ &\Leftrightarrow \cap_{i=1}^n f^{-1}(\overline{\gamma}_i) \neq \emptyset \\ &\Leftrightarrow \{f^{-1}(\overline{\gamma}_i)\}_{i=1}^n \in \mathcal{N}(\mathcal{V}). \end{aligned}$$

Lastly, we note that the coverings are good, i. e. all the nonempty intersections are contractible, as they all reduce to the closure of a simplex:

$$\cap_{i=1}^n \overline{\gamma}_i \overset{3.2.3a}{=} \overline{\cap_{i=1}^n \gamma_i} = \overline{\sigma}, \tag{$*$}$$

for some simplex $\sigma \in L$. Similarly

$$\begin{aligned} \cap_{i=1}^n f^{-1}(\overline{\gamma}_i) &\overset{3.2.3b}{=} f^{-1}(\cap_{i=1}^n \overline{\gamma}_i) \\ &\overset{3.2.3a}{=} f^{-1}(\overline{\cap_{i=1}^n \gamma_i}) \overset{(*)}{=} f^{-1}(\overline{\sigma}) \\ &\overset{3.2.3c}{=} \overline{f^{-1}(\sigma)} = \overline{\tau}, \end{aligned}$$

for some simplex $\tau \in f^{-1}(L)$.

Applying the Nerve Theorem (Theorem 3.1.6) gives us the desired result:

$$L \simeq \mathcal{N}(\mathcal{U}) \cong \mathcal{N}(\mathcal{V}) \simeq f^{-1}(L).$$

□

Consequently, Corollary 3.2.3 shows that in terms of homology, L and $f^{-1}(L)$ are equal.

We are finally ready to put all of this in the desired context of *reducing* a simplicial complex. Alluded to in the start of this section, this construction is based on reducing the vertex set by mapping it onto a smaller set, i. e. taking the quotient:

Definition 3.2.4 (Quotient complex). Let (K, V) be a simplicial complex with vertex set V and let $f\colon V \to W$ be a surjective map. Define the *quotient complex* L_f to be the set $\{f(\sigma) \mid \sigma \in K\}$.

The map f will also be referred to as the *quotient map*.

Note that, by the previous corollary, the quotient complex L_f is homotopy equivalent to $f^{-1}(L_f)$.

Remark 3.2.5. $K \subseteq f^{-1}(L_f)$.

Proof. Let σ be a simplex of K. From Definition 3.2.1 and Definition 3.2.4 it follows that

$$\sigma \subseteq f^{-1}(f(\sigma)) \in f^{-1}(L_f).$$

This shows that σ is contained in a simplex of $f^{-1}(L_f)$, i. e. $\sigma \in f^{-1}(L_f)$. $\square$

4 APPLICATION AND EXAMPLES

The goal of this chapter is to present an approximation of the Vietoris-Rips filtration in terms of persistent homology. That is, we will construct a filtration which is (additively) interleaved with the Vietoris-Rips filtration. This is done by using the quotient complex construction, with a quotient map fulfilling some properties, on each of the members in the Vietoris-Rips filtration. Note that we are able to use the same map as all the members share the same vertex set.

We will end the chapter with an algorithm for creating one such quotient map. This will be accompanied by some few examples where we compare the resulting barcodes of each filtration.

4.1 APPLICATION ON THE VIETORIS-RIPS FILTRATION

Given a finite dataset $P \subseteq \mathbb{R}^n$, we will from now on denote the simplicial complex $(K, V) := (R_t(P), P)$. We need to choose a function $f\colon V \to W$ such that the resulting quotient complex is "nice", i.e. it retains most of the topological features. This comes in the form of an *ϵ-restriction* on f:

For any two points $v, v' \in V$, we have the following implication: $f(v) = f(v') \Rightarrow d(v, v') < \epsilon$ for some positive real number ϵ.

The restriction essentially means that the only points which f identifies are within ϵ-distance. This makes intuitively sense as we clearly cannot identify points too far from each other without changing the topology. We can also think of f as a clustering algorithm as it shares similar features.

Proposition 4.1.1. *Let (K, V) be as above and assume that $f\colon V \to W$ is a surjective map which fulfills the ϵ-restriction. Define the quotient complex L_f as from Definition 3.2.4 and change the indexing such that $L_t := L_f$. Then we have the following inclusion:*

$$f^{-1}(L_t) \subseteq R_{t+2\epsilon}(P).$$

Proof. Let τ be a simplex of $f^{-1}(L_t)$. We want to show that τ is also a simplex of $R_{t+2\epsilon}(P)$. If $\tau \in f^{-1}(L_t)$, then we know that $f(\tau) \in L_t$ by definition of $f^{-1}(L_t)$. Being a simplex of L_t means there exists some σ in K such that $f(\sigma) = f(\tau)$ by the definition of L_t.

For arbitrary points $p, q \in \tau$, choose corresponding points $p', q' \in \sigma$ such that $f(p) = f(p')$ and $f(q) = f(q')$. Then we have by the triangle inequality (twice):

$$d(p, q) \leqslant d(p, p') + d(p', q') + d(q', q) = \epsilon + t + \epsilon = t + 2\epsilon.$$

Thus, any two points in τ have distance less than or equal to $t + 2\epsilon$. It follows that τ is a simplex of $R_{t+2\epsilon}(P)$. $\square$

If a map f fulfills the ϵ-restriction, then the corresponding preimage filtration will be interleaved with the Vietoris-Rips filtration:

Corollary 4.1.2. *The Vietoris-Rips filtration and the preimage filtration are 2ϵ-interleaved, i. e.*

$$R_t(P) \subseteq f^{-1}(L_t) \subseteq R_{t+2\epsilon}(P).$$

Proof. This follows from Remark 3.2.5 and Proposition 4.1.1. $\square$

Recall from Remark 2.2.7 that as a consequence, the persistent homology of the two filtrations are 2ϵ-interleaved. Since the quotient map f is surjective by definition, we have the diagonal homotopy equivalences in the following diagram

$$\begin{tikzcd}
R_t(P) \arrow[d, "f"] & \subseteq & f^{-1}(L_t) \arrow[dll, "\simeq"] & \subseteq & R_{t+2\epsilon}(P) \arrow[d, "f"] & \subseteq & f^{-1}(L_{t+2\epsilon}) \arrow[dll, "\simeq"] & \subseteq & \cdots \\
L_t & & & \subseteq & L_{t+2\epsilon} & & & \subseteq & \cdots
\end{tikzcd}$$

by Corollary 3.2.3. In terms of persistent homology, the two filtrations $(L_t)_{t\geqslant 0}$ and $\left(f^{-1}(L_t)\right)_{t\geqslant 0}$ become interchangable. We summarize this as the following corollary:

Corollary 4.1.3. *The persistent homology of the filtrations $(R_t)_{t\geqslant 0}$ and $(L_t)_{t\geqslant 0}$ are 2ϵ-interleaved.*

We use this fact extensively in the next section as computing the quotient filtration $(L_t)_{t\geqslant 0}$ is much easier than the corresponding preimage filtration $\left(f^{-1}(L_t)\right)_{t\geqslant 0}$.

4.2 ALGORITHM AND EXAMPLES

In this section we discuss and explain the implementation of the quotient complex. We use to our advantage that the quotient complex retains in some way the structure of the original simplicial complex, which, for our purposes, will be the Vietoris-Rips complex. Tools for computing the Vietoris-Rips complex are readily available and we are able to use this machinery *if* the resulting quotient filtration is indeed a filtration. We achieve this by fixing a quotient map f for each filtration index t such that a map between any two quotient complexes becomes an inclusion. For a discussion on quotient maps f_t dependent on filtration index t, see Chapter 5.

We have shown previously in Corollary 4.1.3 that for a quotient map satisfying the ϵ-restriction the resulting quotient filtration approximates the Vietoris-Rips filtration. That is, the bottleneck distance between the corresponding persistence diagrams is less than or equal to 2ϵ. We want an algorithm for constructing such a map while also being:

- lightweight and easy to implement. Obviously we do not want something more complicated than computing the whole Vietoris-Rips complex, as that defeats the purpose of an approximation;
- "locally maximal". In other words, the resulting quotient map cannot identify any more points without failing to satisfy the ϵ-restriction.

We present here a possible algorithm which satisfies both the desired properties. Refer to Figure 7 for intuition.

Assume the input to be a finite dataset $P \subseteq \mathbb{R}^d$. Choose an ordering on P, that is, $P = \{p_0, p_1, \ldots, p_n\}$, and create the distance matrix D of the points using the metric d inherited from $\mathbb{R}^d$ such that $D_{ij} = d(p_i, p_j)$. Algorithm 1 shows how to construct the desired quotient map f in terms of an equivalence relation on P.

Algoritm 1 : Algorithm for constructing a quotient map satisfying the ϵ-restriction for a given $\epsilon \geqslant 0$

```
1  QuotientMap($D, \epsilon$);
   Input  : A distance matrix D and a real number $\epsilon \geqslant 0$
   Output : A list $W$ of equivalence classes
2  $N \leftarrow$ length($D$)
3  $W \leftarrow \{\{0\}\}$
4  for $i \leftarrow 1$ to $N$:
5      $T \leftarrow$ false
6      foreach equivalence class $E$ in $W$:
7          if $D_{ie} < \epsilon$ foreach vertex $e$ in $E$:
8              $E \leftarrow E \cup \{i\}$
9              $T \leftarrow$ true
10             break
11     if not $T$:
12         $W \leftarrow W \cup \{\{i\}\}$
13 return $W$
```

We begin by initializing the set of equivalence classes W on line 3 such that it contains (part of) the first equivalence class $\{p_0\}$. Note that, in the algorithm, the points of P are only referred to by their indices. The *for*-loop on line 4 iterates through all the points of $P \setminus \{p_0\}$ and determines whether it can be related to any previously defined equivalence class.

At step i in the *for*-loop we compare the distance between the point p_i, with all the points of a given equivalence class $E \in W$. If all the distances are less than ϵ, then p_i can be added to the class E without breaking the ϵ-restriction on W. The *foreach*-loop on line 6 guarantees that we do this with all classes of W. If it turns out to be so that p_i cannot be added to any of the equivalence classes we add it to W as its own class on line 12. The resulting output W satisfies the ϵ-restriction by construction. Note that the algorithm depends on the ordering chosen on the dataset P.

To construct the corresponding quotient filtration we just compute the Vietoris-Rips complex with W as the vertex set; we define the *quotient distance* d_Q between any two classes w, w' in W to be

$$d_Q(w, w') := \min\{d(v, v') \mid v \in w \text{ and } v' \in w'\},$$

where d is the aforementioned metric on P. Algorithms for calculating the quotient distance and the associated distance matrix are given in Algorithms 3 and 2, respectively.

Algoritm 2 : Algorithm for constructing a distance matrix based on the equivalence classes

```
1 QuotientMatrix($D, W$);
  Input  : A distance matrix $D$ and a list of equivalence
           classes $W$
  Output : A distance matrix $Q$
2 $N \leftarrow \text{length}(W)$
3 $Q \leftarrow \text{matrix}(N, N)$
4 for $i \leftarrow 0$ to $(N-1)$:
5     for $j \leftarrow i$ to $(N-1)$:
6         $M \leftarrow \text{MinimumDistance}(D, i, j, W)$
7         $Q_{ij} \leftarrow M$
8 return $Q$
```

Algoritm 3 : Algorithm for computing the minimal distance between equivalence classes i and j

```
1 MinimumDistance($D, i, j, W$);
  Input  : A distance matrix $D$, positive integers $i, j$ and a list
           of equivalence classes $W$
  Output : A real number $m$.
2 foreach vertex $v$ in $W[i]$:
3     foreach vertex $w$ in $W[j]$:
4         $L \leftarrow L \cup \{D_{vw}\}$
5 $m \leftarrow \min\{L\}$
6 return $m$
```

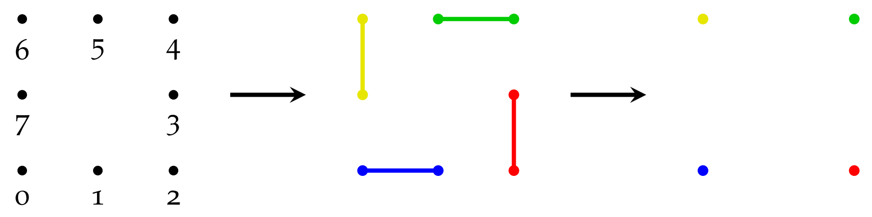

Figure 7: Intuition behind the algorithm for $\epsilon = 1.5$. Let V be the set of eight points, creating the outline of a 2×2-square, at the first step of the figure and let $\epsilon = 1.5$. The chosen ordering of the points are as shown. We start by setting the first point 0 as its own equivalence class and continue along the remaining points in the given ordering: The point 1 is within ϵ-distance of the point 0 and thus, we can identify the two points, shown here in blue at the middle step. Next up we look at the point 2; it is indeed within ϵ-distance of the point 1, but we have to consider the equivalence class of 1. The distance between points 0 and 2 is greater than ϵ, so 2 starts its own equivalence class, shown here in red. Continuing in this fashion we end up with the four equivalence classes, i. e. the four points at the last step, as the new vertex set W. Note that the last step does not give a realistic representation of the distances between points.

4.2.1 *Examples*

In this section, we compare the performance of our various algorithms on synthetic data. For the purpose of this thesis we have implemented an algorithm for constructing the Vietoris-Rips filtration and although it might not be efficient, comparing it with the quotient filtration is still valid; the filtrations themselves are constructed in the same manner and thus, the efficiency of both algorithms are heavily correlated.

All our implementations are done in Python through a Jupyter Notebook. For computing persistent homology we use a few tools from the Python package *phat* [1] and all the timings are measured with the built-in magic command *%time* in Jupyter. The specifics of the implementations and code are available in the Appendix.

The datasets found in this section are all taken from [24] and we have chosen here to use the value $\epsilon = 2$ for each of them. As our implementation of the Vietoris-Rips filtration only considers simplices up to dimension $k = 2$, the resulting barcodes only show the 0th and 1st persistent homology in blue and orange, respectively.

When the filtration value increases beyond a certain value, the resulting simplicial complex becomes simply connected. In the context of barcodes this means that all but one bar, i. e. the main path-component, should end at finite filtration values. One might however notice that this bar is missing in all the barcodes and this is just done by choice.

The first dataset *Spiral* consists of 312 points forming three distinctive and disjoint strands of points spiralling towards the center, simi-

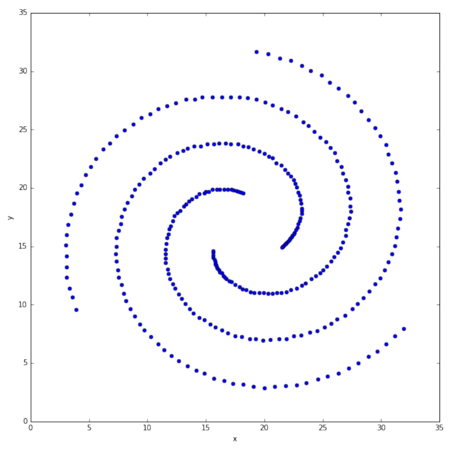

Figure 8: The *Spiral* dataset

lar to a hurricane or a cyclone seen from above. The path-components corresponding to these are represented in the barcode in Figure 9 as the two long bars (and the hidden main bar) of the 0th persistent homology. Lastly we also see a long bar in the 1th persistent homology, which represents the "eye" of the storm. As the three components merge into one component, a hole is created in the middle which becomes a persistent generator in our barcode.

The barcode for the corresponding quotient filtration is quite similar, more than what should be expected. As ϵ here is equal to 2, and the interleaving distance between the two persistence modules is 2ϵ, we should expect the difference in the bar lengths to vary by a similar amount; however, this is not the case. The only noticeable difference seems to be the amount of simplices; the topological information that we can infer from both barcodes is identical.

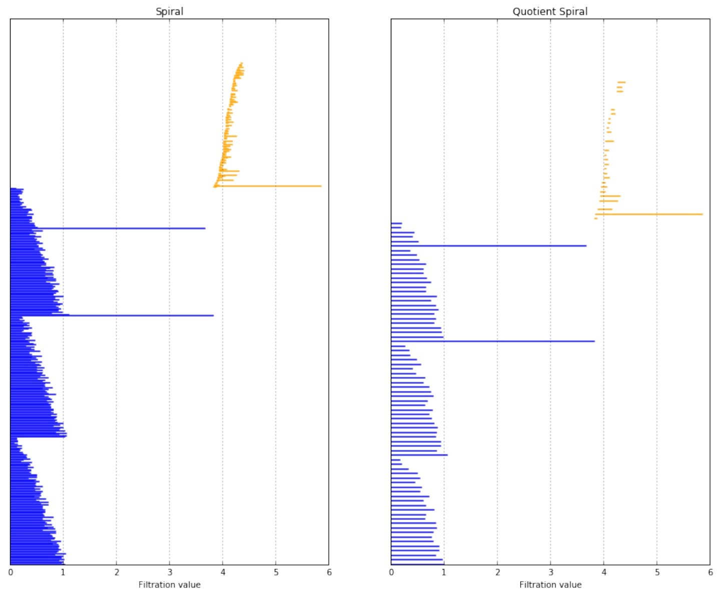

Figure 9: The barcodes for "Spiral" with $\epsilon = 2$

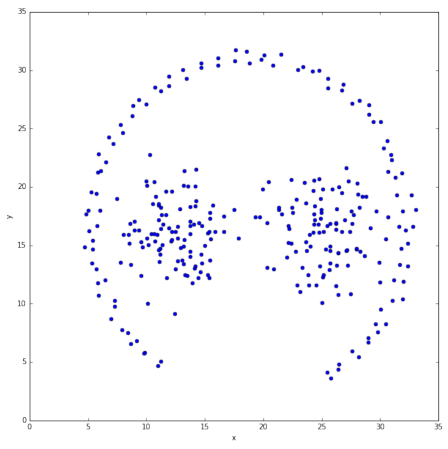

Figure 10: The *Path-based1* dataset

Our second dataset *Path-based1* consists of exactly 300 points. Similarly to *Spiral* there seems to be three components, one being in the shape of an incomplete circle while the other two being clusters inside the circle. This is however not easily seen in the barcode in Figure 11, as all three of them are too close to each other and thus the bars representing these three components are seen as noise in the barcode. However, we can get some information, as the big hole at the topside of the circle gets created at an early filtration value, which makes it the most persistent topological feature of the dataset, shown here as the single long bar of the barcode.

While there seem to be a few differences in the shorter bars for the quotient filtration there is again no significant change in the long bars. Compared to the original barcode, it is practically the same.

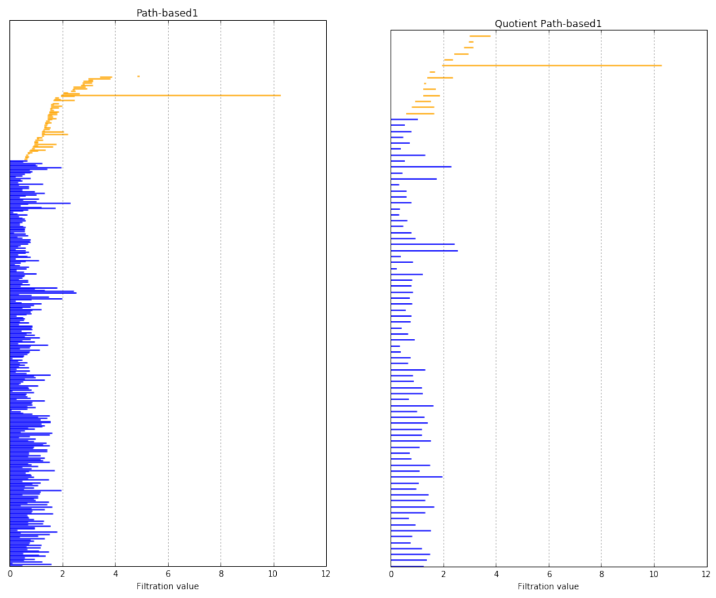

Figure 11: The barcodes for "Path-based1" with $\epsilon = 2$

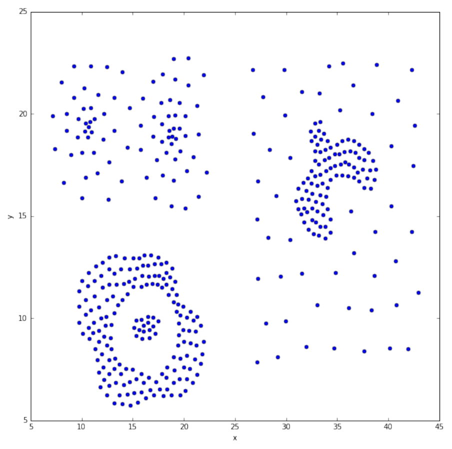

Figure 12: The *Zahn's Compound* dataset

Lastly, we consider the more unique dataset with 399 points; *Zahn's Compound*, a collage of various elements. The lower left of this dataset can only be described as a fried egg with the outer rim of the yolk forming a hole. The width of this yolk is relatively small so we do not expect this feature to be very persistent in the barcode in Figure 13. Indeed, the corresponding bar, born at approximately filtration value 0.6 and ending at filtration value 2.3, is too short to be considered as anything but noise in our dataset.

Above the egg we see two disk-shaped clusters of points. The distance between these is very small and we expect them to merge at an early filtration value, most probably together with the egg mentioned earlier. This can be seen in the barcode as the two longest bars in the 0-th persistent homology. Note that, while the shorter one of the two does represent the egg component, it is too short to be considered a persistent topological feature of the dataset. The last and rightmost element of the dataset is a high density cluster lying amidst a sea of points. This cluster is represented in the barcode as the hidden main bar.

The 1st persistent homology has a relatively long bar that we have not discussed, appearing at a later filtration value. We can see that this bar is born when there is only one path-component left and it represents the hole appearing in the middle of our dataset between the three clusters.

While the other quotient barcodes mostly resembled the original Vietoris-Rips barcodes with a fewer amount of bars, the lower section of the quotient barcode in Figure 13 has changed. This illustrates the effect of which the parameter ϵ has on different datasets. The aforementioned sea on the right side of the dataset has a low density, meaning that the distances between each point and its neighbours are relatively high. Because Algorithm 1 is only able to identify points within ϵ-distance, we can see, by choosing $\epsilon = 2$, that very few of the points in the sea are affected by the algorithm. Thus, the bars at the bottom of both barcodes are mostly identical, the only difference being the scale of the y-axis. The important point is that all the long bars remain unchanged.

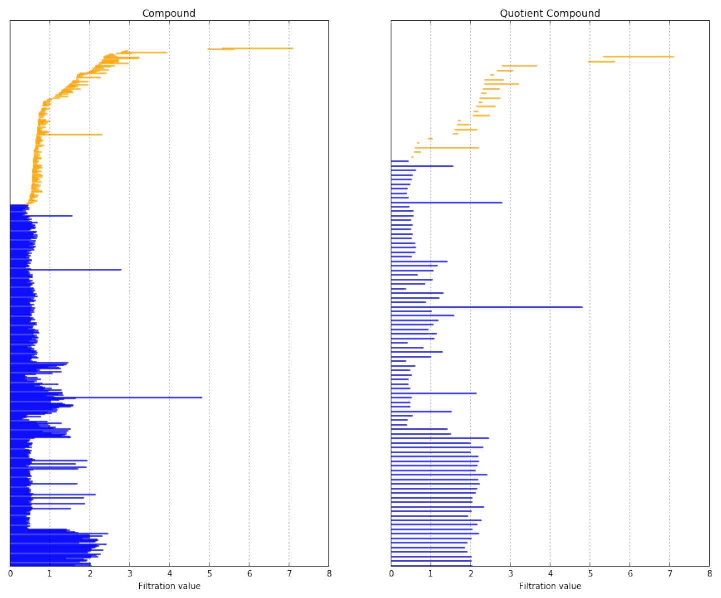

Figure 13: The barcodes for "Zahn's Compound" with $\epsilon = 2$

Table 1 and Table 2 compare the efficiency between computing the filtration and barcodes for the Vietoris-Rips filtration and the quotient filtration. The results are as expected: as both algorithms construct the filtration (and barcodes) in the same fashion, the filtration with the smaller starting vertex set is faster to compute. Note that the times for the quotient filtration in Table 1 also include the time for running Algorithm 1 to construct the quotient map.

Dataset	Vietoris-Rips Time	Quotient Time
Spiral	3min 19s	4.61s
Path-based1	2min 47s	4.47s
Zahn's Compound	6min 38s	7.69s

Table 1: Table of times for computing the various filtrations

Dataset	Vietoris-Rips Time	Quotient Time
Spiral	5min 46s	1.9s
Path-based1	3min 39s	1.65s
Zahn's Compound	14min 40s	2.87s

Table 2: Table of times for computing the various persistence barcodes with *phat*

5 AFTERTHOUGHTS

In this thesis, we presented an approach to approximate the Vietoris-Rips filtration by taking the quotient of the vertex set. The algorithm produced a quotient filtration, shown to be 2ϵ-interleaved with the Vietoris-Rips filtration if the quotient map satisfied the ϵ-restriction.

The results from the previous chapter showed that the efficiency of computing the filtration and barcodes were greatly improved, even when the time used to construct the quotient map were taken into consideration. In fact, the construction of the quotient map were largely irrelevant taking less than a fraction of a second to compute, as seen in Table 3. The main improvement stemmed from the reduced vertex set. There were also promising tendencies to be seen in the barcodes themselves: by choosing an appropriate value for ϵ, the differences between the original barcode and the quotient barcode became minuscule. As all the long bars were identical in terms of length and no new long bars were introduced, the topological features discerned between the barcodes remained the same.

We mentioned in Section 4.2 the idea of quotient maps f_t dependent on filtration values t. Unfortunately, the result after applying these maps to a Vietoris-Rips filtration may not remain a filtration, i. e. the connecting maps will not necessarily be inclusions. All the maps, however, are simplicial maps and there exist various methods for computing topological persistence of such sequences. See [8] and [9].

An attempt was made to implement Algorithm 1 having ϵ as a linear function of t instead, but this did not prove to be very efficient and the idea was scrapped early.

Dataset	Time
Spiral	315ms
Path-based1	287ms
Zahn's Compound	596ms

Table 3: Table of the times for Algorithm 1

5.1 future work

While we did achieve a positive result approximating the Vietoris-Rips filtration in this thesis, there is a lot which could be improved and more work that could be done.

The implementations were done admittedly with little prior knowledge of Python and thus, they are most likely not an aid for anyone other than the author. Implementations for the Vietoris-Rips filtration are readily available elsewhere in a more refined form and we advise the interested reader to use those whenever possible.

The proposed algorithm for constructing the quotient map relies on a chosen ordering of the points. Because the datasets in Section 4.2.1 are read in as a list of coordinates, it is natural to order the points as they appear in the list. This is what we decided to do, but one could try to optimize this choice even more. An alternative could be to make a different algorithm altogether which might be able to incorporate dependent quotient maps better.

In addition to changing the implementation we would also have liked to have done more comparisons, either by varying the parameter ϵ in our algorithm or performing a similar analysis on other datasets. The three datasets used in this thesis were all synthetic, which may have caused the results to be an inaccurate representation of the quotient filtration's effectiveness.

We note that the quotient complex construction can be applied to any simplicial complex, so another possibility would have been to apply this to some other filtration, e. g. the Čech filtration. The ϵ-restriction defined in Section 4.1 only works for the case of the Vietoris-Rips filtration. For any other filtration, one has to define a similar restriction to make the resulting quotient filtration interleaved with the desired filtration.

Lastly, we would have liked to explain the barcodes a bit more. Looking back at the results, we concluded that the approximation showed no loss in topological information and while this remains true, the remarkable fact was that none of the long bars had changed even the slightest. Because the persistence modules of the filtrations were 4-interleaved, we expected this to show up in the barcodes. This did not happen for any of our datasets and it would be interesting to be able to understand this peculiarity.

Part III

APPENDIX

We present here the Jupyter Notebook (converted to LaTeX) with all the documentation and source code for the implementation used in the thesis.

1 VIETORIS-RIPS FILTRATION

Here's the algorithm for computing the Vietoris-Rips filtration for a given dataset.

1.1 *Preamble*

Importing the necessary packages.

```
In [ ]: import numpy as np
        import pandas as pd
        from scipy.spatial.distance import cdist
        from timeit import default_timer as timer
```

1.2 *Zero*

Creating a function for listing the 0-simplices.

```
In [ ]: def simp0(df):
            simp = []
            N = len(df)
            for i in range(N):
                simp.append([i, 0])
            return simp
```

1.3 *One*

Similarly, a function for the 1-simplices. We extract the filtration time out of a distrance matrix and sort the list by filtration time.

```
In [ ]: def simp1(df):
            simp = []
            N = len(df)

            #Loop for appending 1-simlices to the list "simp".
            for i in [x for x in range(N)]:
                for j in [x for x in range(N) if x > i]:
                    simp.append([i, j, df.values[i,j]])

            #Sorting based on filtration value.
            from operator import itemgetter
            simp.sort(key=itemgetter(2))
            return simp
```

1.4 *Two*

Lastly, a function for the 2-simplices. The algorithm consists of two steps: 1. Create a dictionary which stores the filtration time for each 2-simplex. Here we use the fact that the filtration time of a simplex is determined by the maximum filtration time of it's boundary. 2. Unload the dictionary above and write it out as a list, similar to what we did with the 0- and 1-simplices.

```
In [ ]: def simp2(simp0, simp1):
            ##Step1
```

```
    myDict = dict()
    N = len(simp0)

    for s in reversed(simp1):
        a = list(range(N))
        a = [x for x in a if x != s[0] if x != s[1]]
        for i in a:
            L = sorted([i,s[0],s[1]], key=int)
            key = ".".join(str(z) for z in L)
            if key not in myDict:
                myDict[key] = s[2]

    ##Step2
    simp = []
    step = list(myDict.items())
    for tup in step:
        spl = [int(x) for x in tup[0].split('.')]
        spl.extend([tup[1]])
        simp.append(spl)

    from operator import itemgetter
    simp.sort(key=itemgetter(3))
    return simp
```

1.5 *Input/Output*

Here we connect it all together; we read the given data file as a data frame using *pandas* and calculate the distance matrix. Then we use the previous functions to output the filtration.

```
In [ ]: def VRFiltration(inFile,outFile):
    #Read in file and calculate the distance matrix.
    table = pd.read_table(inFile, header=None)
    data = table.values
    data = data.astype(float)
    distMatrix = cdist(data,data)
    df = pd.DataFrame(distMatrix)

    #Calculating simplices and adding them to a list.
    sim0 = simp0(df)
    sim1 = simp1(df)
    sim2 = simp2(sim0,sim1)
    filtration = sim0 + sim1 + sim2

    #Output
    with open(outFile, 'w') as fileOut:
        for item in filtration:
            text = ''
            for i in item:
                text = text + str(i) + ' '
            text = text[:-1]
            fileOut.write(text + '\n')
```

2 QUOTIENT COMPLEX/FILTRATION

We introduce here an algorithm for constructing a surjective function f which does not depend on filtration value t.

2.1 *Preamble*

Importing packages which are needed for the code below.

```
In [ ]: import numpy as np
        import pandas as pd
        import itertools as it
        from scipy.spatial.distance import cdist
        from timeit import default_timer as timer
```

2.2 *Function*

Defining the desired function/relation.

```
In [ ]: def naiveRelation(df, epsilon):
            N = len(df)
            W = [[0]]

            #Looping over all points in the data.
            for i in range(1,N):

                #Boolean variable which tells us if we merged the point with a cluster.
                T = False

                #Checking if the point has distance less than epsilon from a cluster.
                for vertex in W:
                    if all(df.values[i,v] < epsilon for v in vertex):
                        vertex.extend([i])
                        T = True
                        break
                #Making point "i" into an own cluster if the above does not hold true.
                if not T:
                    W.append([i])
            return W
```

2.3 *Quotient filtration*

We can reuse the algorithms from the Vietoris-Rips section, but we need to create a new distance matrix. The problem lies in the fact that we're creating 1-simplices between clusters of vertices, which will lead to multiple filtration values for each 1-simplex. The function below chooses the minimum filtration value, i.e. the minimum distance between clusters and creates a distance matrix based on it.

```
In [ ]: def quotientDistance(df,rel):
            N = len(rel)
            dist = pd.DataFrame(index=range(N), columns=range(N))
```

```
        #Finding the minimum filtration value and adding it to the dataframe.
        for i in range(N):
            for j in [x for x in range(N) if x > i]:
                M = min([df.values[m,n] for m in rel[i] for n in rel[j]])
                dist.set_value(i, j, M)

        return dist
```

2.4 *Preimage filtration*

For completion, we'll also make a filtration of $f^{-1}(L_f)$ such that we can compare the barcodes.

```
In [ ]: def preDistance(simp, rel):
            #Creating empty dataframe to store distances/filtration value.
            N =  max([max(a) for a in rel])
            dist = pd.DataFrame(index=range(N+1), columns=range(N+1))

            #For each simplex, find the pre-images of points.
            for simplex in simp:
                pairs = [sorted([i,j],key=int) for i in rel[simplex[0]]
                                 for j in rel[simplex[1]]]

                #For each pair append it to a list together with the filtration value.
                for P in pairs:
                    dist.set_value(P[0],P[1],simplex[2])

            #We miss the pairs within a simplex, so we need to add the remaining zeros.
            return dist.fillna(0)
```

2.5 *Input/Output*

Here's the functions which reads and writes the two filtrations.

```
In [ ]: def quotientFiltration(inFile,outFile,epsilon):
            #Read in file and calculate the distance matrix.
            table = pd.read_table(inFile, header=None)
            data = table.values
            data = data.astype(float)
            distMatrix = cdist(data,data)
            df = pd.DataFrame(distMatrix)

            #Applying the relation.
            rel = naiveRelation(df,epsilon)
            #Calculating the simplicies and adding them to a list.
            sim0 = simp0(rel)
            sim1 = simp1(quotientDistance(df,rel))
            sim2 = simp2(sim0,sim1)
            filtration = sim0 + sim1 + sim2

            with open(outFile, 'w') as fileOut:
                for item in filtration:
                    text = ''
                    for i in item:
                        text = text + str(i) + ' '
                    text = text[:-1]
                    fileOut.write(text + '\n')
```

```
In [ ]: def preimageFiltration(inFile,outFile,epsilon):
            #Read in file and calculate the distance matrix.
            table = pd.read_table(inFile, header = None)
            data = table.values
            data = data.astype(float)
            distMatrix = cdist(data,data)
            df = pd.DataFrame(distMatrix)

            #Applying the relation and calculating the 1-simplices.
            rel = naiveRelation(df,epsilon)
            quoSimp = simp1(quotientDistance(df,rel))
            #Using the above to calculate the preimage.
            sim0 = simp0(df)
            sim1 = simp1(preDistance(quoSimp,rel))
            sim2 = simp2(sim0,sim1)
            filtration = sim0 + sim1 + sim2

            with open(outFile, 'w') as fileOut:
                for item in filtration:
                    text = ''
                    for i in item:
                        text = text + str(i) + ' '
                    text = text[:-1]
                    fileOut.write(text + '\n')
```

2.6 *Example*

We compute the filtrations defined above for the “Spiral” dataset.

2.6.1 *Spiral*

```
In [ ]: spiral = '/mnt/c/Users/Tam/OneDrive/Documents/Python/Data/spiral.txt'
        outSpiral = '/mnt/c/Users/Tam/OneDrive/Documents/Python/Data/spiralOutput.txt'
        quoSpiral = '/mnt/c/Users/Tam/OneDrive/Documents/Python/Data/spiralQuotient.txt'
        preSpiral = '/mnt/c/Users/Tam/OneDrive/Documents/Python/Data/spiralPreimage.txt'

In [ ]: %time VRFiltration(spiral, outSpiral)

In [ ]: %time quotientFiltration(spiral, quoSpiral, 2)

In [ ]: %time preimageFiltration(spiral, preSpiral, 2)
```

---

3 PHAT

3.1 *Functions*

Hastily made function for computing persistent homology.

```
In [ ]: import phat
```

Converting here from a text file (filtration) to persistence pairs.

```
In [ ]: def computePersistence(inFile):
            #Reading in file.
            print('Reading file')
            filtration = []
            with open(inFile) as my_file:
                for line in my_file:
                    fs = [float(f) for f in line.split()]
                    filtration.append(fs)

            #Sort filtration, first priority on filtration, second priority dimension.
            print('Sorting the filtration')
            filtration.sort(key=lambda obj: len(obj))
            filtration.sort(key=lambda obj: obj[-1])

            #Create a appropiate dictionary for remembering index.
            print('Creating the index-dictionary')
            indices = dict()
            for i in range(len(filtration)):
                #For each list, convert all the values except the last to integer type.
                inte = tuple([int(f) for f in filtration[i][:-1]])
                #Adding the index of simplex in the filtration as a dictionary value.
                indices[inte] = i

            #Creating boundary columns
            print('Creating the boundary columns')
            b_matrix = []
            N = len(filtration)
            for i in range(N):

                #Pick out the dimension
                ni = len(filtration[i])-1

                #Check if the dimension is 0.
                if (ni == 1):
                    b_matrix.append(tuple([0, []]))
                    continue

                boundary = []
                for a in range(ni):
                    dimma = list(range(ni))
                    dimma.pop(a)
                    da = tuple([filtration[i][k] for k in dimma])
                    boundary.append(indices[da])

                #Append the boundary
                b_matrix.append(tuple([ni-2, list(sorted(boundary))]))

            #Using PHAT to create the boundary matrix.
            print('Using PHAT')

            boundary_matrix = phat.boundary_matrix(
                representation = phat.representations.vector_vector)
            boundary_matrix.columns = b_matrix

            print('Calculating persitence pairs:')
            pairs = boundary_matrix.compute_persistence_pairs()
```

```
        pairs.sort()

        i = 0
        L = []
        for pair in pairs:
            if (filtration[pair[0]][-1] != filtration[pair[1]][-1]):
                L.append([pair, filtration[pair[0]][-1],filtration[pair[1]][-1]])
                i = i+1

        return L
```

---

4 COMPUTATIONS AND PLOTS

Using matplotlib to plot datasets and create barcodes from filtrations.

4.1 *Preamble*

```
In [ ]: import matplotlib.pyplot as plt; plt.rcdefaults()
        import numpy as np
        import pandas as pd
```

4.2 *Functions*

Function for plotting the dataset

```
In [ ]: def plot_data(inFile):
            table = pd.read_table(inFile, header=None)
            data = table.values
            data = data.astype(float)
            plt.figure(figsize=(10,10))
            plt.plot(data[:, 0], data[:, 1], 'bo', markersize=5)
            plt.xlabel('x')
            plt.ylabel("y")
            plt.title("")
            plt.show()
```

Functions for plotting the barcode. The second one is the one used for the final revision of the thesis.

```
In [ ]: def plotBarcode(barcode, name):
            i = 1
            time = 0
            for b in reversed(barcode):
                plt.plot([b[1], b[2]], [i, i])
                i = i+1
                if b[2] > time:
                    time = b[2]
            plt.axis([0, time+1, 0, i])
            plt.xlabel('Filtration value')
            plt.title(name)
            plt.show()
```

```
In [ ]: def plot_barcodes(pers, title):
            pers = pd.DataFrame(np.array(pers))
            pers.columns = ["kA", "birth", "death"]
            pers = pers.ix[pers.ix[:, "death"] != pers.ix[:, "birth"], :]
            pers["dim"] = 1 - (pers["birth"] == 0.0)
            pers["y"] = range(pers.shape[0])
            groups = pers.groupby("dim")

            fig = plt.figure(figsize=(6,10))
            ax = fig.add_subplot(111)
            colors = ['blue', 'orange']
            for name, group in groups:
                ax.plot((group.birth, group.death), (group.y, group.y),
                        colors[name], lw = 1.5, label=name)
            ax.set_xlabel("Filtration value")
            ax.set_ylabel("")
            ax.set_title(title)
            fig.tight_layout()
            plt.xlim(xmin=0)
            #ax.xaxis.set_ticks(ticks)
            ax.yaxis.set_ticks([])
            ax.grid()
```

4.3 *Example*

Testing the functions on the previous dataset.

4.3.1 *Spiral*

ORIGINAL

```
In [ ]: %%time
        %matplotlib inline
        os = computePersistence(outSpiral)
        plotBarcode(os, 'Spiral')
        plot_barcodes(os, 'Spiral')
```

QUOTIENT

```
In [ ]: %%time
        %matplotlib inline
        qs = computePersistence(quoSpiral)
        plotBarcode(qs, 'Quotient Spiral')
        plot_barcodes(qs, 'Quotient Spiral')
```

PREIMAGE

```
In [ ]: %%time
        %matplotlib inline
        ps = computePersistence(preSpiral)
        plotBarcode(ps, 'Preimage Spiral')
        plot_barcodes(ps, 'Preimage Spiral')
```

BIBLIOGRAPHY

[1] Ulrich Bauer, Michael Kerber, and Jan Reininghaus. *PHAT (Persistent Homology Algorithm Toolbox)*. https://bitbucket.org/phat-code/phat. [Online; accessed 17-April-2017]. 2013.

[2] Anders Björner. "Toplogical Methods." In: *Handbook of Combinatorics* 1,2 (1995), pp. 1819–1872.

[3] Anders Björner, László Lovász, Rade T. Zivaljevic, and Siniša T. Vrećica. "Chessboard Complexes and Matching Complexes." In: *Journal of the London Mathematical Society* 49.1 (1994), pp. 25–39. ISSN: 1469-7750. DOI: 10.1112/jlms/49.1.25. URL: http://dx.doi.org/10.1112/jlms/49.1.25.

[4] Corrie J. Carstens and Kathy J. Horadam. "Persistent Homology of Collaboration Networks." In: *Mathematical Problems in Engineering* (2013), p. 7. DOI: 10.1155/2013/815035. URL: http://dx.doi.org/10.1090/s0273-0979-09-01249-x.

[5] Joseph Chan, Gunnar Carlsson, and Raul Rabadan. "Topology of Viral Evolution." In: 110 (2013).

[6] Frederic Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. "The Structure and Stability of Persistence Modules." In: *ArXiv e-prints* (July 2012). arXiv: 1207.3674 [math.AT].

[7] Samir Chowdhury and Facundo Mémoli. "Persistent Path Homology of Directed Networks." In: *ArXiv e-prints* (Jan. 2017). arXiv: 1701.00565 [math.AT].

[8] Tamal K. Dey, Fengtao Fan, and Yusu Wang. "Computing Topological Persistence for Simplicial Maps." Proceedings of 30th Annual Symposium Computational Geometry. 2014.

[9] Tamal K. Dey, Fengtao Fan, Dayu Shi, and Yusu Wang. *SimPers: Software for Topological Persistence under Simplicial Maps*. http://web.cse.ohio-state.edu/~dey.8/SimPers/Simpers.html. [Online; accessed 04-June-2017].

[10] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010.

[11] Herbert Edelsbrunner, Grzegorz Jabłoński, and Marian Mrozek. "The Persistent Homology of a Self-Map." In: *Foundations of Computational Mathematics* 15.5 (2015), pp. 1213–1244. ISSN: 1615-3383. DOI: 10.1007/s10208-014-9223-y. URL: https://doi.org/10.1007/s10208-014-9223-y.

[12] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. "Topological Persistence and Simplification." In: *Discrete & Computational Geometry* 28 (2002), pp. 511–533. ISSN: 1432-0444. DOI: 10.1007/s00454-002-2885-2. URL: https://doi.org/10.1007/s00454-002-2885-2.

[13] John B. Fraleigh. *A First Course in Abstract Algebra*. 7th. Pearson, 2014.

[14] Chad Giusti, Eva Pastalkova, Carina Curto, and Vladimir Itskov. "Clique Topology Reveals Intrinsic Geometric Structure in Neural Correlations." In: *Proceedings of the National Academy of Science* 112 (2015), pp. 13455–13460. DOI: 10.1073/pnas.1506407112.

[15] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. URL: http://www.math.cornell.edu/~hatcher.

[16] Saunders Mac Lane. *Categories for the Working Mathematician*. 2nd. Springer-Verlag, 1971.

[17] Takenobu Nakamura, Yasuaki Hiraoka, Akihiko Hirata, Emerson G. Escolar, and Yasumasa Nishiura. "Persistent Homology and Many-body Atomic Structure for Medium-range Order in the Glass." In: *Nanotechnology* 26 (July 2015), p. 304001. DOI: 10.1088/0957-4484/26/30/304001. eprint: 1502.07445.

[18] Klaus B. Schebesch and Ralf W. Stecking. "Topological Data Analysis for Extracting Hidden Features of Client Data." In: *Operations Research Proceedings 2015: Selected Papers of the International Conference of the German, Austrian and Swiss Operations Research Societies (GOR, ÖGOR, SVOR/ASRO), University of Vienna, Austria, September 1-4, 2015*. Springer International Publishing, 2017, pp. 483 –489. ISBN: 978-3-319-42902-1. DOI: 10.1007/978-3-319-42902-1_65. URL: https://doi.org/10.1007/978-3-319-42902-1_65.

[19] Gurjeet Singh, Facundo Mémoli, and Gunnar Carlsson. "Mapper: a Topological Mapping Tool for Point Cloud Data." Proceedings of Eurographics Symposium on Point-based Graphics. 1991.

[20] James W. Walker. "Homotopy Type and Euler Characteristic of Partially Ordered Set." In: *European Journal of Combinatorics* 2 (1981), pp. 373–384.

[21] Jaejun Yoo, Eun Young Kim, Yong Min Ahn, and Jong Chul Ye. "Topological Persistence Vineyard for Dynamic Functional Brain Connectivity During Resting and Gaming Stages." In: *Journal of Neuroscience Methods* 267 (2016), pp. 1 –13. ISSN: 0165-0270. DOI: https://doi.org/10.1016/j.jneumeth.2016.04.001. URL: http://www.sciencedirect.com/science/article/pii/S0165027016300395.

[22] Afra Zomorodian. "Fast Construction of the Vietoris-Rips Complex." In: *Computer and Graphics* 34 (2010), pp. 263–271.

[23] Afra Zomorodian and Gunnar Carlsson. "Computing Persistent Homology." In: *Discrete & Computational Geometry* 33 (2005), pp. 249–274. ISSN: 1432-0444. DOI: 10.1007/s00454-004-1146-y. URL: https://doi.org/10.1007/s00454-004-1146-y.

[24] Pasi Fränti et al. *Clustering datasets*. 2015. URL: http://cs.uef.fi/sipu/datasets/.