École polytechnique fédérale de Lausanne

Audio identification using persistent homology

Master Project in Computational science and Engineering

Author:
Wojciech Reise

Supervisors : Professor Kathryn Hess Bellwald,
Professor Heather Harrington,
Dr. Mariano Beguerisse-Diaz.

March 3, 2024

Abstract

A fingerprint of a song is a descriptive summary of the audio file that encodes relevant information about that song. Signatures are tailored to exhibit particular invariants or properties, with specific applications in mind. One such application is an audio search engine: given a short, possibly noisy recording of a song, the service aims to retrieve the title and the name of the artist. Audio identification — at the core of audio search engines — has witnessed scientific interest, thanks to its potential industrial applications.

We aim to leverage the properties of persistent homology - a method from the field of topological data analysis, which has been successfully applied for classification and signal analysis in the context of dynamical systems. It is known for its robustness, as — in contrast to other statistical methods — it can capture and identify the global structure in noisy data.

In general, Audio identification is conducted using signatures of spectral representations of songs. In this work, we define fingerprints based on persistent homology of those representations. We present different approaches, depending on the desired type of robustness, including noise addition, frequency or time inversion. We propose two ways to fingerprint spectrograms locally, and devise a set of topological characteristics for each audio file. With classification algorithms in mind, we examine the vectorization and kernel methods defined for topological features and we evaluate our methods on a subset of the Million Song Dataset.

The persistent homology fingerprints that we propose prove useful in audio identification. Using local topological information, we obtain a system that exhibits robustness to obfuscations, allowing us to correctly retrieve the labels of a fraction of songs, in a simulated audio identification task.

Acknowledgements

I would like to thank Prof. Kathryn Hess Bellwald, my supervisor from École Polytechnique Fédérale de Lausanne, and Professor Heather Harrington from the Mathematical Institute at the University of Oxford, for providing me with the opportunity to complete my project as an academic visitor at the Mathematical Institute and supporting me throughout my stay.

I am grateful to Prof. Harrington and Dr. Mariano Beguerisse-Diaz, from Spotify, for their guidance and invaluable feedback. They remained available at every stage of the project, allowing me to pursue and develop my interests. They encouraged me to share my work with other researchers and gave opportunities to present it in academic and industrial contexts. I would also like to express great appreciation to Professor Ulrike Tillmann, who has helped me grasp the theoretical meaning of my experimental observations.

I wish to thank Prof. Vidit Nanda, Dr. David O'Sullivan, Bernadette Stolz, Jacob Leygonie and Oliver Vipond from the Mathematical Institute for their feedback on my work, and their suggestions, which lead to new ideas being explored in the scope of this project.

Finally, I want to acknowledge Dr. Martin Gould and Maria Dominguez from Spotify, who clarified how to treat audio data, shared invaluable experience and provided me with benchmarks for my results.

Contents

Chapter 1

Introduction and motivation

In this project, we aim to explore the applications of topological data analysis (TDA) for audio identification (audio id). To be more precise, the task consists of identifying a song - recovering its name and that of the artist, from a noisy snippet, spanning only a few seconds of the total track. This problem has been studied before with other methods [Wan03, BC07, YHS05]. In a collaboration with researchers from Spotify, we explore alternative solutions based on persistent homology, a method capturing characteristics that normal statistical approaches do not, making it a valuable addition to a data scientist's toolbox.

Audio id is typically based on spectral representations (spectrograms), which we review in chap. 2, and usually consists of two steps - fingerprinting and querying. Specifically, the fingerprint of a song needs to be robust to obfuscations and windowing. The query for a sample s - asking for the label (title and artist name), consists of a comparison of the fingerprint of that sample to other's fingerprints and returning the most likely (or closest) match. The Shazam system, described in [Wan03], relies on a test for whether the sample can be a time-shifted version of a song s' from the database. The fingerprint of each song consists of a set of statistically robust features, and the querying - of finding a plausible alignment between s and s', based on those features.Rather than the global-, local alignment is at the core of the method in [TB15]. It is a solution to cover identification, which is another song similarity detection, problem. In [YHS05], a computer-vision inspired audio id method is proposed. Fingerprint robustness is added by examining wavelet decompositions of time-windows of a spectrogram. Building on similar ideas, the work [BC07] improves on locally-sensitive hashing for querying and introducing an explicit model for false-positive matching. The success of these methods proves the efficacy of spectrograms based approaches.

Topology is the study of properties of spaces, which are invariant to transformations like deformations or stretching. One example of the topological descriptors - tools used for such studies, are homology groups, and the associated Betti numbers. They represent the number of 'holes' of various dimensions (connected components, loops, cavities) in a given shape. The roughness of descriptors translates into robustness and can capture common characteristics in distinct shapes, what makes computational homology an attractive tool to study data. Persistent homology (PH)— an extension of homology proposed by Edelsbrunner et al [ELZ02] and reviewed in chap. 3.3, has been successfully used in applications, such as in problems from material sciences [HNH$^+$16], object recognition [LOC14, RHBK14].

It has been also used for classification [LJY16] or parameter inference in time-series analysis - a setting seemingly similar to audio id. An overview of applications is provided by [GZ18]. A methodology for musical instrument identification has been put forward by Sanderson et al. in [SSM$^+$17]. The authors characterise topologically the time-series produced when playing a fixed note on different instruments and use these characteristics for classification. They build a shape based on delay-embeddings [dSSVJ12] and use persistent rank functions introduced by Robins and Turner in [VK16]. The success of the methods based on topology in the context of time-series analysis relies on some periodicity or recurrence. Songs exhibit such a structure to a lesser extent.

Due to the complexity of songs on one hand, and the success of audio id techniques based on spectral representations, we propose to apply persistent homology on spectral representations of songs. We picture the flow we propose in fig 1.1 and detail our contribution, which is twofold. In Section 4.1, we propose five types of filtration on spectrograms - one intensity filtration, two axis filtrations (time and frequency) and two

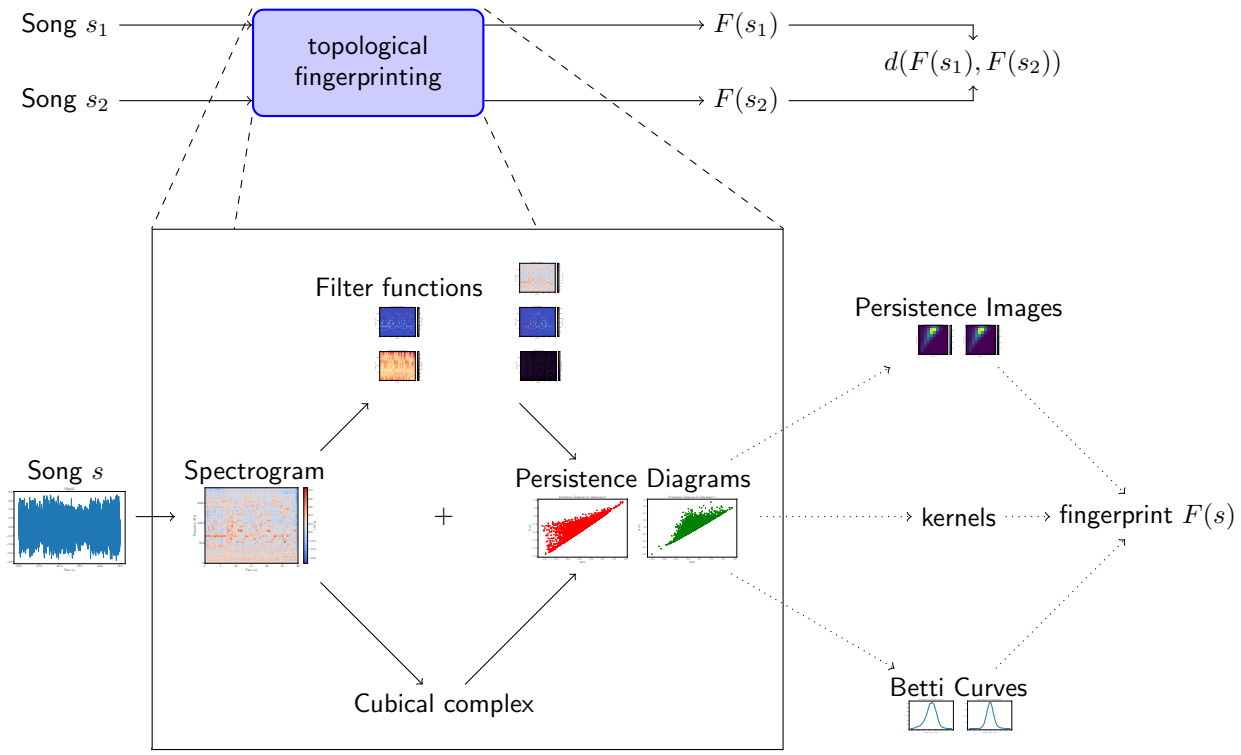

Figure 1.1: Our fingerprinting and comparison method. Given a song, we compute a spectral representation, and, using a filter function, its topological fingerprint. We compare these signatures with distances induced by kernel or vectorisation methods.

distance transform-based (on intensity and time-edge level-sets) ones. Second, we provide two scopes to analyse spectrograms: a global and a local one, and we evaluate the proposed filtrations. Finally, we comment on existing techniques to treat the results of persistent homology for classification. We evaluate two vectorization (Betti Curves, Persistence Images [AEK+17]) and two kernel methods (Sliced-Wasserstein [CCO17], Persistence Fisher kernels [LY18]), for persistence diagrams (PDs) in the context of those filtrations. Using GUDHI [Dlo15] and the sklearn tda package [Car18], we compare the performance of our methods on a subset of the Million Song Dataset [BMEWL11].

This report is organized as follows: in chap. 2, we provide an overview of spectral representations for audio data and elaborate on the current audio id methodologies. Chapter 3 describes the topological methods we rely on. After an introduction of persistent homology in sec. 3.3, we review previously-proposed vectorization methods in sec. 3.5. In Section 4.1, we propose filtrations on spectrograms, defining the corresponding fingerprints and their applications on parts of the spectrogram (time windows, or patches around computer-vision detected edges) in sec. 4.3. We start sec. 4.5 by outlining the challenges when applying vectorization methods for the newly-introduced filtrations and, using that insight, analyse audio id classification results for a subset of the Million Song Dataset.

Chapter 2

Spectral representations of audio

Spectral representations are common in signal processing, and in particular, in audio id, as they can reflect the perception of sound. A song is dynamic - its constituent frequencies change over time. To capture this, rather than performing a spectral decomposition of the whole signal, it is decomposed on small, successive, overlapping windows.

Setting a sampling rate $f_s = 11025Hz$, an initially-continuous song is represented as a vector $s = (s(t_k))_{k=1}^N = (s_k)_{k=1}^N \in \mathbb{R}^N$ encoding the sound wave at discrete time steps, where $N = Tf_s$, $t_k = k/f_s$ and T is the duration of the song.

Definition 1. Consider a song s and a vector $w \in \mathbb{R}^{N_FT}$ representing a window function. Fix a hop length h and denote $N_T = \lfloor N/h \rfloor + 1$. The spectrogram of s is its Short-Time Fourier Transform (STFT): a complex-valued matrix, denoted by $STFT(s) = S \in \mathbb{C}^{(1+N_{FT}/2)\times N_T}$, where

$$STFT(s)_{N_f,k_0} = \sum_{k=-N_{FT}/2}^{N_{FT}/2} s_{k+hk_0} w_k \exp(-2\pi i k N_f / N_{FT}). \tag{2.1}$$

For example, the entry S_{N_f,k_0} corresponds to the magnitude of the N_f-th frequency bin (that is, frequencies $f \in \left](N_f-1)\frac{f_s}{N_{FT}}, (N_f+1)\frac{f_s}{N_{FT}}\right[$) in the signal $\left(s_{-N_{FT}/2+hk_0} w_{-N_{FT}/2}, \ldots, s_{-N_{FT}/2+hk_0} w_{-N_{FT}/2}\right)$. The aim of introducing the window vector w, centred at k_0, is to restrict the time range that impacts the column $S_{:,k_0}$. The components of a window vector are evaluations of a continuous function with a maximum at 0. This is what makes the spectral decomposition local. One common choice is the Hann window described in [Jul11, sec.4.2.1],

$$w(k) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi k}{N_{FT}-1}\right)\right), \qquad k = \{-N_{FT}/2, \ldots, N_{FT}/2\}.$$

The length of the window function is defined by the desired frequency resolution $\frac{f_s}{N_{FT}}$. For a hop length h, two successive columns of the spectrogram are $\frac{h}{f_s}$ seconds apart.

Remark. One faces the time-frequency resolution trade-off [RJ58]. Increasing the time resolution requires decreasing the window size (at fixed f_s), compromising Δf. Using a logarithmic spacing of frequencies, we can afford to lose frequency-resolution higher in the scale, as the quantity of interest is the quality factor $Q = \frac{f_k}{\Delta f_k}$, where f_k, Δf_k are the location and resolution of the k-th frequency bin filter. This lead to the introduction of the constant-Q Transform (CQT) [Bro91], which, in contrast to the STFT, keeps the quality factor constant.

We adopt an alternative to the CQT transform, in which, after a STFT, we change the frequency bins (to the mel scale) and apply a weighting filter column wise: the equal loudness filter. The result of this process is shown in fig 2.1 (left). The mel scale, introduced by Stevens et al. in [SVN37], has been defined so as two frequencies are perceived by listeners as equidistant. Therefore, changing bins from the linear to those defined by the mel scale should represent the same information on a more natural way, which we refer to as the mel spectrogram. In practice, a set of filters (filter bank), with a triangular frequency response each, is defined, and represented as a matrix. Its product with the STFT gives the mel spectrogram and is in fact a basis change.

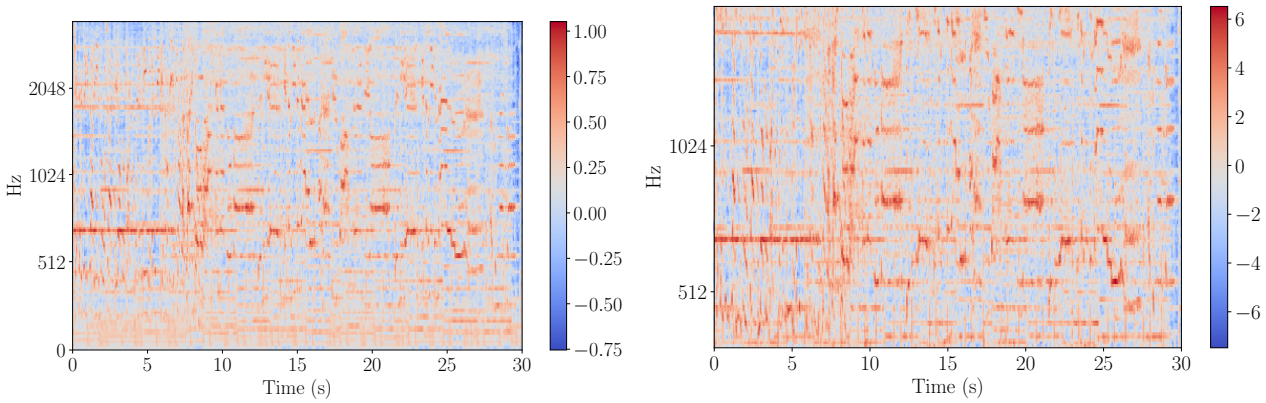

Figure 2.1: Spectrograms, plotted on the mel frequency scale. On the left, a spectrogram with the equal-loudness filter (ISO 226). The slightly red colour in the lower region shows a consistent bass component. On the other hand, the intensities of bins in the mid and higher frequencies exhibit greater variances. The line at 614 Hz indicates that there is a consistent component, appearing at the beginning of the snippet and also at $18-20$ and $22-25$ seconds. On the right, the same mel spectrogram, at which, instead of applying the ISO 226 filter, we look at the log scale, relative to its median: $\log(S/\operatorname{median}(S))$.

Another modification of the standard STFT consists of re-weighting the intensities according to the frequency. The human ear is thought to under estimate lower frequencies and over estimate higher ones, compared to the discrete Fourier transform. A standard has been established by Suzuki et al. in [SMR+03] to correct for this discrepancy.

Remark. An alternative to re-weighting is to cut-off the low-frequencies. In particular, for audio id, one can argue that the corresponding coefficients do not contain discriminatory information- they tend to be similar among songs.

The STFT is a 'local version' of the Discrete Fourier Transform (DFT), which is invertible. This property can be leveraged to define an inverse for the STFT. In applications like audio id, one is more interested in how power is distributed among frequencies at different times, rather than applying the inverse transform. Therefore, phase is discarded by taking the modulus of each entry in $STFT(s)$.

Remark. The list of spectral representations presented above is not exhaustive. Others, that are vastly used include the Mel-frequency cepstrum coefficients (MFCCs), and chroma features - both described in [Gut06].

2.1 Audio identification methods

Having defined the spectrograms, we provide more details regarding existing audio id methods, each of them consisting of two steps: fingerprinting and querying.

In the algorithm behind Shazam described in [Wan03], starting from a spectrogram of S of a song s, a set of landmarks $\{(t_k, f_k)\}_k$ is identified. The fingerprint $\phi(s) = \{(t_k, v_{k,k'})\}$ consists of a set of points, with the first component t_k being a time-stamp and the second: $v_{k,k'} = (t_k - t_{k'}, f_k, f_{k'})$, for carefully paired indices (k, k'). Querying for a song s is described as comparing it to other songs s' and determining a possible alignment between the two - the label of s is then defined as the label of the song s' with the most likely alignment. For two songs s, s', the set of matching landmark pairs is retained: $M_{s,s'} = \{((t_k, v_{k,k'}), (t'_l, v_{l,l'})) \mid v_{k,k'} = v_{l,l'}, v_{k,k'} \in \phi(s), v_{l,l'} \in \phi(s')\}$. Then, the number of occurrences of each alignment $\{t - t' \mid ((t, v), (t', v')) \in M_{s,s'}\}$ is counted. In addition, this method is constructive- it outputs an explicit time-alignment of the snippet s in the song s'.

An approach based on image analysis techniques and locally-sensitive hashing is proposed by Baluja et al. in [BC07]. The fingerprinting step is based on [YHS05], where Yan Ke et al. treat a spectrogram as a collection of 951ms-wide separate images and compute a wavelet-like decomposition on the filters proposed by Viola and Jones in [VJ01]. In image processing, such decompositions are aimed at extracting robust summaries, or feature

detection: faces in the case of [VJ01]. By introducing min-hashing, Baluja et al. [BC07] improve on the querying step of their audio id solution. Min-hashing and locally-sensitive hashing in general, consists of aggregating several hash functions— each puts similar items in the same bins — to augment their individual discriminative powers [IM98]. That approach is particularly suitable for nearest neighbour search in settings where computing all the distances explicitly is computationally prohibitive.

A time-alignment algorithm for cover song identification is proposed by Tralie and Bendich in [TB15]. The MFCC representations of different versions of the same song are found to be related. Treating the coefficients from each of the versions as a point cloud, the authors find that one of the point cloud is a transformed (translated, scaled and rotated) version of the other. They propose to use the Smith-Waterman algorithm on a cross-similarity measure between two songs. Intuitively, if we define a vector s' representing samples from a song, by cutting a sample s into chunks and reordering them, the songs represented by s and s' will be similar in the Smith-Waterman metric. This shows that, as intended, the authors devise a metric insensitive to local reordering of parts of a song. In the context of songs, this can be thought of as a different structure (intro, chorus, solo etc.).

Chapter 3

Cubical Complexes, Homology and Persistent Homology

Topological characteristics of a metric space are invariant with respect to a wide class of transformations like bending, stretching or rotations and are therefore suited to situations when one is interested in properties more general than its geometry. A well known example of such characteristics are homology groups, which, intuitively, represent holes of different dimensions. Disconnected components, a loop and a cavity correspond to zero-, one- and two- dimensional holes respectively. More formally, to a topological space, we associate a series of groups, whose generators are those holes, which are illustrated by an example in fig. 3.1. In general, calculating homology groups of an arbitrary shape is not possible. In applications, the shape is often represented by a discrete structure (graph or set of points), what reduces the computations to linear algebra, making them tractable [KMM11]. A simplicial complex, which can be thought of as a triangulation of the underlying shape, is commonly used as such a discrete structure. In that case, we refer to simplicial homology. Persistent homology — an extension of the classic concept of homology — has gained recognition thanks to three factors, summarized by Otter et al. [OPT$^+$17]. It is based on the well-understood framework of computational homology, it remains computable and exhibits robustness to perturbations of the data. In applications where a point-cloud is sampled from that distribution, persistent homology extends simplicial homology. In this work, we apply persistent homology mainly to images, with pixels being interpreted as cubes, rather than triangles, giving the setting of cubical homology.

We first introduce cubical complexes and define the homology groups. Then, we will outline how homology extends to persistent homology, and provide the decomposition theorem that makes it so useful to compare shapes. Finally, we present how one encodes information from persistent homology groups via vectorization or kernel methods, making them suitable for learning algorithms.

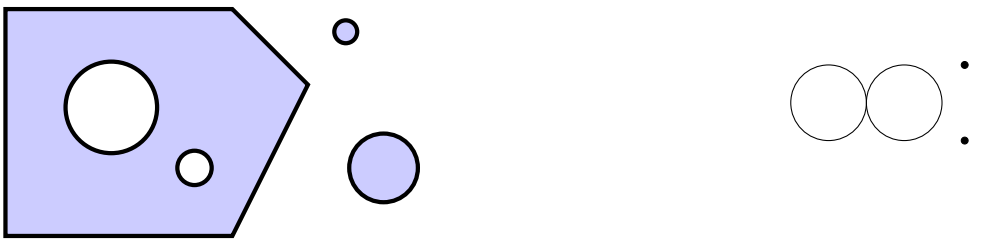

Figure 3.1: The blue shape (left) features three connected components and two one-dimensional holes. Its zero homology group H_0 has 3 generators - each corresponding to one of the connected components. Similarly, the first homology group has two generators: each can be thought of as representing one of the holes in the polygonal connected component. The black shape (right), comprised of two tangent circles and two additional points, has the same homology groups - two one-cycles (loops) and three connected components. Hence, these two shapes are homologically indistinguishable.

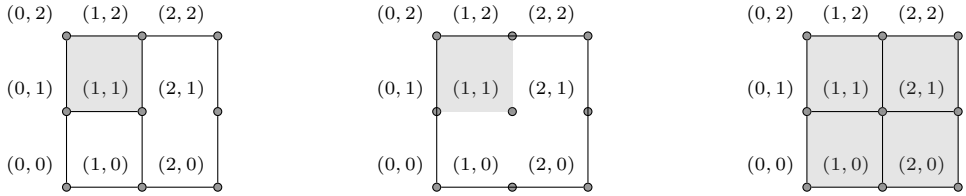

Figure 3.2: On the left, a cubical complex of dimension two with embedding dimension two, composed of nine zero-cubes, sixteen one-cubes and one two-cube. Even though the cubes have different dimensions, they all have embedding dimension two, since they are included in the plane. The faces of the two-cube $[0,1] \times [1,2]$ are $[0,0] \times [1,2], [1,1] \times [1,2], [0,1] \times [1,1], [0,1] \times [2,2]$. If we remove any of them, for example $[0,1] \times [1,1]$ and $[0,0] \times [1,2]$ as it is shown in the centre, the resulting collection of cubes is no longer a cubical complex. On the right, the maximal cubical complex $Q(X)$ on the given set of vertices $X = \{(i,j) \mid i,j = 0,1,2\}$.

3.1 Cubical complexes

In [KMM11, Chapter 2], cubes - the elementary building blocks for cubical complexes are defined as products of intervals.

Definition 2. A closed interval $I = [a,b] \subset \mathbb{R}$ is called elementary if $b \in \{a, a+1\} \subset \mathbb{Z}$. In the first case $a = b$, the interval is called degenerate. An elementary cube $Q \subset \mathbb{R}^d$ is a product of d elementary intervals.

Let $I_1, \ldots, I_d$ be elementary intervals. For a cube $Q = I_1 \times \ldots \times I_d$, we say that

- d is the embedding number of Q,
- I_k is the k-th component of Q,
- $\dim(Q) = |\{l \mid I_l \text{ is not degenerate}\}|$ is the dimension of Q,
- Q is called a vertex if $\dim(Q) = 0$.

A cube with an embedding number d and at least one non-degenerate component may contain a cube of smaller dimension - similarly to an elementary interval containing its two endpoints. This is made rigorous by the notion of faces.

Definition 3. A cube Q_2 is said to be a face of a cube Q if $Q_2 \subset Q$. In addition, if $\dim(Q_2) = \dim(Q) - 1$, we call Q_2 a proper face of Q. We denote by $\mathcal{F}(Q)$ the set of proper faces of Q.

Cubical complexes — particular collection of cubes, are defined below and exemplified in fig. 3.2.

Definition 4. Let K be a collection of cubes of the same embedding dimension. Then, K is a cubical complex if

- for any cube $Q \in K$, its faces are also in K,
- for all cubes $Q_1, Q_2 \in K$, the intersection $Q_1 \cap Q_2 \in K$ is either empty or a face of Q_1 and Q_2,

Similarly as for cubes, we can speak of the dimension of a cubical complex, which is defined as the maximal dimension of a cube in the complex and of its embedding dimension - the embedding dimension of the cubes respectively.

Remark. The definition of elementary cubes is in general restrictive, due to requirements on the coordinates being integers and the elementary intervals all of length one. Although it is possible to lift that restriction and allow coordinates that are in $\mathbb{R}$, we prefer to adopt the approach of mapping the endpoints of the intervals to integers, which simplifies the definition of faces.

For a finite collection of vertices X, we denote by $Q(X)$ the maximal cubical complex on those vertices. We add edges and cubes of higher dimensions, starting from the lower dimensions, until we cannot add any more without violating the constraints imposed in the definition of a cubical complex. Such a complex is shown on the right in fig. 3.2.

Figure 3.3: On the left, the dashed path is the geometric representation of the boundary of the three shaded two-cubes. On the right, the boundary of a path is marked by empty vertices .

3.2 Homology of Cubical complexes

Fix the maximal complex generated on $X_{M,N} = \{(i,j) \mid 0 \leq i \leq M, 0 \leq j \leq N\}$ a finite, two-dimensional Cartesian grid. and consider a cubical sub-complex K. Although homology as a concept reflects the intuitive and geometric notion of holes in a space (example in fig. 3.2), it is made rigorous by considering boundaries of building blocks of a shape: cubes in the case of cubical complexes. For example, a one-dimensional hole is nothing else than a cycle - a linear combination of intervals, which is not a boundary of a linear combination of squares. This is made rigorous by homology groups, where we first identify cycles and discard those that can be expressed as a boundary of other cubes.

Homology groups are algebraic objects, which we introduce by presenting the work by Kaczynski et al. in [KMM11]. After the linear combinations of cubes, we define the boundary of cube and finally, homology groups. Example 1 shows that the definition coincides with the intuition conveyed above. Unless otherwise mentioned, we identify a cube $Q \in X_{M,N}$ with an algebraic object, denoted by $\hat{Q}$ in [KMM11], and vice-versa.

Definition 5. Let $\mathbb{F}$ be a field. We call k-chains $C_k(K)$ the vector space on $\mathbb{F}$, generated by elementary cubes of dimension k, namely

$$C_k(K) = \left\{ \sum_{Q \ k-\text{cube}} a_Q \widehat{Q} \mid a_Q \in \mathbb{F} \right\}.$$

The notion of holes of various dimensions is based on the idea of boundary - for example, a cycle is a one-dimensional hole that is not a boundary of a two-dimensional shape. The boundary is made rigorous and then exemplified below.

Definition 6. Let Q be a cube of dimension k and embedding dimension d, that we decompose as a product of two cubes $Q = \prod_{l=1}^{d} I_l = [a_1, b_1] \times Q'$. Then, the boundary of $\widehat{Q}$ is defined recursively as the $(k-1)$-chain equal to the alternate sum of its proper faces

$$\partial_k \left(\widehat{Q} \right) = \partial_{k_1} \left(\widehat{[a_1, b_1]} \right) \times \widehat{Q'} + (-1)^{k_1} \widehat{[a_1, b_1]} \times \partial_{k_2} \left(\widehat{Q'} \right),$$

where k_1, k_2 are the dimensions of $[a_1, b_1]$ and Q' respectively, while

$$\partial_1 \left(\widehat{[a,b]} \right) = \hat{b} - \hat{a}, \qquad \partial_0 \left(\widehat{[a,a]} \right) = 0.$$

Remark. For the boundary to be well defined, the decomposition of a cube into intervals must exist, what is given by the definition of a cube. The uniqueness of such a decomposition is shown in [KMM11].

The boundary operators reflect the intuitive notion of a boundary of a cube, which is a linear combination of its proper faces. For example, as shown in fig. 3.3, for an interval (a one-cube), its boundary is comprised of its endpoints (zero-cubes), while for a square (a two-cube), it corresponds to its four sides (one-cubes). The boundary is defined on elementary cubes and extended to the space of chains by linearity in $\mathbb{F}$. With the convention $C_{-1}(K) = 0$, it yields the chain complex (3.1), with the property given by lemma 1. It implies that $\mathrm{Im}(\partial_{k+1}) \subset \mathrm{Ker}(\partial_k)$ and leads to the definition of homology groups.

Remark. Note that Q' is no longer embedded in dimension d, but in $d-1$. Therefore, to be more precise, we would need to define the boundary operator recursively not only in the dimension k of a cube, but also in the embedding dimension d. In addition, the Cartesian product of two algebraic cubes refers to the algebraic object of the cartesian product of their geometric representatives $\widehat{Q} \times \widehat{Q'} = \widehat{Q \times Q'}$.

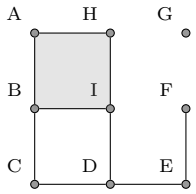

Figure 3.4: The cubical complex K, with vertices on the grid X. It has two connected components and features two one-cycles: one on vertices A, B, I, H and the other on B, C, D, I, but the former is 'filled' by the 2-cube $[ABHI]$. Example 1 shows how this can be retrieved through the definitions and rigorous calculations.

$$\ldots \xrightarrow{\partial_{k+2}} C_{k+1}(K) \xrightarrow{\partial_{k+1}} C_k(K) \xrightarrow{\partial_k} C_{k-1}(K) \xrightarrow{\partial_{k-1}} \ldots. \tag{3.1}$$

Lemma 1. *The composition of two operators is trivial, that is*

$$\partial_k \circ \partial_{k+1} = 0.$$

Proof. We proceed by induction. For $k = 0$, the result follows immediately as $\partial_0 = 0$. To prove the result for k, let us consider a $(k+1)$-cube $Q = [a, b] \times Q'$. Then,

$$\begin{aligned}\partial_{k+1}(Q) &= \partial_{k_1}([a,b]) \times Q' + (-1)^{k_1}[a,b] \times \partial_{k_2}(Q')) \\ &= \begin{cases} [b,b] \times Q' - [a,a] \times Q' - [a,b] \times \partial_k(Q') & \text{if } k_1 = 1, \text{ ie } a \neq b, \\ [a,a] \times \partial_{k+1}(Q') & \text{if } k_1 = 0. \end{cases}\end{aligned}$$

Examining the case $k_1 = 0$, it is clear that the boundary operator will 'commute' with the product, until it encounters a non-trivial first interval. Therefore, we can assume that $k_1 = 1$ and hence $k_2 = k$ as $k_1 + k_2 = k+1$. By using this observation about commutativity on the image of the second boundary operator,

$$\begin{aligned}\partial_k \circ \partial_{k+1}(Q) &= [b,b] \times \partial_k(Q') - [a,a] \times \partial_k(Q') - (\partial_1([a,b]) \times \partial_k(Q') - [a,b] \times \partial_{k-1} \circ \partial_k(Q')) \\ &= [a,b] \times \partial_{k-1} \circ \partial_k(Q') \\ &= 0,\end{aligned}$$

where the induction hypothesis yields the last equality. □

Definition 7. The k-th homology group of K is defined as

$$H_k(K) = \operatorname{Ker}(\partial_k) / \operatorname{Im}(\partial_{k+1}).$$

Example 1 (Calculation of homology groups). Consider a grid-shaped, finite collection of points $X \subset \mathbb{Z}^2$, that is

$$X = \{(i, j) \in \mathbb{Z}^2 \mid 0 \leq i \leq 3,\ 0 \leq j \leq 3\},$$

and the cubical complex K from 3.4, whose vertices $\operatorname{Vert}(K) \subset X$ are included in that grid. For clarity, we denote the vertices with letters $A, \ldots, I$ instead of their coordinates, and label cubes of higher dimensions (intervals and squares) by the vertices they share. The spaces of chains are generated by

$$\begin{aligned}C_0(K) :&\{\widehat{[A]}, \widehat{[B]}, \ldots, \widehat{[I]}\}, \\ C_1(K) :&\{\widehat{[AB]}, \widehat{[BC]}, \widehat{[HI]}, \widehat{[ID]}, \widehat{[FE]}\} \cup \{\widehat{[AH]}, \widehat{[BI]}, \widehat{[CD]}, \widehat{[DE]}\}, \\ C_2(K) :&\{\widehat{[ABIH]}\}.\end{aligned}$$

The boundary matrices are

$$\partial_1 = \begin{array}{c} \begin{array}{ccccccccc} AB & BC & HI & ID & FE & AH & BI & CD & DE \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array}, \quad \partial_2 = \begin{array}{c} \begin{array}{c} ABIH \end{array} \\ \begin{pmatrix} -1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \end{array} \begin{array}{c} AB \\ BC \\ HI \\ ID \\ FE \\ AH \\ BI \\ CD \\ DE \end{array} \tag{3.2}$$

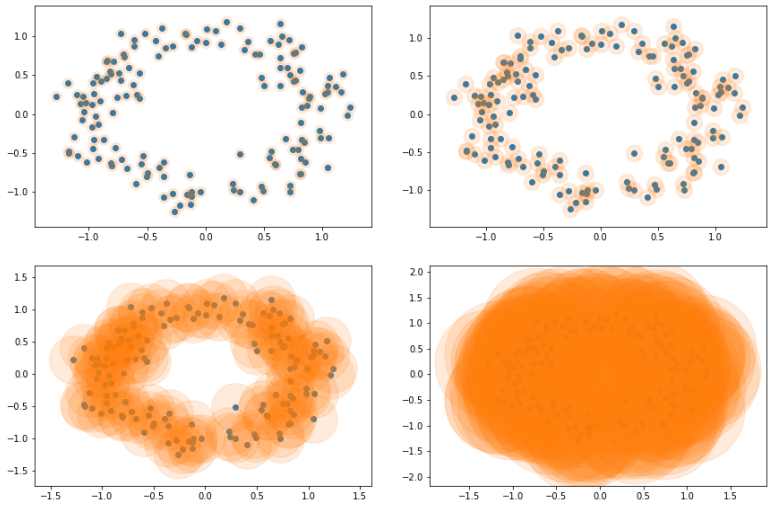

Figure 3.5: A noise point cloud, sampled from a circle. To use homology to characterise the support of the initial distribution, one examines the space defined by the union of balls of radius r centred at the sampled points. However, the homology of the resulting space depends on r - for small values, the space is disconnected (top row). After an increase of the parameter, as shown on the bottom left image, the resulting space is connected and has the homology of a circle. When r is bigger (bottom right), the space hole in the middle gets covered by the balls and the homology of the space is the same as that of a point. Choosing 'the right value' for r is in general not obvious— such a value may not even exist.

Now, we can determine the image and kernels of each of those matrices, by performing elementary operations to reduce the matrices. Finally, we can read the homology groups

$$H_0(K) = \mathbb{F}[\widehat{A}] \oplus \mathbb{F}[\widehat{G}],$$
$$H_1(K) = \mathbb{F}\left([\widehat{BC}] + [\widehat{CD}] - [\widehat{ID}] - [\widehat{BI}]\right),$$

where $\oplus$ is the direct sum.

Remark. In the introduction, we have mentioned that another type of complexes (simplicial) is often used. One can, for each cube, define a triangulation and therefore, get a different combinatorial structure, but with the same homology groups.

3.3 Persistent Homology of cubical complexes

Persistent homology is an extension of homology: instead of looking at the homology of a fixed topological space, represented by a complex, we let the complex 'grow'. The motivation for this extension comes from problems of homological inference. Starting from a point-cloud, one would build a simplicial complex and study its homology. However, the results of such a procedure highly depend on the scale parameter chosen to construct that complex, as depicted in fig. 3.5. A way of building many complexes, and studying how the homology changes as the parameter varies was proposed by Edelsbrunner et al. [ELZ02]. This is made rigorous thanks to the definition of a filtration — a growing family of spaces, where each space corresponds to a complex at a given scale — and shown in fig.3.6.

Definition 8. A filtration of a cubical complex K is a collection of growing cubical complexes $(K_r)_{r\in R}$. That is, indexed by an order set $(R, \leq)$, it satisfies

$$r_1 \leq r_2 \implies K_{r_1} \subseteq K_{r_2}.$$

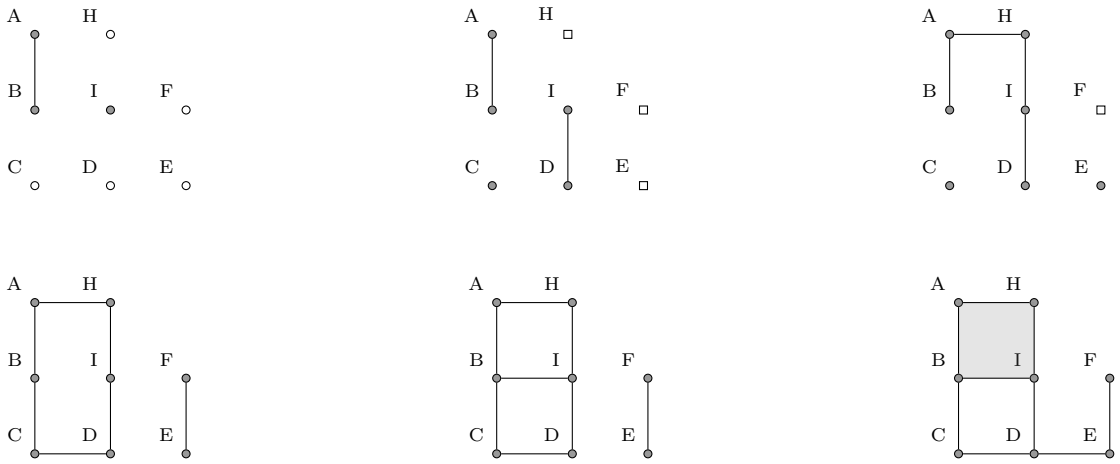

Figure 3.6: Part of a filtration of the cubical complex in fig. 3.4. It is read from left to right and top to bottom.

Two observations are crucial for the definition of persistent homology. First, for each $r \in R$, K_r is a cubical complex and therefore we can compute its homology groups $H_k(K_r)$. Second, the inclusions $\iota_{r_1,r_2} : K_{r_1} \to K_{r_2}$ induce morphisms between homology groups $\left(\iota^*_{r_1,r_2}\right)_{r_1 \le r_2 \in R}$, where $\iota^*_{r_1,r_2} : H_k(K_{r_1}) \to H_k(K_{r_1})$. Indeed, the morphism are first induced between modules $\iota_{r_1,r_2} : C_k(K_{r_1}) \to C_k(K_{r_2})$, for all k and we notice that they commute with boundary operators (3.3), what shows that they are well-defined on homology groups as well.

$$\begin{array}{ccc} C_k(K_{r_1}) & \xrightarrow{\partial_k} & C_{k-1}(K_{r_1}) \\ \downarrow{\scriptstyle \iota^*_{r_1,r_2}} & & \downarrow{\scriptstyle \iota^*_{r_1,r_2}} \\ C_k(K_{r_2}) & \xrightarrow[\partial_k]{} & C_{k-1}(K_{r_2}) \end{array} \tag{3.3}$$

Definition 9. The k-th persistent homology group of a filtration $(K_r)_{r\in R}$ is the collection of homology groups $(H_k(K_r))_{r\in R}$ and morphisms $\left(\iota^*_{r_1,r_2} : H_k(K_{r_1}) \to H_k(K_{r_1})\right)_{r_1 \le r_2 \in R}$ induced by the inclusions $\iota_{r_1,r_2} : K_{r_1} \to K_{r_2}$.

In applications, the cubical complexes are often finite [OPT+17]. Therefore, the filtration can be assumed to be of finite length, and the set R can be thought of as a finite subset of $\mathbb{N}$. In particular, the persistent homology groups are finitely-generated. A persistent homology group can be then seen as a persistence-module [CdSGO12], which has a (unique) decomposition that describes the persistence of the generators [Zom05, Section 3.2]

$$H_k((K_r)_{r\in R}) = \bigoplus_{i=1}^{N} e_{(b_i,d_i)}(r), \tag{3.4}$$

where $b_i, d_i \in R \cup \{\pm\infty\}$ and

$$e_{(b_i,d_i)}(r) = \begin{cases} \mathbb{F}, & b_i \le r < d_i, \\ 0, & \text{otherwise.} \end{cases}$$

The decomposition (3.4) is shown to be unique, up to ordering of intervals [CdSGO12, Theorem 1.3] and provides an interpretation of persistent homology groups as comprised of generators that appear at a certain time, persist and then vanish. To be more precise, for a generator $e_{(b,d)}$, we say that it is born at b, and dies at d. We call its persistence interval the range of values of r for which the space spanned by $e_{(b_i,d_i)}(r)$ is of dimension one - that is $[b, d]$. The length of this interval is called the persistence of the generator.

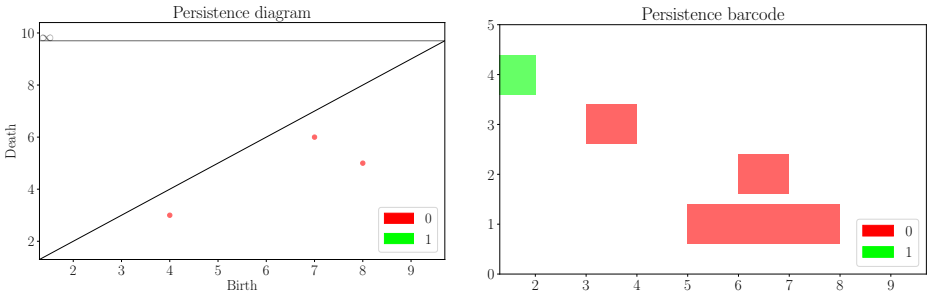

Figure 3.7: The persistence diagram and barcode corresponding to the persistent homology groups filtration in fig. 3.6. The zero and first persistence diagrams are $D^0 = \{(0,\infty),(0,2),(1,3),(2,5)\}$ and $D^1 = \{(3,\infty),(4,5)\}$.

The decomposition (3.4) has also a geometric interpretation. As discussed in 3.2, homology groups represent holes. Intuitively, a generator with birth b and death value d indicates that a cube generating a hole appears in the filtration at $r = b$, and that at $r = d$, this hole is filled as a cube of higher dimension appears in the complex. This leads to the interpretation that the geometric holes exists only through the interval $[b, d]$.

Finally, as the decomposition (3.4) characterizes the persistent homology groups, it provides graphical representations shown in fig. 3.7: the barcode and the persistence diagram [CSEH07, ELZ02], the latter defined in (3.5) as the multi-set

$$D^k = \{(b_i, d_i)\}_{i=1}^N \in R^2. \tag{3.5}$$

3.4 Distance on the space of Persistent Diagrams

One motivation for introduction of homology groups is the classification of spaces. In other words, comparing them, up to isomorphism of the homology groups. For the case of persistent homology, one could adopt a similar approach, looking at isomorphism classes of persistence homology groups - for $((A_r)_{r\in R}, (\iota^{A,*}_{r_1,r_2})_{r_1\le r_2\in S})$, $((B_r)_r, (\iota^{B,*}_{r_1,r_2})_{r_1\le r_2\in R})$, we say they are isomorphic, if there exists a set of isomorphisms $\phi_r : A_r \to B_r$ that commute with the filtration-induced morphisms

$$\phi_{r_2} \circ \iota^{A,*}_{r,r_2} = \iota^{B,*}_{r,r_2} \circ \phi_r, \qquad \iota^{A,*}_{r,r_2} \circ \phi_r^{-1} = \phi_{r_2}^{-1} \circ \iota^{B,*}_{r,r_2}. \tag{3.6}$$

One of the motivations for introducing persistent homology groups was to capture similarity in shapes, underlying a noisy sample. However, an isomorphism of persistent homology groups (3.6) is too restrictive in this context. In particular, a slight change in the birth or death of one of the generators would mean that two groups are no longer isomorphic. An example where a slightly perturbed dataset gives different persistent homology groups, but which we would still like to classify as similar, is shown in fig. 3.8.

Therefore, more useful than isomorphism classes is a distance on the space $\mathcal{D}$ of finite persistence diagrams with filtrations with parameter values in R. To be more precise, each element D of $\mathcal{D}$ is a multi-set of points: a set, where elements are counted with multiplicity. For completeness, as described in [MMH11, Section 2.4], assume that elements of $\mathcal{D}$ are the so-called generalized persistence diagrams - that is, they contain the diagonal $\Delta = \{(b, b) \mid b \in R\}$ with infinite multiplicity: for a persistence diagram D of a cubical complex, the generalized persistence diagram is $\bar{D} = D \cup \Delta$. The p-th Wasserstein distance on $\mathcal{D}$ is defined as

$$W_p(D_1, D_2) = \left(\inf_{\gamma} \sum_{x\in\bar{D}_1} \|x - \gamma(x)\|_\infty^p \right)^{1/p}, \tag{3.7}$$

where the infimum is taken over all bijections $\gamma : \bar{D}_1 \to \bar{D}_2$ between generalized persistence diagrams of D_1 and D_2.

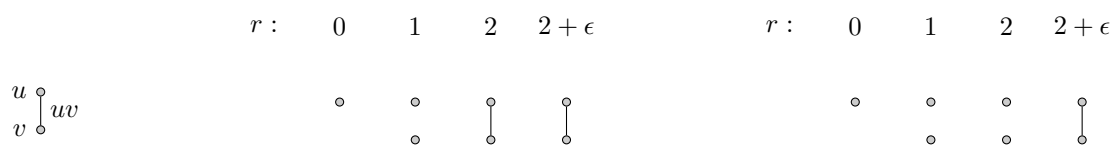

Figure 3.8: Two different filtrations on the complex $K = \{u, v, uv\}$, for $R = \{0, 1, 2, 2+\epsilon\} \subset \mathbb{R}$. As only their zero persistent homology groups have non-trivial elements, we compare only the corresponding persistence diagrams, which are $D_1^0 = \{(0, \infty), (1, 2)\}$ and $D_1^0 = \{(0, \infty), (1, 2+\epsilon)\}$ for the filtrations on the left and right. For any $\epsilon > 0$, as the persistence diagrams are distinct, the persistent homology groups are not isomorphic. However, given that the diagrams are close, we would like to say that for ϵ small, they are similar, what is quantified by the wasserstein (3.7) and bottleneck (3.8) distances.

Remark. The distance is well defined for PDs with a finite number of off-diagonal points - the trivial bijection that sends all off-diagonal points from $\bar{D}_1$ to the diagonal in $\bar{D}_2$ and whose inverse maps points from $\bar{D}_2$ to the diagonal, gives an upper bound and guarantees that the set of bijections is not empty.

The special case when $p = \infty$ is called the bottleneck distance

$$d_B(D_1, D_2) = \left(\inf_{\gamma} \max_{x \in \bar{D}_1} \|x - \gamma(x)\|_\infty \right). \tag{3.8}$$

It has been of particular importance for applications as it provides a stability guarantee (theorem 2), for filtrations that are similar. The notion of similarity of filtrations requires the introduction of filter functions. For a filtration $(K_r)_{r \in R}$, we define the corresponding function f on K by $f(Q) = \inf\{r \in R \mid Q \in K_r\}$. Conversely, for an appropriate function $f : K \to R$, we can define a sequence of complexes

$$K_r = f^{-1}(]-\infty, r]). \tag{3.9}$$

Example 2. The filter functions f, g that define the two filtrations in fig. 3.8 are defined element-wise by

$$f(u) = 0 = g(u), \quad f(v) = 1 = g(v), \quad f(uv) = 2 \neq 2 + \epsilon = g(uv).$$

Hence, $\|f - g\|_\infty = \epsilon$.

Remark. For the sequence of spaces defined by (3.9) to be a filtration, it is necessary that the value of f on the faces of a cube are smaller (or equal) to the value of that cube $Q \subset Q' \implies f(Q) \leq f(Q')$.

Theorem 2 (Bottleneck stability [CSEH07])**.** *Let $f, g : X \to R$ be two continuous functions on a topological space X. Let D_f^k denote the persistence diagram of the k-th persistent homology group induced by the filtration of f (3.9). Then, the distance between D_f, D_g is bounded by the distance between f and g in the sup-norm*

$$d_B(D_f, D_g) \leq \|f - g\|_\infty. \tag{3.10}$$

This theorem is valid in the combinatorial setting of simplicial or cubical complexes, justifying the use of persistent homology in data analysis. In applications, due to noise, perturbations of the filtration (through the filter function) as shown in fig.3.8 are to be expected. The theorem 2 guarantees that if the noise remains small, the resulting filtration yields a persistent diagram close to that from unperturbed data.

Remark. Due to the ∞-norm, only the maximal discrepancies in the persistence diagrams and the filtration functions are considered. Both sides of the inequality (3.10) may be impacted by a change in the value of filter function on one cube only.

Remark. The bottleneck distance has also a more algebraic interpretation. Theorem 4.11 [CdSGO12] states that the map that associates to a persistence module its Persistence Diagram is an isometry.

3.5 Vectorization methods for Persistent Homology

For applications that benefit from learning algorithms, like for example fingerprinting, the space of persistence diagrams $\mathcal{D}$ has some drawbacks. Such settings require embedding $\mathcal{D}$ in a Hilbert space, either via an explicit

vectorization or a kernel. The space $\mathcal{D}$ lacks the structure of a vector space and neither the Wasserstein nor the bottleneck distance provide appropriate embeddings [CCO17]. Several methods to represent persistence homology groups(some of which are solutions to the above problem) have been proposed. Some feature explicit vector representations - such as Persistence Images [AEK+17] or Betti Curves [SGB15], while others do not, for example the Persistence Fisher Kernel [LY18] or the Sliced Wasserstein Distance [CCO17].

These methods map a persistence diagram D to a complete vector space $(V, \|\cdot\|)$, via a feature map that we will denote $\phi : \mathcal{D} \to V$. Based on [Loz18], we recall a few common representations, their properties, and we aim to compare them for the present applications, depending on the filtration.

3.5.1 Betti Curves

Betti numbers of a space are the dimensions of the homology groups. To be more precise, the k Betti number associated to a cubical complex K is $\beta_k = \dim(H_k(K))$. Persistent homology provides a natural extension to Betti curves, which tracks the evolution of Betti numbers as a function of the persistence parameter. For a persistent diagram D, the corresponding Betti curve $\phi_B(D) \in L^p(R)$ is

$$\begin{array}{rccc} \phi_B(D): & R & \to & \mathbb{Z} \\ & x & \mapsto & \sum\limits_{(b,d)\in\mathcal{D}} 1_{[b,d[}(x), \end{array}$$

where $1_{[b,d[}(x) = 1$ if $b \leq x < d$ and 0 otherwise, is the indicator function of the interval. Please note that even though the cardinality $|D|$ of a generalized persistence diagram is not finite, the sum is finite and bounded by the number of points with non-zero persistence. The distance between two persistence diagrams in the feature space can then be defined as

$$d_{\phi_B}(D_1, D_2) = \|\phi_B(D_1)(x) - \phi_B(D_2)(x)\|_{L^p(R)}.$$

While one possibility is to store the values of the function at each point where the PH changes, it amounts to storing the Persistence Diagram. One reduces the memory-footprint of ϕ_B and the computational effort for d_{ϕ_B} by, for example, discarding points of small persistence, or using a discretization. This gives two approximations of ϕ_B

$$\widehat{\phi}_B(D) = (\phi_B(D)(x_n))_{n=1}^{N_r}, \tag{3.11}$$

$$\widetilde{\phi}_B(D) = \phi(\psi_{N_s}(D))), \tag{3.12}$$

where $(x_n)_{n=1}^{N_r}$ is a fixed discretization of R and ψ_N returns the subset of the N most persistent points (completing with points on the diagonal at the end, in case there are less than N_s points initially). Both are extended on R by linear interpolation.

In light of this extension, one could choose to keep the same metric. For $d_{\widehat{\phi_B}}$, the integration is straightforward, while $d_{\widetilde{\phi_B}}$ requires, for each pair of Persistence Diagrams, for example, interpolating the curves over the union of all points where persistence changes. All in all, these approximations have a memory footprint and comparison time which are independent from the number of points. However, the initial computations depend on the number of points - for a persistence diagram with N points, $\widehat{\phi}_B$ and $\widetilde{\phi}_B$ are $\mathcal{O}(N_r N)$ and $\mathcal{O}(N + N_r \log N_r)$.

Remark. Carriere et al. [CCO17] make a practical remark on how to compute the Wasserstein distance between Dirac delta distributions on the real line. That method relies on ordering the centres of the distributions and can be used to compute the distances between features defined by ϕ_B or $\widetilde{\phi}_B$ when using $p = 1$, by, in a similar fashion, ordering separately the births and deaths.

The map is stable on the subspace of persistence diagrams of a bounded cardinality. By considering $\mathcal{D}_C$ — the subset of persistence diagrams that have at most C points (counting with multiplicity) that do not lie on the diagonal, we can related the distance d_{ϕ_B} between Betti curves to the bottleneck distance d_B via theorem 3.

Theorem 3. *Let D_1, D_2 be two Persistence Diagrams. Then, the map $\phi_B : \mathcal{D}_C \to L^p(R)$ is Lipschitz*

$$d_{\phi_B}(D_1, D_2) \leq 2C d_b(D_1, D_2).$$

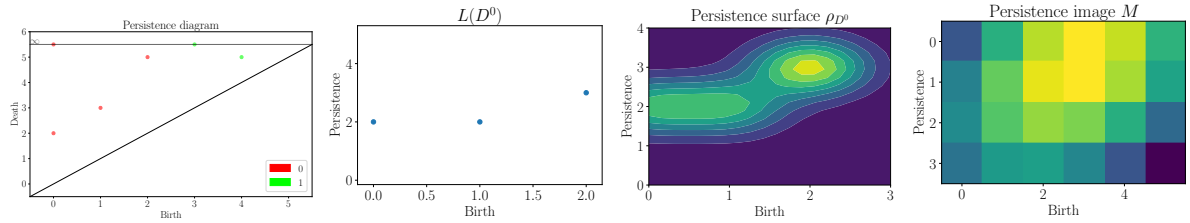

Figure 3.9: The persistence image of D^0 of the filtration in 3.6. On the left, the persistence diagram $L(D^0)$ and to its right, the diagram in the transformed axis. The third figure from the right represents the contour of the persistence surface, on a 20×20 discretization of $[0,3] \times [0,4]$. We can see that the most persistent point $(2,5) \in D^0$ is given more weight than the less persistent points. On the right, the persistence image, with resolution 6×4.

Proof. Let $\gamma : PD_1 \to PD_2$ be a matching. Let us consider $(b,d) \in PD_1$ and let $(b',d') = \gamma(b,d)$. Then, $\|1_{[b,d[} - 1_{[b',d'[}\|_{L^p} = |b-b'| + |d-d'| \leq 2\max\{|b-b'|, |d-d'|\}$. Hence, by triangle inequality

$$\left\| \sum_{(b,d)\in PD} 1_{[b,d[} - 1_{[\gamma(b,d)[} \right\|_{L^p} \leq \sum_{(b,d)\in PD} \left\| 1_{[b,d[} - 1_{[\gamma(b,d)[} \right\|_{L^p} \leq 2C d_b(PD_1, PD_2).$$

□

However, the approximation $\hat{\phi}_B$ does not exhibit stability properties. In contrast, for $N_r \geq C$, $\tilde{\phi}_B = \phi_B$ is stable.

3.5.2 Persistence Images

A persistence surface can be though of as a smoothed version of a persistence diagrams interpreted as a sum of Dirac measures on the plane. A persistence image is a representation of a Persistence Surface in $\mathbb{R}^{n_1 \times n_2}$. There are three steps that lead us from a persistence diagram to a persistence image

1. a change of coordinates (3.13) - from the birth-death axes to birth-persistence,
2. convolution with a smoothing kernel (3.14),
3. discretization (3.15).

The transformation is defined to obtain a rectangular- instead of a triangular region, mapping the death-axis to the persistence-axis

$$\begin{array}{rccc} L: & \{(x,y) \mid y \geq x, x \geq 0\} & \to & \mathbb{R}^2_+ \\ & (x,y) & \mapsto & (x, y-x). \end{array} \tag{3.13}$$

The next step consists of obtaining the Persistence Surface - a smoother version of PD

$$\rho_D : q \mapsto \sum_{p \in L(D)} f(p) k(q,p), \tag{3.14}$$

where $k(q,p) = k_\sigma(q,p) = \exp\left(-\frac{\|q-p\|}{\sigma}\right)$. Finally, fix the discretizations $x_1, \ldots, x_{n_1}$ and $y_1, \ldots, y_{n_2}$ of both axes. For each square $[x_i, x_{i+1}[\times [y_j, y_{j+1}[$, the value of the persistence image on that cell is encode in the element (i,j)

$$M_{i,j} = \int_{[x_i, x_{i+1}] \times [y_j, y_{j+1}]} \rho_{PD}(q) \mathrm{d}q. \tag{3.15}$$

Remark. The persistence image M is seen as a matrix. For machine-learning applications like [ZZJS16], it is often useful to think of it as a column-vector. If that is the case, a bijection $\{1, \ldots, n_1\} \times \{1, \ldots, n_2\} \to \{1, \ldots, n_1 n_2\}$ needs to be defined.

The space of persistence images, defined as the image of the space of persistence diagrams, can be seen as a subset of $\mathbb{R}^{n_1 \times n_2}$. Even though it is not a vector space, it is embedded in one, and it is complete as a subspace, with respect to the usual metrics on finite-dimensional real vector-spaces. If f is zero only on the diagonal, the map $D \mapsto L(D)$ is injective [Loz18]. However, it is not necessarily surjective. In particular, a statistic of a set of Persistence Images, may not have a persistence diagram representation.

In practice, as shown in the implementation [Dlo17], Persistence Images are not calculated exactly - only an approximation is obtained. Authors of [AEK$^+$17] have exhibited stability properties of Persistence Surfaces and Images. However, in case the computational procedure described above is applied for a given discretization, they no longer hold. The computational complexity for calculating the persistence image of size $n_1 \times n_2$, for a diagram with N points, is $\mathcal{O}(n_1 n_2 N)$.

Both of these problems- instability and lack of injectivity, can be mitigated using a finer discretization, which, however, involves a higher computational cost. A remedy for the hard assignments (but not the injectivity) could be using soft pixel- assignments, which we will describe in chap. 4.2.

Remark. Especially for images of high dimensions, or long persistence scales, it is often the case that, due to the uniform discretization of the axes, most pixels have small values. One could try to use dimensionality reduction techniques, or identify pixels that carry discriminatory information, as it was done in [OHK18].

Choice of parameters for Persistence Images

The map $D \mapsto M$ implies the choice of a weighting function, the kernel k, and a discretization. While Adams et al. [AEK$^+$17] claim that this representation is not too sensitive to the last two choices, we will reflect on our experience in chap. 4.2.

It is believed that points with small persistence are attributed to noise (a discussion is provided by Stolz et al [SHP17]), and, in general, occur in greater numbers. In addition, the points on the diagonal should be completely discarded. On the other hand, examples where the information carried by points of small persistence is important exist. In [SHP17], Stolz et al. show that persistence is not enough to distinguish between the Kuramoto- and null-models: the birth-time proves equally important in that particular example. A case relying directly on Persistent Images appears in [ZZJS16], where the authors reflect on the importance of features with low persistence. In a probabilistic setting for specific filtrations on simplicial complexes, the authors [DP18] offer a stability- and convergence-motivated approach to choosing weighting functions systematically. To summarize, f should be continuously differentiable, its partial derivative with respect to persistence $\partial_y f(u, v) > 0$ for small v and the points with no persistence should be discarded $f(u, 0) = 0$. The proposed weighting function is also bounded

$$f(x, y) = \begin{cases} 0, & y \leq 0, \\ y/b, & 0 < y \leq b, \\ 1, & b < y, \end{cases}$$

where b is the persistence of the most persistent feature from all the trials.

It has been shown in [AEK$^+$17, Figure 2] that the resolution does not impact the classification accuracy. To be more precise, the authors show that, regardless whether one takes images of size 5×5 or 100×100, the classification results are similar. However, to the best of our knowledge, the exact choice of the discretization (range and width of pixels) is not described.

The kernel is often a Gaussian function that reduces the problem to fixing the standard deviation σ of the distribution. Similarly to resolution choices, it has been shown of limited importance [AEK$^+$17], but the authors leave it as an 'open problem'.

3.5.3 The Sliced-Wasserstein kernel

We first recall the definition of a kernel [Loz18] and show how to generate a positive definite one. This will motivate the introduction of the Sliced Wasserstein distance [CCO17] that follows.

Definition 10. A function $\kappa : V \times V \to \mathbb{R}$ is said to be a kernel on V if there exists a Hilbert space $\mathcal{H}$ and a map $\phi : V \to \mathcal{H}$, such that

$$\kappa(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}.$$

The space $\mathcal{H}$ and the map ϕ are called the feature space and map respectively.

In general, given a distance function $f : V \times V \to \mathbb{R}$, one can define a kernel by choosing a bandwidth σ and composing with an exponential function

$$\kappa_{f,\sigma}(x, y) = \exp\left(\frac{-f(x, y)}{2\sigma^2}\right).$$

Then, as stated in [CCO17, Section 2.2], the kernel $\kappa_{f,\sigma}$ is positive definite if and only if f is conditionally negative, that is

$$\sum_k a_k = 0 \implies \sum_{k,l} a_k a_l f(x_k, x_l) \leq 0, \qquad \forall \{a_k\}_{k=1}^N \subset \mathbb{R},\ \{x_k\}_{k=1}^N \subset V.$$

Consider $V = \mathcal{D}$, despite its popularity, the bottleneck distance d_b is not conditionally negative [CCO17], and nor is the Wasserstein distance [LY18]. An alternative is provided by a one-dimensional approximation - the Sliced Wasserstein distance.

For $\theta \in [\frac{\pi}{2}, -\frac{\pi}{2}]$, let v_θ denote the unit vector in direction θ and consider the projection operator $\pi_\theta : x \mapsto \langle x, \theta\rangle v_\theta$ as well as the measure induced by PD on the real-line

$$\mu^\theta = \sum_{p \in PD} \delta_{\pi_\theta(p)}, \qquad \mu^\theta_\Delta = \sum_{p \in PD} \delta_{\pi_\theta \circ \pi_\Delta(p)}.$$

Definition 11. The Sliced-Wasserstein distance between diagrams PD_1, PD_2 is defined as

$$SWD(PD_1, PD_2) = \frac{1}{2\pi} \int_{\mathbb{S}_1} \mathcal{W}(\mu_1^\theta + \mu_{2,\Delta}^\theta, \mu_2^\theta + \mu_{1,\Delta}^\theta) \mathrm{d}\theta. \tag{3.16}$$

Computing the Wasserstein distance is, in general, intractable- apart from special cases, like Gaussian distributions. For two measures which are sums of Diracs $\mu = \sum_{i=1}^N \delta_{x_i}$, $\nu = \sum_{j=1}^N \delta_{y_j}$, the Wasserstein distance reads

$$W(\mu, \nu) = \sum_{i=1}^N |x_i - y_i|,$$

where we assume that the vectors $(x_1, \ldots, x_N)$, $(y_1, \ldots, y_N)$ are ordered: $i \leq j \implies (x_i \leq x_j,\ y_i \leq y_j)$. Under mild assumptions on the alignment of points, an algorithm for exact computations of (3.16) is provided in [CCO17].

The authors also provide an approximation - it consists of taking M directions $(\theta_i)_{i=1}^M$ and averaging the Wasserstein distances calculated for the lines $L(\theta_i)$, giving

$$\widehat{SWD}(D_1, D_2) = \frac{1}{M} \sum_{i=1}^M \mathcal{W}(\mu_1^{\theta_i} + \mu_{2,\Delta}^{\theta_i}, \mu_2^{\theta_i} + \mu_{1,\Delta}^{\theta_i}).$$

In particular, this approach does not compromise stability

The Sliced-Wasserstein kernel is then $k_{SWD,\sigma}$, for σ the median SWD distance. As shown in [CCO17], this kernel exhibits desirable properties

- equivalence to the 'matching distance'

$$C_1 W_1(D_1, D_2) \leq SWD(D_1, D_2) \leq C_2 W_1(D_1, D_2),$$

- injectivity of the associated feature map ϕ,

$$\phi_{SWD}(D_1) = \phi_{SWD}(D_2) \implies D_1 = D_2,$$

- an approximation of SWD by $\widehat{SWD}$ that has an explicit feature map $\phi_{\widehat{SWD}}$.

First, the equivalence with the matching distance guarantees that the SWD exhibits similar discriminative power, while additionally defining a kernel. Injectivity of the feature map (on the set of finite and bounded Persistence Diagrams) shows that no information is lost. However, that map is intractable. On the other hand, one can consider an approximation of the distance, which has two advantages over the exact form. First, it remains conditionally negative definite, and hence defines a kernel. Second, it has an explicit feature map.

3.5.4 Persistence Fisher Kernel

Both the bottleneck distance (with a Persistence Diagram seen as a collection of point measures) and the SWD can be seen as measuring the cost of transport of one measure $PD_1 \cup \pi(PD_2)$ into another one $PD_2 \cup \pi(PD_1)$, with π denoting the projection on the diagonal. Proposed in [LY18], the Persistence Fisher Distance (PFD) relies on a similar idea, but defines transport using the Fisher information metric, rather than the Wasserstein distance.

Definition 12. To a (multi-)set of points D, the probability distribution $\bar{\rho}_D$ is the normalized persistence surface

$$\bar{\rho}_D(x) = \frac{1}{C}\rho_D(x), \tag{3.17}$$

where $C = \int_{\mathbb{R}^2} \rho_D(x)\mathrm{d}x$ is the normalization constant obtained with a constant weight function $f(x, y) = 1$, for all $(x, y) \in \mathbb{R}$.

The Fisher information metric between two distributions $\bar{\rho_1}$, $\bar{\rho_2}$ is

$$d_F(\bar{\rho_1}, \bar{\rho_2}) = \arccos\left(\int \sqrt{\bar{\rho_1}(x)\bar{\rho_2}(x)}\right).$$

Recalling the projection on the diagonal $\pi : (x, y) \mapsto (\frac{x+y}{2}, \frac{x+y}{2})$, we call Persistence Fisher Distance between two diagrams D_1, D_2 the Fisher information metric between the distributions $\bar{\rho}_{D_1 \cup \pi(D_2)}$ and $\bar{\rho}_{D_2 \cup \pi(D_1)}$

$$PFD(D_1, D_2) = d_F\left(\bar{\rho}_{D_1 \cup \pi(D_2)}, \bar{\rho}_{D_2 \cup \pi(D_1)}\right). \tag{3.18}$$

Similarly to SWD, the distance PFD is shown to be conditionally negative definite- to be more precise, Le and Yamada [LY18] show that $d_F - \tau$ is negative definite, for all $\tau \geq \frac{\pi}{2}$. Therefore, one can use (3.5.3) to define the Persistence Fisher Kernel on $\mathcal{D}$ the set of finite persistence diagrams

$$\kappa_{PF}(D_1, D_2) = \kappa_{PFD,\sigma}(D_1, D_2). \tag{3.19}$$

A stability property of the distance $d(D_1, D_2) = \kappa_{PF}(D_1, D_1) + \kappa_{PF}(D_2, D_2) - 2\kappa_{PF}(D_1, D_2)$ defined from κ_{PF} is shown [LY18], with respect to PFD. However, as this result is based on the Fisher Information Metric, it does not directly provide stability with respect to the usual metrics on $\mathcal{D}$, like the bottleneck, Wasserstein or matching distances.

The exact computations of PFD, and hence of κ_{PF}, are costly. For two persistence diagrams, with cardinalities bounded by N each, constructing each of the measures $4N^2$ evaluations of the Gaussian kernel, from which only $N^2 + N^2$ can be precomputed. However, there are at least two possibilities to approximate this kernel. First, Le and Yamaga [LY18] suggest an using fast Gaussian summation - numerous possibilities are review and proposed by Morariu et al. [MSR+09]. In the implementation of the PFD, Carriere [Car18] leaves the possibility to use kernel approximations such as the radial basis function kernel [RR07]

Chapter 4

Topological fingerprinting of audio signals

In this chapter, we go back to the problem of audio identification. Similarly to work presented in 2.1, our approach is composed of two steps - for a given audio snippet, we compute its fingerprint and, in the second step, compare it with the precomputed fingerprints of a catalogue of songs. We show how to use the topological framework from chap. 3 on the structures suited for audio analysis reviewed in chap. 2 for fingerprinting, and how to compute distances between the representations from sec. 3.5. To be more precise, in sec. 4.1, we define five filtrations - intensity, time, frequency and two distance transform-based filtrations on spectrograms. In sec. 4.3, we introduce local fingerprints, with which we aim to increase the robustness of our methodology to obfuscations.

4.1 Filtrations on spectrograms

Motivating their computer-vision based approach, the authors of [YHS05] outline several characteristics that they deem important in a spectrogram. They mention

1. peaks of power across frequencies and time,
2. differences of power in neighbouring frequency - or time- bands,
3. shifts in dominant frequency over time.

Motivated by these characteristics of the spectrogram, we propose filter functions, that define filtrations on the cubical complex built on the set of vertices of entries of the spectrogram. We first show how we extend a function defined on vertices to a filtration and then, propose the filtrations.

In Section 3.3, we introduced filtrations on cubical complexes and the persistence homology groups. We have shown (3.9) how to define a filtration from a filter function on the whole complex. In applications [ZZJS16], as it is the case for this work, we consider a function $f : X \to \mathbb{R}$ defined only on the set of vertices $X = \mathrm{Vert}(K) = \{(i, j) \in \mathbb{Z}^2 \mid x_m \leq i \leq x_M,\, y_m \leq j \leq y_M\}$ for some complex K. We show how to extend it to a filter function.

Consider the sup- and sub-level sets of f, given by

$$X_r = f^{-1}\left(]-\infty, r]\right),$$
$$X^r = f^{-1}\left([r, \infty[\right).$$

We extend f defined on the set of vertices X to $\tilde{f}$ defined on the maximal complex $Q(X)$ on X, by adding the cubes as soon as possible in the filtration: recall the definition of a filtration 8, which, in that case, implies that Q will appear in the filtration as soon as all of its faces have appeared. We define two different filter functions,

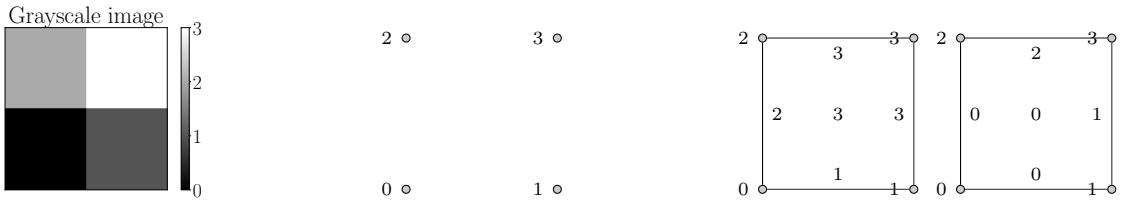

Figure 4.1: A grayscale image (left), a function defined on a set of vertices (centre) and the two filter functions (lower- and upper-star) that it induces (on the right). Concerned about clarity, we drop the notation for cubes and exhibit only the values of the filter function on vertices, edges and the unique two-cube. Notice that while f is injective, the filter functions are not, and more than one cube appears at each value of r.

that give filtrations (13) corresponding to the above intuition. An example is given in fig. 4.1.

$$\tilde{f}_l(\sigma) = \max_{x \in \mathrm{Vert}(\sigma)} f(x),$$
$$\tilde{f}_u(\sigma) = \min_{x \in \mathrm{Vert}(\sigma)} f(x).$$

We will sometimes call them filter and cofilter respectively.

Definition 13. The upper-star filtration associated to f is the filtration $(K^r)_{r\in\mathbb{R}}$, where

$$K^r = \tilde{f}_u^{-1}\left([r, \infty[\right). \tag{4.1}$$

Similarly, the lower-star filtration is $(K_r)_{r\in\mathbb{R}}$, where

$$K_r = \tilde{f}_l^{-1}\left(]-\infty, r]\right). \tag{4.2}$$

When the filtration function is clear, we denote their k-th persistent homology groups by $H_{k,\bullet}$ and $H_k^\bullet$ respectively. In general, we will reduce our considerations to upper-star filtrations.

Definition 14. For a spectrogram S and function f, we denote by $D(S, f, k)$ the persistence diagram of the upper-star filtration induced by f on S. By abuse of notation, for a song s, we use the same notation $D(s, f, k) = D(STFT(s), f, k)$.

Remark. Recall that $Q(V)$ is the maximal cubical subcomplex on the set of vertices $V \subset X$, so the sub- and sup-level sets define filtrations $(Q(X_r))_{r\in\mathbb{R}}$, $(Q(X^r))_{r\in\mathbb{R}}$ on the cubical complex $Q(X)$. These filtrations coincide with $(K_r)_{r\in\mathbb{R}}$ and $(K^r)_{r\in\mathbb{R}}$. Therefore, it is possible to define them without the auxiliary extensions $\tilde{f}_l$, $\tilde{f}_u$.

According to Definition 8, the upper-star filtration is not a filtration, because the inclusions are reversed

$$r_1 \geq r_2 \implies (Q(X^{r_1})) \subseteq (Q(X^{r_2})).$$

While we are not the first to use filtrations based on sup-level sets [BEK10], we make pragmatic observations that we have not seen elsewhere. First, consider the lower-star filtration defined by $\hat{f} = -f$, in which case, the elementary cubes appear in the same order as for an upper-star filtration, but the codomain of f changes sign (what should be born in mind when interpreting the results). Another way around the problem is to recall that PH groups are defined thanks to the monotonicity of the filtration, as it implied the existence of morphisms between the groups of chain. Even when the order is reversed, those morphisms will exist, but for $\phi_{r_1,r_2} : (Q(X^{r_1})) \to (Q(X^{r_2}))$, where $r_1 \geq r_2$. In practice, the only inconvenience is now that the birth value of a PH class will be higher than the death.

For applications, one should be careful as some software packages assume that the death index is always higher than the birth index. In that case, we would recommend using a lower-star filtration with $\hat{f}$, and, if needed, transforming the PDs back to the ones obtained from f, using $(b, d) \mapsto (-b, -d)$.

Remark. The filter functions $\tilde{f}_l$, $\tilde{f}_u$ are not injective, even if f is. Up to four edges and four two-cube can appear at the same time.

Remark. A filter function f induces $\tilde{f}_l$, $\tilde{f}_u$, which exhibit a kind of duality. We notice that some points $(b, d) \in H_{1,\bullet} \cap H_0^\bullet$. In Appendix A, we show that if f is injective, the unique vertex $v_b \in f^{-1}(b)$ is a saddle point.

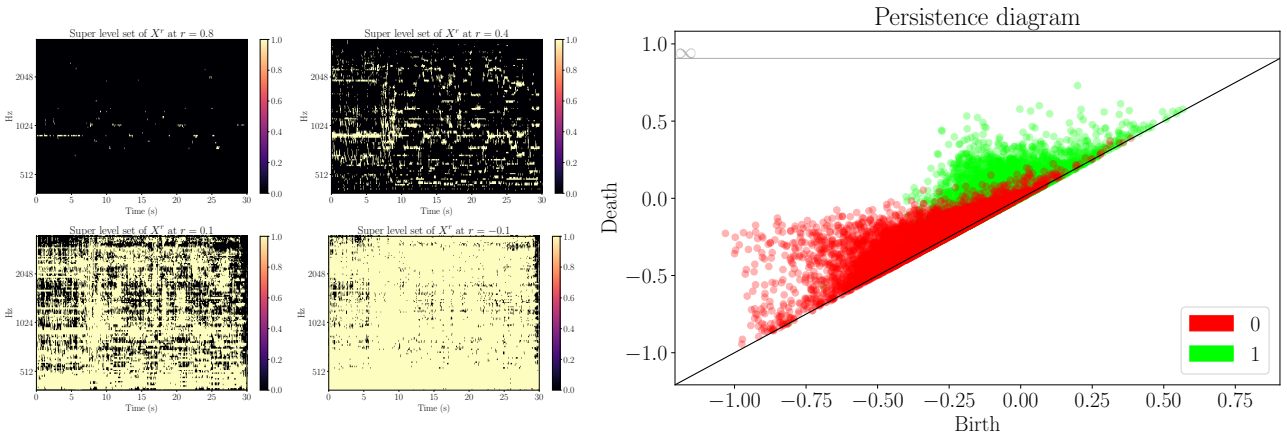

Figure 4.2: Three super-level sets X^q, on the spectrogram fig.2.1, at levels 0.8, 0.4, 0.1, -0.1. As described at the start of sec. 4.1, we are using sub-level filtration on $\tilde{f} = -f$ rather than super-level filtration on f. Using the persistence diagrams from the reversed intensity filtration (on the right), we can identify distinct 'regimes' in the level sets of the spectrogram. At first, the connectivity changes a lot (red points have birth and death values that correspond to high intensity), while the first persistent homology group characterises more the lower levels (intensities from 0.4 to -0.5).

4.1.1 Intensity filtration

Getting back to spectrograms, consider a matrix $S \in \mathbb{R}^{(1+N_{FT}/2)\times N_T}$: the element $S_{N_f,k}$ represents the intensity of frequencies around $N_f \frac{f_s}{N_{FT}}$, in a window of $\frac{N_{FT}}{f_s}$ seconds centred at $k\frac{h}{f_s}$, where f_s was used to denote the sampling frequency and N_{FT} the length of the window function in the STFT (2.1). Inspired by the authors [Wan03], we look interactions between the set of pixels defined by super-level sets.

Definition 15. Let X be the set of vertices corresponding to entries in the spectrogram and $K = Q(X)$ the cubical complex. We call the intensity filtration the upper-star filtration induced by the intensities of S

$$\begin{array}{rccc} f_I: & X & \to & \mathbb{R} \\ & (i,j) & \mapsto & S_{i,j}. \end{array}$$

Let s be a snippet from a song. We denote by S its spectrogram and we get two persistent homology groups

$$\Phi_I : s \mapsto (D(s, f_I, k))_{k=0,1}\,. \tag{4.3}$$

We vectorize them, each with a vectorization method ϕ_k, $k = 0, 1$, in order to apply learning algorithms, as explained in sec. 3.5. Therefore, for one snippet s of a song, we get two elements in $\mathcal{H}$ that we consider separately, as in [CNO18]. In order to compare two songs, we use two metrics - one for each $k = 0, 1$ of the two dimensions

$$d_{\Phi_I,\phi,k}(s, s') = d_{\phi_k}\left(\Phi_I(s)_k, \Phi_I(s')_k\right). \tag{4.4}$$

The stability theorem 2 gives a justification for the use of persistent homology on cubical complexes, and, in our case, applying it on spectrograms. However, to guarantee stability with respect to the initial data (a recording of a song), we need to verify that the filtration function also exhibits such a stability. The intensity filter function is stable, since the spectrogram (2.1) and the equal-loudness mel variant that we use are Lipschitz. Therefore, the persistent homology groups are Lipschitz (in the bottleneck distance), with respect to the audio input.

This seemingly natural filtration has a few shortcomings. First, it discards information about the orientation of a spectrogram, or of its fragments. It implies that a spectrogram S and its transpose S^T give the same persistent homology groups, while, in general, it is not true that they would represent the same song. This comes from the false symmetry between the axes, present in the definition of a cubical complex. While for an image, like a picture, the axes represent the same dimensions, making this symmetry desirable, it is not the case of spectrograms, where the time and frequency scales are different. Second, it is challenging to rely on the

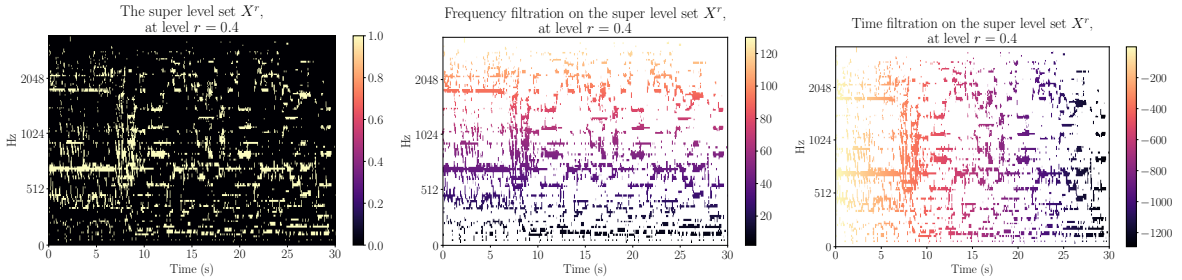

Figure 4.3: On the left, a binary image representing X^r,for $r = 0.4$: white pixels correspond to elements present in the picture. In the centre, a frequency filtration on that spectrogram. The pixels in the top rows have higher values and are added first. On the right, a time filtration, where the vertices are added from left to right.

vectorization methods that use fixed bins. In our case, this concerns persistence images or Betti curves. To be more precise, on raw spectrograms, we do not know what is the birth nor death scale of features. While, for example, the equal-loudness mel spectrogram is bounded, we do not know at which level the different 'regimes' exemplified in Figure 4.2 occur. One faces two choices - normalising the filtration or adjusting the vectorization method. Normalisation has the disadvantage of shifting the whole persistence scale, based on the value of the maximum, which may change, especially when extracting windows. We choose therefore to adjust the vectorization methods, to always take into account the same (and wider) range of values.

4.1.2 Time and frequency filtrations

The undesired invariance to rotation of f_I leads us to consider filtrating functions defined by the axes. In fig.4.2, we noticed that there are many 'lines' in the super-level sets, especially for $r = 0.1, 0.4$ in that example. Let us describe how we use persistent homology to study the branching of these 'lines': their formation, merging and splitting of those lines.

For $r \in R$, we fix an intensity super-level set $X^r = f_I^{-1}([r, \infty[)$: that is, we look at the set of vertices, where the intensity values in the spectrogram are greater than r. Starting from an empty set of vertices, the time filtration is defined by adding vertices from X^r successively from left to right. The frequency filtration is similar, although we proceed by in a top to bottom manner: adding first vertices that correspond to high frequencies in the spectrogram, moving towards those of lower frequency. We can think of the persistent homology of the time filtration as capturing the 'branching' in the set X^r. Formally, the frequency filtration is the upper-star filtration induced by (4.5) as shown in (4.1), with an example in fig.4.3. Similarly for the time filtration, which is characterised analogously, with (4.6).

$$\begin{aligned} f_{F,r}: \quad X^r &\to \mathbb{N} \\ (i,j) &\mapsto \begin{cases} i, & S_{i,j} \geq r \\ -\infty, & \text{otherwise,} \end{cases} \end{aligned} \tag{4.5}$$

$$\begin{aligned} f_{T,r}: \quad X^r &\to \mathbb{N} \\ (i,j) &\mapsto \begin{cases} -j, & S_{i,j} \geq r \\ -\infty, & \text{otherwise.} \end{cases} \end{aligned} \tag{4.6}$$

Remark. The negative sign in the time filtration serves to keep consistent with other filtration types by having an upper-star filtration. We could have as well defined $f_{T,r}(i,j) = j$ for $\hat{S}_{i,j} \geq r$ and used the lower-star filtration.

For these two filtrations, the intensity level now becomes a parameter - for example, different levels do not exhibit the same connectivity, meaning that even when the filtration has finished (all the elements from X^r have been added), the number of connected components will be very different. In this work, we choose to examine three level-sets: one where most of the changes occur at the connectivity level, a middle one where both homology groups are affected and one where the number of holes (H_1) changes. The intensity values of interest

are defined as the quantiles, computed for $q = 0.25, 0.5, 0.75$. For a spectrogram S, we label the quantile q by $L_S(q)$. The feature map is then composed of six persistence diagrams - two per level. In our feature maps and the corresponding metrics, we choose to aggregate the fingerprints from different levels for the same dimension: this approach is motivated by studying the spectrogram itself, underlying the three level-sets. Thus, we still obtain two fingerprints for one song. To persistence diagrams $\Phi_F(S)_0$, $\Phi_F(S)_1$, where

$$\Phi_F : S \mapsto (D(s, f_{F,L_S(0.25)}, k), D(s, f_{F,L_S(0.5)}, k), D(s, f_{F,L_S(0.75)}, k))_{k=0,1},$$

we apply vectorization methods $\phi_k, k = 0, 1$ choosing parameters for each level and dimension separately.

$$d_{\Phi_F,k}(S, S') = \sum_{r \in L(\{0.25,0.5,0.75\})} d_{\phi_k}(\phi_k(D(S, f_{F,L_S(r)}, k)), \phi_k(D(S', f_{F,L'_S(r)}, k))),$$

The definitions of Φ_T and $d_{\Phi_T,k}$ are obtained analogously.

Remark. Recall that some distances d_Φ could be turned into kernels, by (3.5.3). By defining a metric between a collection of features as a sum of conditionally-negative distances, we obtain a conditionally negative distance as well. Therefore, it defines a positive-definite kernel.

The cubical complex K^r as defined in (4.1) is not contractible - depending on the value of r, it can have multiple connected components and non-trivial loops. In fact, none of the classes in $H_{1,\bullet}$ ceases to persist, and they thus all have deaths equal to $-\infty$. This presents a challenge when comparing persistence diagrams, because the distance (its infinite) for diagrams with a different number of points with infinite persistence. To solve this issue, we replace the $-\inf$ in (4.5) and (4.6) by finite values r_∞, such that, at K^{r_∞}, all the missing cubes appear, resulting in only one connected component remaining and no non-trivial element in $H_0(K^{r_\infty})$.

Let us compare these filtrations to the intensity filtration. First, for each spectrogram S with all distinct intensities, there exists a neighbourhood in $\mathbb{R}^{(1+N_{FT}/2)\times N_T}$ of S, such that the level-sets remain the same, and therefore, the filtration is unperturbed. However, its is not uniform, over $\mathbb{R}^{N_f \times N_t}$. In addition, if for a quantile q, the level-sets $X^{L_S(q)}$, $X^{L_{S'}(q)}$ are not the same for two spectrograms S, S', the difference between their corresponding filter functions (either frequency or time) is not bounded.

4.1.3 Distance transform filtrations

In the previous section, we defined the axis-filtrations on super-level sets, which are binary images. In image processing, to look at the morphology of such images, one uses the distance transform. For each pixel present in the image (foreground), its distance from the boundary (or background) is computed. We use distance transforms in two contexts - intensity level-sets and vertical edges. First, we recall its definition, and we use it to study the intensity-induced morphology (sec.4.1.3), and then, we turn to analyse how each frequency changes over time (sec. 4.1.3). For a binary matrix $M \in \{0, 1\}^{n\times m}$, the distance transform $T(M)$ is

$$T(M)_{i,j} = \min_{(k,l)\in M_0} \{\|(i, j) - (k, l)\|_1\},$$

where $M_0 = \{(k, l) \mid M_{k,l} = 0\}$ is the background. An example is shown in fig. 4.4.

Remark. In practice, calculating the distance transform can be performed by successive erosions of the foreground. The structuring (eroding) element should be chosen according to the distance to be used. To get the Manhattan (ℓ^1) distance transform, we use the 3×3 element

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

This complexity of this method is linear in the number of elements of the image [CM99], and, therefore, the overall complexity of fingerprinting a song is still dominated by computations of persistent homology.

There is also a dual distance transform, where one looks at the distance of background pixels to the foreground - it corresponds to $T(1-M)$. This is the basis of our intensity-level distance transform filtration described next.

We calculate the distance transform from a binary image. Therefore, similarly to the axis filtrations from Section 4.1.2, they involve the choice of level sets. Aggregating the fingerprints for different levels is done in the same way as for the axis-filtrations. When calculating the distances between fingerprints composed of persistent homology groups for different levels, we compute the distances for each level separately and add them together.

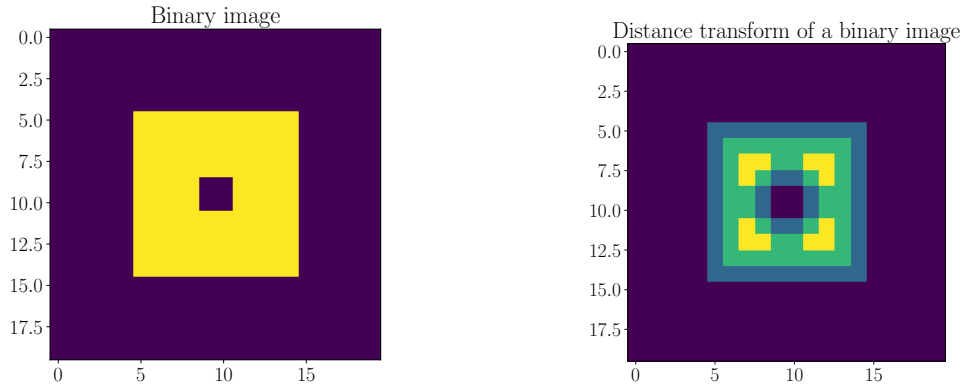

Figure 4.4: A binary image and a distance transform of a binary image M. For each pixel in the foreground, its distance to the background is calculated.

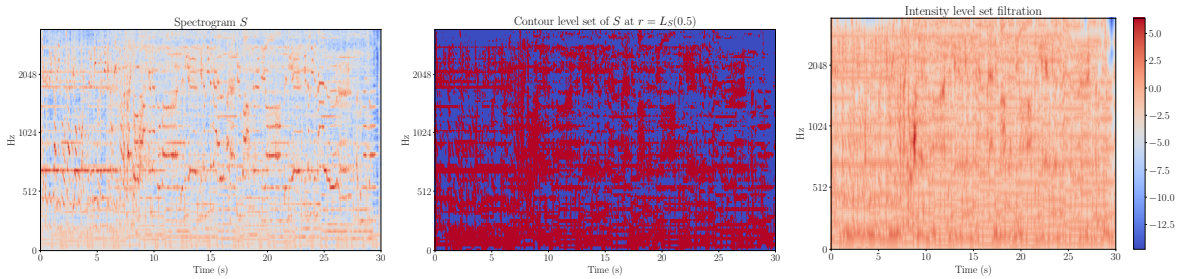

Figure 4.5: In the middle, the matrix representing the median level-set: pixels with values higher (resp. lower) than the median are marked in red (resp. blue). The filter function defined by (4.7) is shown on the right. We can see that many pixels have very similar filtration values, meaning that many cubes will appear at the same time.

Intensity level-sets

When applying persistent homology to point-cloud, not only the proximity should be taken into account. Indeed, while persistent homology is not sensitive to noise in that context, the introduction of a single point far from the others can significantly alter the results. One remedy is to include information from the co-density function, which is effectively a density estimation based on the sample. In [Vip18], a Gaussian density is used, with the bandwidth as a parameter. Here, rather than the density function, the intensity defines the level set.

For an intensity r, we look at the distance transform of X^r and X_r, by interpreting them as binary matrices: $X^r_{i,j} = 1$, if $S_{i,j} \geq r$ and $X^r_{i,j} = 0$ otherwise. The intensity level-set filter function represents the distance of each pixel to the level set $f^{-1}(r)$.

$$\begin{array}{rccc} f_{T(I)}: & \mathbb{R}^{n_1 \times n_2} & \to & \mathbb{Z}^{n_1 \times n_2} \\ & S & \mapsto & T(X^r) - T(1 - X^r), \end{array} \tag{4.7}$$

where $(S^r)_{i,j} = 1_{S_{i,j} \geq r}$ and $n_1 = (1 + N_{FT}/2)$, $n_2 = N_T$. We call this the intensity-level distance transform filter function. The corresponding filtration is its upper-star filtration (3.9), as for the intensity, time or frequency filtrations. We compare this filter function with the intensity one. First, stability with respect to the spectrogram (and hence, the signal) is lost, due to the use of level-sets. On the other hand, the filtration values are bounded (as is the matrix size). The filtration has therefore relatively few steps The computational complexity of defining the filter function is dominated by that of calculating the distance transform, which is linear in the number of pixels in the binary image.

Edges in the time axis

The next filtration we propose is inspired by looking at the changes of the dominating frequency in time and uses basic image analysis techniques. First, we identify the vertical edges in the spectrogram and then, apply a distance transform to a super-level set. In that last step, we use a one-dimensional distance transform, that only takes into account time. This allows us to identify time intervals between 'major' changes in intensity, for a given frequency. While we find this filtration well-motivated, it gives outstandingly-bad results. The simplest vertical edge-detection approach from image processing consists of convolving the image with a differentiation kernel $(-1, 0, 1)$ (or $(-1, 0, 1)^T$ for horizontal edge-detection). We use the Sobel operator, we first differentiate S and convolve the result with a smoothing kernel $(1, 2, 1)^T$, what gives the edge-detection operator

$$\begin{array}{rccl} E: & \mathbb{R}^{N_f \times N_t} & \to & \mathbb{R}^{N_f \times N_t} \\ & S & \mapsto & E(S) = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} * \begin{pmatrix} -1 & 0 & 1 \end{pmatrix} * S, \end{array}$$

where $*$ stands for convolution. For a level-set s, we apply a one-dimensional distance transform to the absolute value of the matrix $|E(S)|^s$

$$\begin{array}{rccl} f_{T(E)}: & \mathbb{R}^{N_f \times N_t} & \to & \mathbb{Z}^{N_f \times N_t} \\ & S & \mapsto & T(E(S)^s), \end{array}$$

where in T, we use a one-dimensional eroding element $(1, 1, 1)$.

The edge filtration has significant advantages compared to previously-outlined methods. First, by differentiating the matrix, the super-level sets are no longer influenced by the absolute, but only relative and local intensities. This can be important when treating small snippets of spectrograms. Second, similarly to the time-axis filtration, it should be not be invariant to a rotation exchanging the time and frequency dimensions. Finally, while $f_{T(E)}$ reflects the distance in time, it is not subject to absolute, but only relative time, thanks to applying distance transform along the time-axis. By introducing the absolute value, we obtain a filtration that is insensitive to time reversion, as shown in table 4.6.

4.1.4 Invariants of persistent homology groups

Homology groups are known to be invariant to a wide range of transformations. This partially extends to persistent homology groups. In this section, we examine the invariants that the proposed filtrations exhibit, in the context of analysis of spectrograms, and audio id in particular. We examine the following transformations

1. rotation by $\pi/2$,
2. time-reversion,
3. frequency reversion,
4. magnitude scaling.

We perform the experiment based on a patch extracted from a spectrogram. Based on these observations, we can infer how the filter functions will behave with respect to other obfuscations, like noise addition, pitch shifting, time stretching or reversing a song.

The persistent homology groups given by intensity filter function and the intensity level-set filter function show the most invariance in Table 4.6. However, the properties of the latter are due to the conservation of the range of values. Recall that the first step consists of generating two images, whose boundary is the level-set defined by the median. While this level-set is stable with respect to shifts in the spectrogram, or scaling of intensity values, it is not with respect to perturbations of individual values, like it is the case for noise addition.

The axis filtrations behave as expected - they are not invariant with respect to a rotation and reversion on the same axis - a frequency filtration is not stable with respect to the frequency reversion. Due to the use of level sets, they are (similarly to the intensity level-set filter function) not sensitive to scaling.

Finally, the edge-filter function gives a filtration that is invariant with respect to all reversions and scaling, but changes when rotated. This is achieved thanks to each of the two components: calculating the gradient along one of the axes, and using a one-dimensional distance transform.

	Intensity	Frequency	Time	Intensity level-sets	Edges
Rotation	Y	N	N	Y	N
Time reversion	Y	Y	N	Y	Y
Frequency reversion	Y	N	Y	Y	Y
Scaling	N	Y	Y	Y	Y.

Figure 4.6: Table summarising the invariance of persistent homology groups induced by the different filter functions proposed in Section 4.1, with respect to the transformations. 'Y' denotes that the invariance is respected, while 'N' that it is not.

Remark. It is difficult to compare the spectrograms of a snippet and that of a song using the framework of cubical complexes and persistent homology. As the spectrograms are of different sizes, we cannot directly compare the norm of the filtration functions. To make a comparison possible, we could add constant-valued bands at the edges of the spectrograms. By including those values in the filtration at the end, we do not perturb the ordering nor the persistent homology groups in the filtration. By making those bands bigger the smaller the window we extract, we keep the domain of constant size.

4.2 Vectorization methods

We present the methods that we use to bridge between between the results of persistent homology groups calculated using filtrations from sec.4.1 and classification algorithms. The vectorization and kernel methods reviewed in sec.3.5 require setting parameters and interpretation for our fingerprints. We comment on how suitable these methods are for the filtrations that we propose.

We use the Persistence Image implementation from the package sklearn tda by Carriere [Car18]. While the persistent homology calculations are one using GUDHI [Dlo15], which also features an implementation of Persistence Images [Dlo17], we find that the idea behind the latter produces instability, and in two different ways. First, the size of the image is fixed and the range (on the persistence scale) is defined using the extremal points of each persistence diagram independently. Hence, by adding one point with low persistence, but born after all the others, the range of the image, and thus the assignments of points to pixels, change. The second issue comes from assigning the points to pixels in a discrete fashion, before the convolution. For a point near the border of a bin, a small perturbations can thus change the bn it is assigned, giving rise to unstability. The approach proposed in the implementation by Carriere [Car18], solves both issues. First, the range is fixed, by fitting the method on a set of persistence diagrams. Then, the persistence image is computed 'exactly'- every point contributes to every pixel, based on its distance to that pixel - while still linear in the number of points in persistence diagram, the computational cost is higher.

We additionally test our own implementation, which is a compromise between the two - similarly to GUDHI, we perform an assignment of points to pixels, preceding the convolution of the image with a Gaussian filter. However, instead of a hard assignment, we distribute the weight of a point to the pixel it should be assigned to, and to its neighbours, based on the distance of that point to the centres of those pixels. While being more accurate than the hard assignments, it is less costly - we need to evaluate only the distances of a point to nine, instead of all, pixels. Our approach has a drawback - points falling far from the pre-determined range are discarded- they do not contribute to any pixel. Hence, we lose discriminatory power when evaluating distances. This contrasts with Carriere's method, which, as far as the distance is concerned, can be then compared to integrating the Persistence Surface over a bounded, rectangular domain, and accounts for the contribution of points that are far away.

Similarly to Persistence Images, we find that the range for Betti Curves should be fixed a priori. In addition, this vector representation is better suited for axis filtrations that we propose. In $H_1^\bullet$, obtained with $f_{T,r}$ or $f_{F,r}$, all the points are essential - the associated homology features do never become trivial. As stated in 4.1.2, we 'complete' the filtration, in order to have finite components, but they are nonetheless completely determined by their birth coordinate. Hence, Betti Curves are non-strictly decreasing functions of the filtration parameter r (or increasing, if we consider the negative scale, as pointed in Section 4.1) encode all information about the points.

Parameter	Choice
Weighting function	$f(x, y) = y$,
Bandwidth	median persistence,
Image size	fixed,
Pixel dimensions	fixed.

Table 4.1: Summary of the choices for the Persistence Images vectorization method.

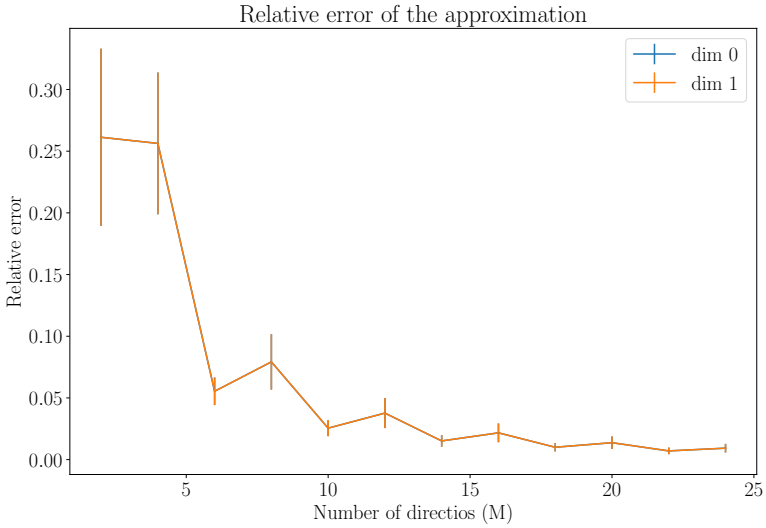

Figure 4.7: The figure shows what M we need to correctly capture the distance between Persistence Diagrams for our application (left) and to correctly classify audio snippets (right). For this experiment, we restrict ourselves to the intensity filtration 4.1.2.

Regarding the sliced Wasserstein distance SWD, we evaluate the number of directions needed to compute the approximation on our dataset. Based on Figure 4.7, we use $M = 10$.

Remark. When computing the SW distance between one persistence diagrams and many others, it is advantageous to compute and sort the projections first. Then, at query time, we can merge two sorted arrays of length N_1, N_2, what is done in $\mathcal{O}(N_1 + N_2)$. On the other hand, this implies storing the projections of persistence diagrams on the set of lines, what increases the memory use by a factor of $M/2$ compared to storing persistence diagrams.

The above-mentioned methods should be evaluated in terms of their suitability for audio id fingerprinting. Notice that, for each song or snippet, the fingerprint is computed exactly once and a classification algorithm is applied. Some algorithms, like nearest neighbours, require multiple comparisons. Therefore, we should rely on methods that do not compromise accuracy (in particular, the discriminative power) and that feature a distance fast to compute. For this reason, kernel methods are less suitable, as the comparison time is linear in the number of points in a persistence diagram. On the other hand, vectorization methods have a codomain of fixed dimension, that can be therefore compared independently of time from the cardinality of $H_{\bullet,k}$. In addition, such a feature map is required by a range of machine learning algorithms, like random forest, or approximate nearest neighbour. In addition to potential gain in time efficiency, the memory consumption of explicit feature maps is constant, and if needed, can be adjusted to be considerably less than storing entire persistence diagrams.

4.3 Local approaches to topological fingerprinting

The fingerprinting methodology presented in previous sections is the following: given a matrix, choose a filtration, calculate and vectorize the resulting persistent homology groups. In our case, the matrix is a spectrogram and we have motivated the filtrations accordingly. By fingerprinting the whole spectrogram, we

implicitly assumes a model, where the spectrogram of a song has topological characteristics that are robust to obfuscations and we can recover from a small window. In this section, we propose two local approaches - time-windows in a spectrogram, and patches around landmarks. The first one is motivated by [TB15, YHS05], while the second one relies on the importance of landmarks (as, for example, those in [Wan03]). Both of these ideas consist of studying local, rather than global, characteristics - in time, in the first case, and in time-frequency in the latter. In approaches like [Wan03, TB15], the similarities of local fingerprints are used to compute a likely alignment of a snippet with respect to a song. In contrast, we forget about the time coordinates of our windows, and treat their fingerprints as a set.

4.3.1 Windows

The audio id method proposed in [YHS05] relies on decompositions of small windows from a snippet and extracting fingerprints that are local in time, but cover the whole frequency spectrogram. We adopt a similar approach. For a spectrogram S, we denote $S_{:,k_1:k_2}$ the window that spans from index k_1 to k_2. Recall that, for an index k, $t_k = \frac{kf_s}{h}$ is the time-position of the centre of the window used to compute the spectral coefficients $S_{:,k}$. We choose $k_2 - k_1 = 34$, what corresponds to $t_{k_2} - t_{k_1} \approxeq 0.8s$ and is close to $0.941s$ used in [BC07]. By choosing a hop length of $0.1s$ (so 4 columns), we guarantee that a pattern that occurs in the spectrogram is captured by at least one of the windows, and we obtain 314 extracted windows for a $30s$ audio. We denote the set of starting window indices k_1 by $W(S)$. To summarize, one song s provides us with 314 windows of sizes 132×34 (for the equal-loudness mel spectrogram), each independently fingerprinted. However, as stated in 4.5, we do not succeed in comparing this approach to the global one and to patches, described below.

4.3.2 Landmarks

Fingerprinting sliding windows in a spectrogram is an approach to extracting local features that treats all parts of the spectrogram with equal importance. Regardless of the actual structure of the song and presence of patterns, we always extract similarly-positioned windows. In this section, we present how to choose patches based on the structure of the intensities. Using the FAST corner- detection algorithm [RD06], we identify landmarks and study their neighbourhoods.

The landmark detection algorithm that we choose is a corner detector introduced in [RD06]. A sixteen-pixel neighbourhood of each pixel is examined - if the pixel is much brighter or darker than most of its neighbours, it is labelled as a corner candidate. Rosten and Drummond propose a method to filter candidates, in order to prevent several neighbouring pixels from being labelled as corners. Once we have a set of landmarks $C(S) = \{(f_k, t_k)\}_{k=1}^{N_P}$, we extract patches of sizes $2p_f \times 2p_t$ centred on each one. Interpreting the patches as a collection of spaces, we choose a filtrating function f, and obtain a set of topological fingerprints for each patch. When the position of the landmark is too close to the boundary, that is $t_k \notin \{40, \dots, 1232-40\}$ or $f_k \notin \{20, \dots, 132-20\}$, we consider a patch of smaller size. The approach is summarized in Algorithm 1 and the detection and extraction of patches is shown in fig.4.8.

Algorithm 1: Fingerprinting method on patches

Input: Song s, filtration
Result: $\Phi(S)$, the fingerprinting of s
Compute the equal-loudness mel spectrogram S;
Detect a set of landmarks $C(S) = \{(f_k, t_k)\}_{k=1}^{N_P}$;
Initialize the set of fingerprints F;
foreach *k=1,...,L* **do**
  Define $X = S_{[f_k - p_f : f_k + p_f, t_k - p_t : t_k + p_t]}$;
  Compute the topological fingerprint $\Phi(X)$;
  $F \leftarrow F \cup \{((f_k, t_k), \Phi(X))\}$;

We use the OpenCV [Bra00] implementation of the FAST algorithm [RD06], in which the threshold and a bound for the number of patches can be set. Thus, our landmark method implies the choice of

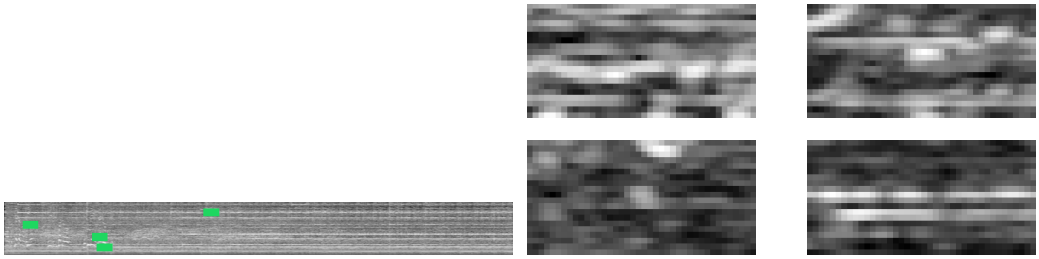

Figure 4.8: A spectrogram (on the left) and a subset of the detected patches are shown. We notice that the top left patch is slightly smaller than other patches, as it was detected too close to the boundary.

- a threshold for candidate point selection,
- a bound on the number of landmarks,
- a size of the patches.

Experimentally, we notice that the number of landmarks increases considerably when noise is added and hence, concerned about robustness of our treatment, we bound the number of selected points by 50. A similar argument is used for the threshold of candidate point selection. In this case, we set it $T = 50$. We have not investigated the impact of this choice on accuracy results rigorously, but it allows us to detect at least one landmark for short snippets (as described in sec. 4.5). While, for an image, such a threshold could be regarded as restrictive, we study patches around corners, and hence, allowing too many landmarks results in patches covering the whole spectrogram. The relation between patch sizes $2p_f \times 2p_t$ and the performance of our method is shown in Figure 4.11. One extension would be adjusting both the threshold and the number of selected landmarks dynamically: the longer a snippet, the more landmarks we should expect.

4.4 Classification

Our audio id approach consists of two steps - fingerprinting and querying. Having described how to obtain a fingerprint from a song: calculating persistent homology groups from a filtration defined in sec. 4.2, on one of the approaches from sec. 4.3, we now describe the querying step. To be more precise, each of the the methods: global, window or patches, implies different

In sec. 3.5, methods to compare persistence diagrams were presented. However, the filtrations that we use and the variety of approaches how to define filtrations on an image, bearing in mind that our images are spectrograms. In sec. 4.3, we we have hinted at how to make comparisons between fingerprints for whole spectrograms, we did not outline how to do so for local approaches.

4.4.1 Distances between fingerprints

Fix a filter function f from one of the functions proposed in sec. 4.1 and a feature map ϕ introduced in sec. 3.5. Recall that such a map is given by either a vectorization method or, implicitly, by a kernel. For a song or snippet s, we compute the spectrogram S. We call computing the fingerprint of the whole spectrogram $F(S) = \Phi(S)$ the global approach. In sec. 4.3, we introduced the windowing and patches methodologies: $F(S) = \{\Phi(S_{:,k:k+34})\}_{k\in W(S)}$ and $F(S) = \{\Phi(S_{[f-p_f:f+p_f,t-p_t:t+p_t]})\}_{(f,t)\in C(S)}$ respectively. Now, we outline how we use those fingerprints for the query step.

The codomain of Φ is a normed vector space V, with a distance d_Φ. In the global approach, comparing two snippets (or songs) s, s' is done by computing the d_Φ between their fingerprints $\Phi(S), \Phi(S')$. To compute the distance between two sets of local fingerprints (for example, one for s and s' each), we propose to use an asymmetric pseudo-metric. Let $\gamma : P \to P'$ be a map between two sets of indices - in our case, P is either $W(S)$ or $C(S)$ (similarly for P'). Then, we choose

$$d_F(F(S), F(S')) = \min_{\gamma} \sum_{p\in P} d_\Phi(\Phi(S_p), \Phi(S'_{\gamma(p)})), \tag{4.8}$$

what is a cost of associating to elements from P to elements from P'. In practice, one would compute all the distances $\left(d_\Phi(\Phi(S_p), \Phi(S'_{p'}))\right)_{p\in P, p'\in P'}$ and construct a minimal matching. Without any restrictions on γ, we can permute the sum with the minimum, what amounts to, for each $p \in P$, choosing its neighbour in P'

$$d_F(F(S), F(S')) = \sum_{p\in P} \min_{p'\in P'} d_\Phi(\Phi(S_p), \Phi(S'_{p'})).$$

When a snippet is extracted from a song, we expect to find all the topological characteristics of the snippet in the whole song, while the converse is not true in general. Hence, we should only require γ to be injective, but this has two drawbacks. First, this measure of dissimilarity is not symmetric. Second, there is a chance of 'false matches', of which we distinguish two types. The first type can occur for both local approaches - an element from the fingerprint set $\Phi(S_p)$ will match to $\Phi(S_{p'})$ that is of small norm. The second one occurs only for the patches methodology, as the window does not span the whole frequency range. It is possible that similar spaces (patches as matrices) are in different parts of the spectrogram. They will inevitably exhibit similar persistent homology fingerprints, but may have nothing in common.

We propose a solution to these problems for patches: we redefine the distance to take into account the frequency components of their locations. To be more precise, to compare two patches, we first verify that they are centred at similar frequencies. If they are, we compare the persistent homology fingerprints, and otherwise, we set that distance to ∞, that is

$$d_{\Phi,F}(\Phi(S_p), \Phi(S'_{p'})) = \begin{cases} \inf, & \text{if } f/f' \in]1/c_F, c_F[\\ d_\Phi(\Phi(S_p), \Phi(S'_{p'})), & \text{otherwise,} \end{cases} \tag{4.9}$$

where $p = (t, f)$, $p' = (t', f')$ and c_F is the threshold for similarity of frequencies. For our case, we take $c_F = 1.3$.

4.4.2 Classification algorithm

The audio id task is a single class classification problem [MG13] - we treat each song as a separate class and assign a single label to a queried snippet. While it may not be realistic in an application setting, we assume that a query can be performed only for snippets coming from songs that are in the database. Algorithms for the classic, binary classification problem can be adapted, via, for example, a one vs all or one vs all approach to multi-class classification. However, given that the number of classes is equal to the number of songs in the database, we use a Nearest Neighbour classifier.

We choose to use a Nearest Neighbour classifier, with $k = 1$. For the global approach, we assign to s the label of its nearest neighbour in the set $\mathcal{S}$ of all songs

$$y_s = \operatorname*{argmin}_{s'\in\mathcal{S}} d_F(F(s), F(s')). \tag{4.10}$$

In (4.8), we have defined a distance between patches and windows that we could similarly use. Instead, we adopt a 'majority voting approach', similar to what is proposed in [Wan03, Tra17]. Computing distances between songs requires comparing patches. Hence, rather than matching a song to other songs, we can retrieve the nearest neighbours of its local fingerprints (either patches or windows) from the set of all local fingerprints $\bigcup_{s'\in\mathcal{S}}\{F(s')\}$. Then, for each song $s \in \mathcal{S}$ with local fingerprints indexed by P, we say that

$$y_s = \operatorname*{argmax}_{s'\in\mathcal{S}} c_{s'}, \qquad \text{where } c(s') = \sum_{p\in P} 1_{s'}(\rho(\tilde{y}_{s,p})),$$

where $c(s')$ counts the number of the number of times that a patch of s' is in the nearest neighbours of a patch of s. Indeed, $\tilde{y}_{s,p} = \rho(\tilde{y}^l_{s,p})_{l=1}^k$ are the labels (song) of the k nearest neighbours of the patch or window p, identified in s, and ρ maps the index of a patch to the song it comes from.

We need to be careful in setting the number of nearest neighbours. First, in the global approach, since we have only one element for each label (song), we need to set it to one. For local features, since we have many songs, it is likely that a similar patch will appear in more than one song. Therefore, we should consider more than one nearest neighbour. On the other hand, with k too high, we witness the false positive matches problem described in Section 4.3. In this work, we set $k = 1$ for the number of neighbours of patches, as for

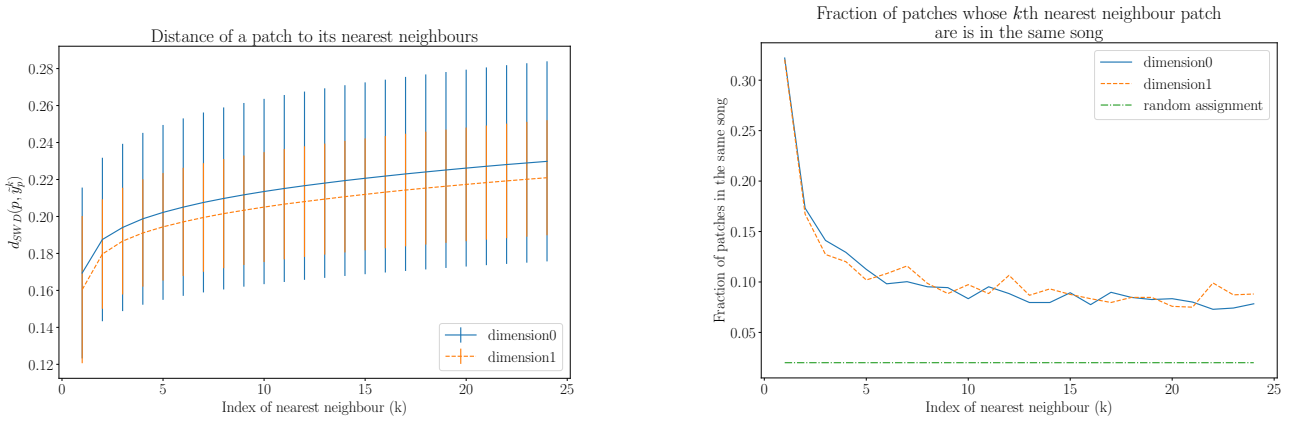

Figure 4.9: The mean distance of a patch to its nearest neighbour is shown on the left. The experiment was performed on a subset of 50 songs, using the frequency filtration. The first nearest neighbour of each patch was correctly identified as itself, with distance zero, and hence, is shown on neither of the two plots. The mean distance plot shows that only the first three neighbours are much closer to the patch than the rest. The right plot shows the fraction of patches, for which the k-th nearest neighbour is in the same song. In addition, the fraction of songs that would be correctly identified from the k-th nearest neighbour of one of its patches - for $k \geq 5$, it drops linearly in k.

small obfuscations, using the distance $d_{\Phi,F}$ (4.9) prevents any false-positive matches. Our approach should be extended by computing features of l obfuscated samples and incorporating them $\mathcal{S}$, with the labels of the original songs, and setting $k = l$. Treating patches windows from the same song as separate points with the same label has the additional benefit of reducing the dimensions in the feature space. Nearest Neighbours suffers from the curse of dimensionality and fingerprinting globally produces one feature of high dimension, while the local approach - multiple small-dimensional.

For our audio id application, the nearest neighbour approach has advantages. Depending on the feature map Φ, adding a new label can be cheap - as it consists of adding a point to the high-dimensional feature space. In practice, for vector maps of fixed codomain dimension, one usually builds index trees, to speed up querying - we rely on Annoy: an approximate nearest neighbours package by Bernhardsson [Ber13]. However, it supports only a limited number of distances on real vector spaces, what makes us turn to the scikit-learn nearest neighbours implementation [PVG+11] when using the distance that takes into account the frequency component (4.9).

4.5 Results

We test our approach on a subset of songs from the Million Song Datasat (MSD) [BMEWL11]. Our comparison is based on four types of obfuscations - noise addition, window extraction, pitch shifting, time stretching. We define them and evaluate the performance of filtrations presented in Section 4.1, applied in the global and local (Section 4.3) settings.

The noise obfuscation is applied on the track level, by adding normally and independently distributed random variables at each sample.

Definition 16. Consider a track $s \in \mathbb{R}^N$. For a noise intensity σ, the noise obfuscation is defined sample-wise as

$$\begin{array}{rccc} T_{N(\sigma)}: & \mathbb{R}^N & \to & \mathbb{R}^N \\ & s & \mapsto & (s_k + \epsilon_k)_{k=1}^N, \end{array}$$

where $\epsilon_k \sim \mathcal{N}(0, \sigma)$ is i.i.d Gaussian Noise.

For the extraction of windows, we have two operators, which will be shown to produce similar results. For a track s, one can extract the window from the signal at the audio level using $T_{W(k_1,k_2)}$, or, extract the window from the spectrogram $S = STFT(s)$: this is what $T_{SW(k_1,k_2)}$ does.

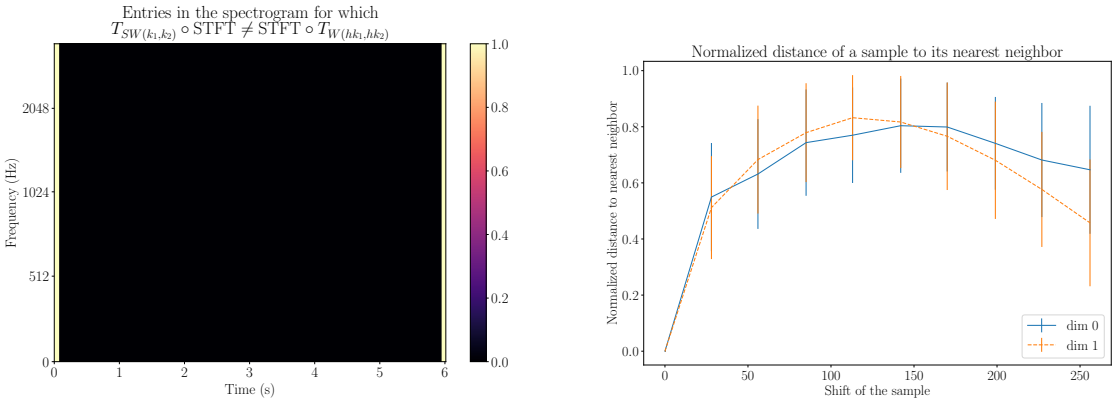

Figure 4.10: On the left, we show that extracting a window from a spectrogram $T_{SW(k_1,k_2)} \circ STFT$ and calculating the spectrogram of an extracted window $STFT \circ T_{W(hk_1,hk_2)}$ are almost the same, except for the last column. On the right, we evaluate the distance between the fingerprint of a window from a song s $\left(\Phi STFT \circ T_{W(hk_1,hk_2)}\right)(s)$ and the fingerprint of a slightly shifted window $\left(\Phi STFT \circ T_{W(hk_1+k^*,hk_2+k^*)}\right)(s)$, for k^* smaller than the hop length h. We verify that for $k^* \approxeq h/2$, the distance between the two fingerprints is the biggest and that it decreases as k^* approaches the hop length $h = 256$. However, for any k^*, we achieve perfect classification (100%) on the subset of 150 songs used for the test.

Definition 17. Set $k_1, k_2 \in \mathbb{N}$. Windowing on the track and the spectrogram is defined using the notation adopted in Section 4.3,

$$\begin{array}{rccc} T_{W(k_1,k_2)}: & \mathbb{R}^N & \to & \mathbb{R}^{k_2-k_1} \\ & s & \mapsto & s_{k_1:k_2}, \end{array} \qquad \begin{array}{rccc} T_{SW(k_1,k_2)}: & \mathbb{R}^{N_f \times N_t} & \to & \mathbb{R}^{N_f \times (k_2-k_1)} \\ & S & \mapsto & S_{:,k_1:k_2}. \end{array}$$

In the audio id setting, T_W models the user's input best. In our experiments, we use T_{SW} instead- in addition to being more convenient from a computational perspective, it is natural from a cubical-complex standpoint, as applying T_{SW} amounts to extracting a sub-complex. While we expect the two spectrograms to coincide to a large extent, we need to quantify the impact that the small discrepancies have on persistent homology - this is illustrated on Figure 4.10. The perfect classification results for shifted spectrograms shows that we can restrict our considerations to extracting windows directly from the spectrogram. For time-stretching and pitch-shifting, we use the Librosa sound-processing library [MRL+15]. Time stretching is done via a phase-vocoder, based on [Ell02] - given a new time-path, the magnitudes of the for the new columns are obtained by linearly interpolating those in the original spectrogram. The phase is adjusted, so as to perform signal reconstruction. Pitch-shifting by r octaves is computed as time stretching by a factor of r, treating the signal as sampled at f_s/r and resampling at r.

Remark. Using the vocoder is necessary if we aim to obtain an audible track, while, for these experiments, we are only interested in the spectrogram. Time stretching could therefore be achieve by interpolating the columns in the spectrogram directly.

For the approach based on landmarks, we fix the size of patches based on an audio identification experiment on a subset of the data: the results are presented in fig. 4.11. As expected, the bigger the patch size, the greater the classification accuracy. For the windows experiment, both 20×20 and 20×30 near the performance of the biggest 20×40 patches, but they fail for noise obfuscations. To avoid compromising the results, we choose the biggest examined patch size 20×40: recalling the notation from sec. 4.3, we set $2p_f = 20$ and $2p_t = 40$. We did not manage to evaluate the windows approach on the same dataset as the global one due to the associated computational cost. While the fingerprinting times shown in table 4.2 are similar to those for the global approach, storing all the fingerprints and comparing them is prohibitive due to the high number of windows. We therefore exclude this approach from the current analysis.

The performance of the filtrations relying on the distance transform for the noise obfuscations and extracting windows, is shown in fig. 4.12. It is significantly lower than that of the intensity, time and frequency filtrations on patches, shown in the bottom row of fig. 4.14. Therefore, we choose to exclude these filtrations from further analysis, and conduct the time stretching and pitch shifting experiments for the remaining three.

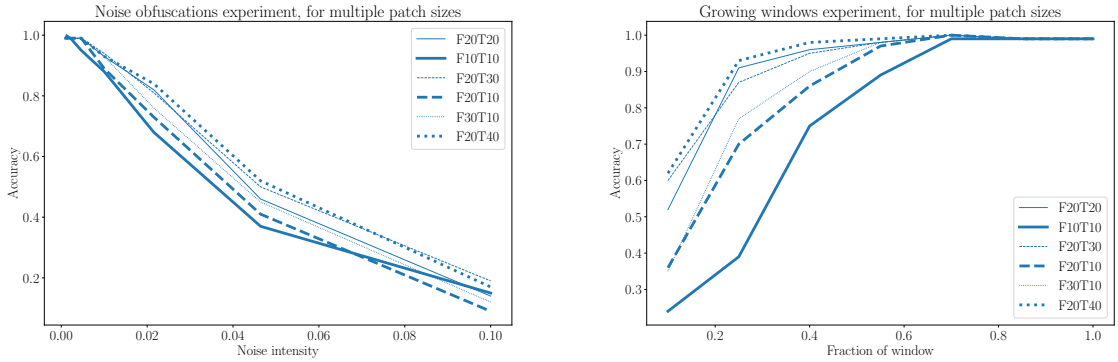

Figure 4.11: Impact of patch size on the classification accuracy as a function of obfuscations' intensity. The notation $FxTy$ stands for a patch of size $x \times y$, that spans x pixels in frequency and y pixels in time. The applied obfuscations are either noise addition (left) or window extraction (right). We use low resolution Betti curves to vectorise persistent homology induced by the frequency filtration on patches. The experiment was conducted on a subset of 100 songs.

	Intensity	Frequency	Time	Intensity level-set	Edge
Global	0.6435	2.4348	0.4860	0.9735	1.4749
Patches	0.2062	0.5221	0.5111	0.2583	0.3830
Windows	0.5461	1.7862	0.4967	0.7544	1.1747

Table 4.2: The time required to fingerprint one spectrogram, excluding the vectorisation method. That is, we measure the time it takes to compute the spectrogram, extract local features (for patches and windows), compute the associated filter functions and their persistent homology groups.

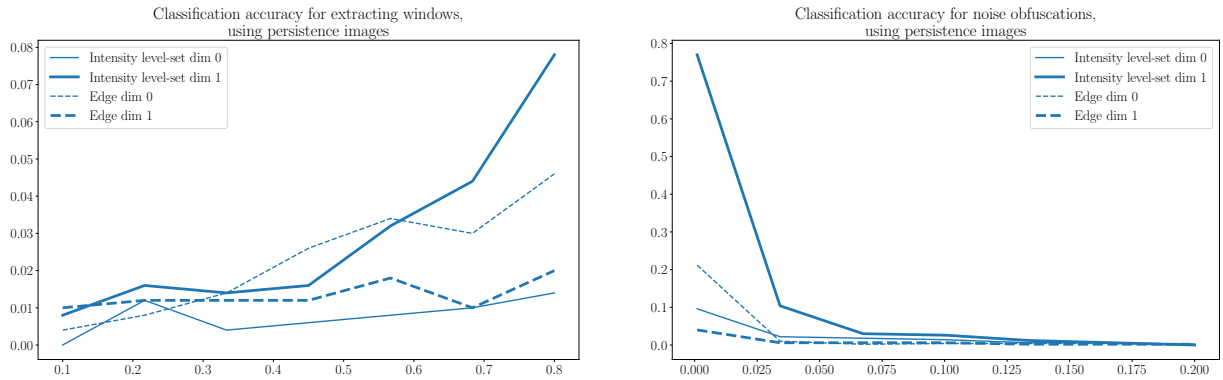

Figure 4.12: Comparison of the performance of the filtrations based on the distance transform, applied on patches and vectorised using persistence images, for two audio identification experiments.

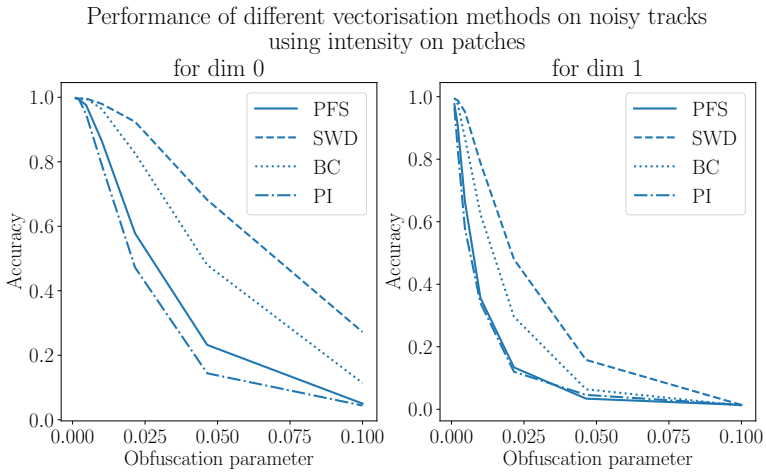

Figure 4.13: Comparison of the classification accuracy for different vectorisation methods. Fingerprinting is done using the patches approach and the intensity filtration.

We use the intensity filtration on patches to compare the four vectorisation methods described in sec. 3.5, and the results shown in fig. 4.13. For both settings, the Sliced Wasserstein distance shows the best accuracy, with the Betti curves method following behind. The persistence Fisher distance (PFD) and persistence images (PI) — both based on distances between mixtures of Gaussian kernels — perform similarly to one another, but significantly worse than the former two methods. We choose to restrict our analysis to the sliced Wasserstein distance and — as vectors of fixed dimensions are required in certain implementation of classification algorithms [Ber13] — and Betti curves methods.

We compare the global and patch approaches, for three filtrations: intensity, time and frequency, using the sliced Wasserstein distance (SWD) and Betti curves as described in sec. 4.2. We perform a series of four experiments: noise addition, window extractions, time stretching and pitch shifting. The results for the first two are shown in fig. 4.14 and in fig. 4.15 for the latter. We treat the fingerprints of each of the two dimensions separately (sec. 4.4.1), what gives us two distances and hence, two classification approaches. For the first two experiments - window extraction and noise obfuscations, the local approach outperforms the global one, as shown in fig. 4.14. The most considerable gain from using patches occurs for extracting windows: for the global approach, only the intensity filtration has an increasing classification rate, but it is still far from matching the accuracy for patches, where, we classify 90% of the songs correctly when given a 20% snippet from the song. We notice that, while for the global approach, the classification rates for dimensions zero and one are similar, they are lower for patches, especially when applying noise obfuscations. For the pitch-shifting experiment, the range of accuracies is not comparable to the values achieved for noise or windows obfuscations. The time-filtration exhibits the best results for the global approach and remains competitive for the local analysis. This is in line with the intuition that pitch shifting can be thought of as a translation of rows of the spectrogram, up or down, to which the time filtration should be the least sensitive. Similarly, we find that the classification rates for the time stretching experiment are highest for the frequency filtration in both the global and local analysis. The time filtration performs well, given that we do not expect it to exhibit invariance with respect to time stretching, but the accuracy rates it exhibits decline much quicker (as the time stretching intensifies) than those for the frequency filtration.

Since the local approach seems to outperform the global one, we extend the comparison of vectorisation methods (fig. 4.13) to other experiments only on patches. To be more specific, we perform the four obfuscations experiments, for all filtrations and vectorise the persistent homology groups using Betti curves. The resulting classification accuracies are shown in fig. 4.16. We notice that for all the experiments, the Sliced Wasserstein method performs better, especially for higher degrees of obfuscations (where the classification accuracy is the least).

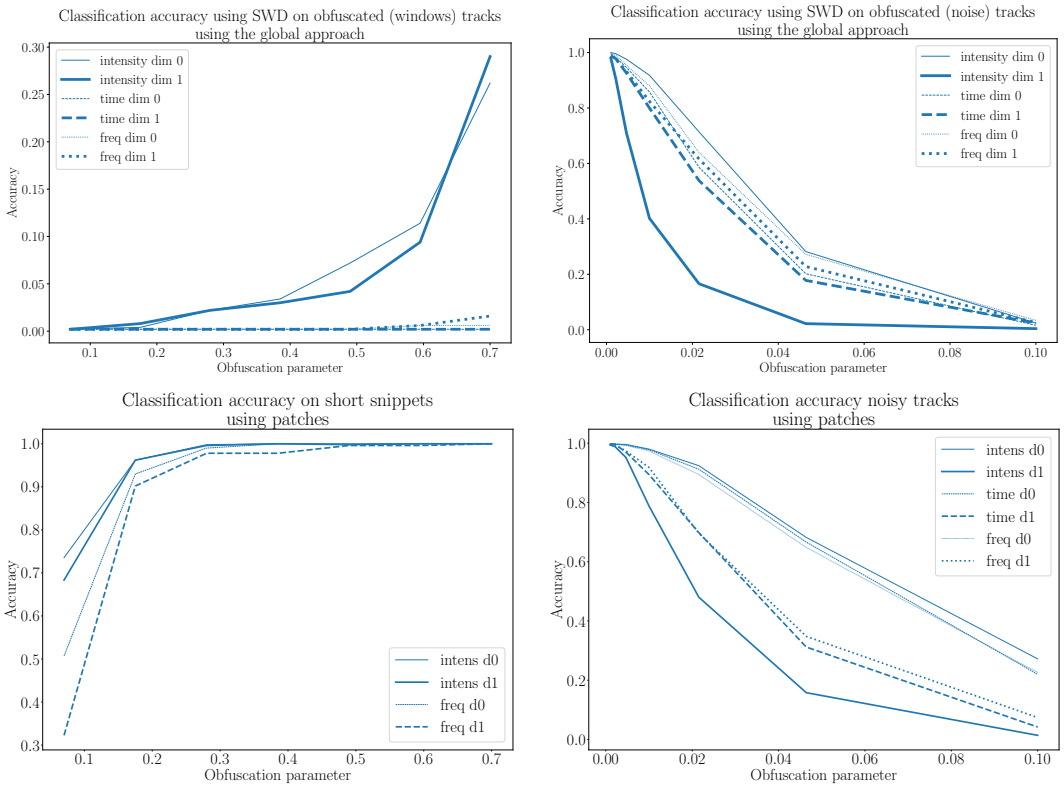

Figure 4.14: A comparison of the local and global approaches on windows and noise experiments using the Sliced-Wasserstein distance on a subset of 500 songs from the dataset. We evaluate the classification accuracy (y-axis) as a function of the obfuscation parameter - standard deviation of noise and window size respectively. As the parameter increases, we expect the classification rate for the noise obfuscation to drop, and to increase when extracting windows.

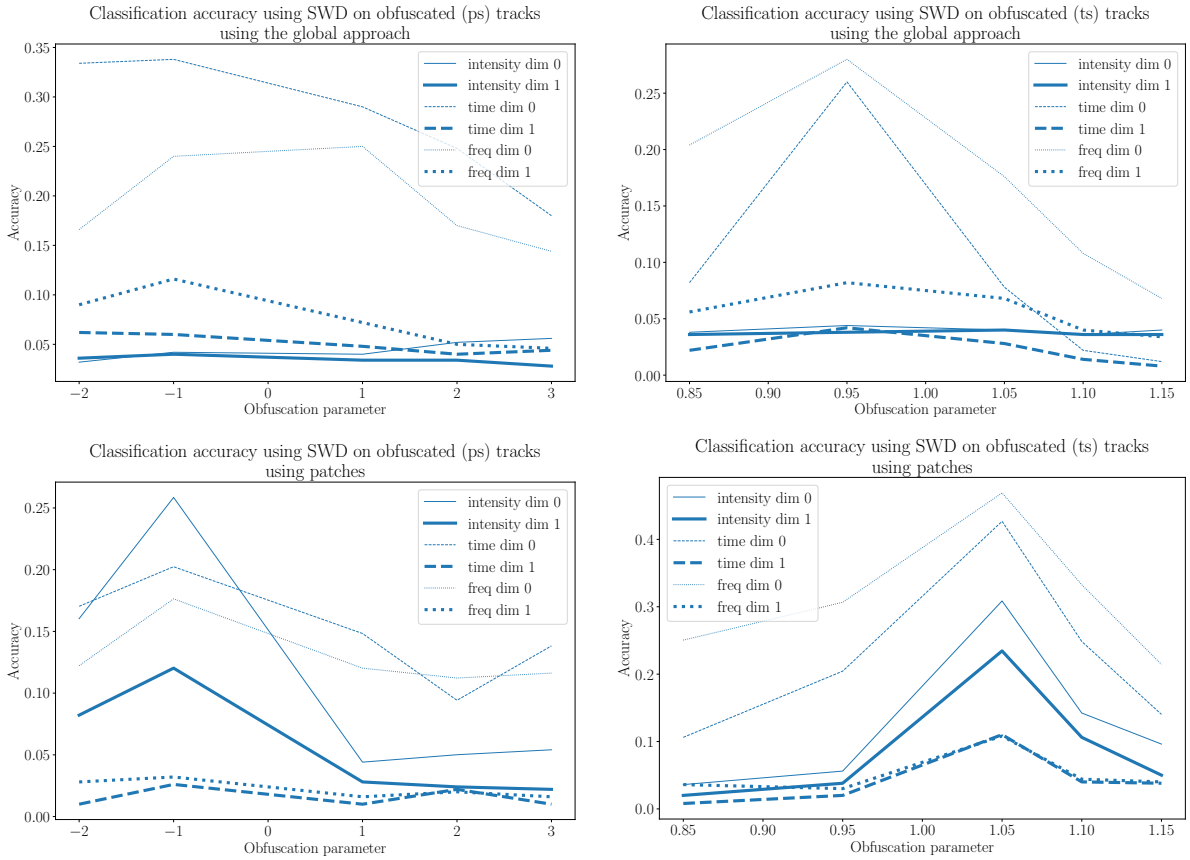

Figure 4.15: A comparison of the local and global approaches for pitch shifting (ps) and time stretching (ts) experiments using the Sliced-Wasserstein distance on a subset of 500 songs from the dataset. A pitch shift of zero corresponds to no obfuscation, hence, we expect the classification rate to be high around zero and decreasing in either direction. Similarly, time stretching with a factor of one does not change the spectrogram and hence, we expect to see high classification accuracy near that region.

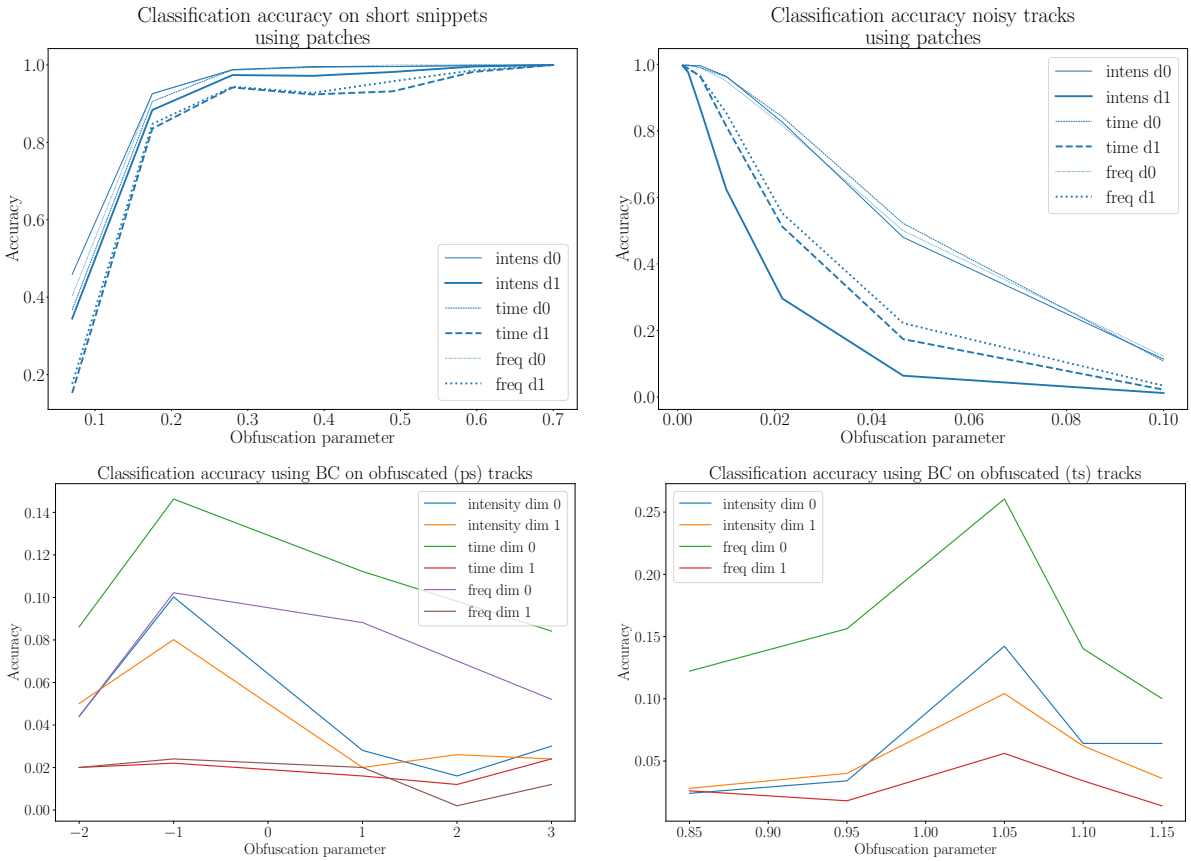

Figure 4.16: Classification accuracy, for all three filtrations on patches, using Betti curves as vector representation. The groups $H_0^\bullet$ outperform $H_1^\bullet$, for all three filtrations and all experiments.

4.6 Discussion

In this work we have presented a new audio identification framework, based exclusively on persistent homology features. Starting from spectrograms, we propose five filtrations: intensity, frequency, time, intensity level set and the edge filtration. We investigate which vectorisation and kernel methods for persistent homology are most suitable. In addition, we two approaches to analysing spectrograms: a global and a local one - based patches. We evaluate our methods for the audio identification task on a subset of the Million Song Dataset. We do so by simulating obfuscations: adding noise, extracting windows, pitch shifting and time stretching.

We find that the three filtrations we propose: the intensity, time and frequency filtrations, are suitable for audio identification. Each of them exhibits invariance to some types of transformations, allowing us to recover the label of a song for a particular obfuscation. The fingerprints for persistent homology groups of dimension zero for the intensity filtration perform best when noise is added or when windows are extracted. The results for windows extraction imply that an unrealistically significant part of the original audio is required to successfully identify the song. For noise addition, we find that persistent homology groups of both dimensions for the frequency and time filtrations achieve similar scores to the intensity filtration, while outperforming it when pitch shifting or time stretching is applied. The two filtrations based on distance did not match the performance of the other three. This might be due to two factors: first, the difficulty in choosing the number and heigh of level sets, and second, using a distance which does not include the frequency component (4.9).

The patches approach improves the performance for noise obfuscations and windows extraction. By including the frequency components of patches, we avoid false positive matches with patches that have trivial persistent homology groups. The improvement over the global approach is particularly noticeable for the latter experiment, where around 20% of the time window allows us to correctly classify 90% of the snippets. As it was the case for the global approach both persistent homology groups for the intensity filtration outperform the other two filtrations. For noise obfuscations, we find that only the classification rate for zero persistent homology groups increases, while the accuracy for the first persistent homology groups remains similar.

The experiments with different vectorisation methods lead us to conclude that the Sliced Wasserstein distance performs best in our context. One could expect persistence images to perform similarly, but we did not manage to tune the method to leverage the performance. In addition, as persistence images rely on persistence, we find that they should be less suitable for persistent homology groups of dimension one for time and frequency filtrations, as explained in sec.4.2.

Applications of persistent homology are often associated with prohibitive computational costs. Using cubical complexes, the size of our combinatorial structure is linear in the size of the spectrogram and give fingerprinting times below one second for all filtrations. With industrial scale applications in mind, we are concerned about the complexity of the current vectorisation and kernel methods that make querying slow. In particular, the local patches and windows approaches introduce more fingerprints per song, making the search of a nearest neighbour require up to 50^2 and 314^2 times more costly, for patches and windows approaches respectively.

4.6.1 Future work

The improvements of our method should be divided into two categories, depending on their aim: improving classification results and the time efficiency. We believe that the parameters (for example, the choice of level sets r) in the proposed filtrations can be optimised to provide better performance. For the time and frequency filtrations, the features obtained for each level could be interpreted jointly, for example within the framework of multi-parameter persistent homology [Vip18] or parametrised persistence. The patch parameters (the number of patches and the threshold for corner detection) should be chosen dynamically. In addition, an improvement on the classification algorithm could be inspired by Chevyrev et al. [CNO18], who witness better performance when using path signatures — the kernel framework they propose — combined with random forest classifiers.

The current querying time could be reduced by implementing kernel approximations, for example as those presented in [LY18]. However, we believe that to match the needs of industrial scale applications, one needs to develop a hashing methodology for persistence diagrams. Locally-sensitive hashing has been used in audio identification systems that we have reviewed, and in our context, would allow us to narrow the set of candidate songs that could be compared to more thoroughly.

With the insight gained in Appendix A, we believe that persistent homology could be used as a landmark identification technique: one could replace the corner detection algorithm by a topological approach, in audio

id systems.

Bibliography

[AEK+17] 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. *The Journal of Machine Learning Research*, 18(1):218–252, January 2017.

[BC07] Shumeet Baluja and Michele Covell. Audio Fingerprinting: Combining Computer Vision & Data Stream Processing. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, pages II–213–II–216, Honolulu, HI, USA, 2007. IEEE.

[BEK10] P. Bendich, H. Edelsbrunner, and M. Kerber. Computing Robustness and Persistence for Images. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1251–1260, November 2010.

[Ber13] Erik Bernhardsson. Annoy: Approximate nearest neighbors in c++/python optimized for memory usage and loading/saving to disk, 2013.

[BMEWL11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[Bro91] Judith C. Brown. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, January 1991.

[Car18] Mathieu Carriere. sklearn-tda: a scikit-learn compatible python package for machine learning and tda, 2018.

[CCO17] Mathieu Carrière, Marco Cuturi, and Steve Oudot. Sliced Wasserstein Kernel for Persistence Diagrams. *Proceedings of the 34th International Conference on Machine Learning*, 70:664–673, June 2017.

[CdSGO12] Frederic Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. The structure and stability of persistence modules. *arXiv:1207.3674*, July 2012. arXiv: 1207.3674.

[CM99] O. Cuisenaire and B. Macq. Fast and exact signed Euclidean distance transformation with linear complexity. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, pages 3293–3296 vol.6, Phoenix, AZ, USA, 1999. IEEE.

[CNO18] Ilya Chevyrev, Vidit Nanda, and Harald Oberhauser. Persistence paths and signature features in topological data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2018. arXiv: 1806.00381.

[CSEH07] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of Persistence Diagrams. *Discrete & Computational Geometry*, 37(1):103–120, January 2007.

[Dlo15] Pawel Dlotko. Cubical complex. In *GUDHI User and Reference Manual*. GUDHI Editorial Board, 2015.

[Dlo17] Pawel Dlotko. Persistence representations. In *GUDHI User and Reference Manual*. GUDHI Editorial Board, 2017.

[DP18] Vincent Divol and Wolfgang Polonik. On the choice of weight functions for linear representations of persistence diagrams. *arXiv:1807.03678*, July 2018. arXiv: 1807.03678.

[dSSVJ12] Vin de Silva, Primoz Skraba, and Mikael Vejdemo-Johansson. Topological Analysis of Recurrent Systems. page 5, 2012.

[Ell02] D. P. W. Ellis. A phase vocoder in Matlab, 2002. Web resource.

[ELZ02] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological Persistence and Simplification. page 13, 2002.

[Gut06] Emilia Gomez Gutierrez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, 2006.

[GZ18] Shafie Gholizadeh and Wlodek Zadrozny. A Short Survey of Topological Data Analysis in Time Series and Systems Analysis. *arXiv:1809.10745 [cs]*, September 2018. arXiv: 1809.10745.

[HNH+16] 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, June 2016.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*, pages 604–613, Dallas, Texas, United States, 1998. ACM Press.

[Jul11] Julius O. Smith. *Spectral Audio Signal Processing*. http://ccrma.stanford.edu/jos/sasp/, 2011. online book, 2011 edition.

[KMM11] Tomasz Kaczynski, Konstantin Michael Mischaikow, and Marian Mrozek. *Computational homology*. Springer, New York; London, 2011. OCLC: 1063425526.

[LJY16] Jen-Yu Liu, Shyh-Kang Jeng, and Yi-Hsuan Yang. Applying Topological Persistence in Convolutional Neural Network for Music Audio Signals. *arXiv:1608.07373 [cs]*, August 2016. arXiv: 1608.07373.

[LOC14] Chunyuan Li, Maks Ovsjanikov, and Frederic Chazal. Persistence-Based Structural Recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2003–2010, Columbus, OH, USA, June 2014. IEEE.

[Loz18] Dimitri Lozeve. *Topological Data Analysis of Temporal Networks*. PhD thesis, University of Oxford, Oxford, September 2018.

[LY18] Tam Le and Makoto Yamada. Persistence Fisher Kernel: A Riemannian Manifold Kernel for Persistence Diagrams. *arXiv:1802.03569 [cs, math, stat]*, February 2018. arXiv: 1802.03569.

[MG13] Neha Mehra and Surendra Gupta. Survey on Multiclass Classification Methods. *International Journal of Computer Science and Information Technologies*, 4:5, 2013.

[MMH11] Yuriy Mileyko, Sayan Mukherjee, and John Harer. Probability measures on the space of persistence diagrams. *Inverse Problems*, 27(12):124007, December 2011.

[MRL+15] Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and Music Signal Analysis in Python. pages 18–24, Austin, Texas, 2015.

[MSR+09] Vlad I. Morariu, Balaji V. Srinivasan, Vikas C Raykar, Ramani Duraiswami, and Larry S Davis. Automatic online tuning for fast Gaussian summation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1113–1120. Curran Associates, Inc., 2009.

[OHK18] Ippei Obayashi, Yasuaki Hiraoka, and Masao Kimura. Persistence diagrams with linear machine learning models. *Journal of Applied and Computational Topology*, 1(3-4):421–449, June 2018.

[OPT+17] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1), December 2017.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RD06] Edward Rosten and Tom Drummond. Machine Learning for High-Speed Corner Detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951, pages 430–443. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[RHBK14] Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A Stable Multi-Scale Kernel for Topological Machine Learning. *arXiv:1412.6821 [cs, math, stat]*, December 2014. arXiv: 1412.6821.

[RJ58] R. B. Blackman and J. W. Tukey. The measurement of power spectra from the point of view of communications engineering — Part I. *The Bell System Technical Journal*, 37(1):185–282, January 1958.

[RR07] Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. *Advances in neural information processing systems*, pages 1177–1184, 2007.

[SGB15] Ann Sizemore, Chad Giusti, and Danielle Bassett. Classification of weighted networks through mesoscale homological features. *arXiv:1512.06457*, December 2015. arXiv: 1512.06457.

[SHP17] Bernadette J. Stolz, Heather A. Harrington, and Mason A. Porter. Persistent homology of time-dependent functional networks constructed from coupled time series. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(4):047410, April 2017. arXiv: 1605.00562.

[SMR+03] Yôiti Suzuki, Volker Mellert, Utz Richter, Physikalisch-Technische Bundesanstalt, Henrik Møller, Leif Nielsen, Rhona Hellman, Kaoru Ashihara, Kenji Ozawa, and Hisashi Takeshima. Precise and Full-range Determination of Two-dimensional Equal Loudness Contours. page 11, January 2003.

[SSM+17] Nicole Sanderson, Elliott Shugerman, Samantha Molnar, James D. Meiss, and Elizabeth Bradley. Computational Topology Techniques for Characterizing Time-Series Data. *arXiv:1708.09359 [cs]*, August 2017. arXiv: 1708.09359.

[SVN37] S S Stevens, J Volkmann, and E B Newman. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, page 7, January 1937.

[TB15] Christopher J Tralie and Paul Bendich. Cover Song Identification with Timbral Shape Sequences. page 12, July 2015.

[Tra17] Christopher J Tralie. Early MFCC And HPCP Fusion for Robust Cover Song Identification. *CoRR*, abs/1707.04680:11, 2017.

[Vip18] Oliver Vipond. Multiparameter Persistence Landscapes. *arXiv:1812.09935 [math]*, December 2018. arXiv: 1812.09935.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–511–I–518, Kauai, HI, USA, 2001. IEEE Comput. Soc.

[VK16] Vanessa Robins and Katharine Turner. Principal component analysis of persistent homology rank functions with case studies of spatial point patterns, sphere packing and colloids. *Physica D*, 334:99–117, March 2016.

[Wan03] Avery Li-Chun Wang. An Industrial-Strength Audio Search Algorithm. page 7, 2003.

[YHS05] Yan Ke, D. Hoiem, and R. Sukthankar. Computer Vision for Music Identification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 597–604, San Diego, CA, USA, 2005. IEEE.

[Zom05] Afra Zomorodian. Computing Persistent Homology. *Discrete & Computational Geometry*, 33(2):15, February 2005.

[ZZJS16] Matthias Zeppelzauer, Bartosz Zieliński, Mateusz Juda, and Markus Seidl. Topological Descriptors for 3d Surface Analysis. In Alexandra Bac and Jean-Luc Mari, editors, *Computational Topology in Image Context*, volume 9667, pages 77–87. Springer International Publishing, Cham, 2016.

Appendix A

Correspondence between upper and lower star filtrations on a cubical complex

In Section 4.1, we have mentioned the existence of a duality for upper- and lower- star filtrations from the same filter function. A characterisation of this duality is provided in terms of the values of the filter function.

Let $f : \mathrm{Vert}(K) \to \mathbb{Z}$ be an injective function defined on a set of vertices of a finite Cartesian grid. We denote by K the cubical complex built on that grid. We define a filtration $K_\bullet = (K_s)_{s\in\mathbb{Z}}$ and a co-filtration $K^\bullet = (K^s)_{s\in\mathbb{Z}}$ using f

$$K_s = \{Q \text{ a cube in } K \mid \max_{v\in\mathrm{Vert}(Q)} f(v) \leq s\}, \qquad K^s = \{Q \text{ a cube in } K \mid \min_{v\in\mathrm{Vert}(Q)} f(v) \geq s\}.$$

We will denote by $C_{\bullet,k}$ (resp. $C^\bullet_k$) the k-th groups of chains and $H_{\bullet,k}$ (resp. $H^\bullet_k$) the k-th homology groups respectively. As f is injective, we can label v_s the unique vertex with f-value s. For convenience, we assume that we work with homology on $R = \mathbb{Z}_2$.

Lemma 4. *Let $(b,d) \in H_{1,\bullet}$. Then, at least two edges appear with the vertex v_b. In other words, $\#\{Q$ a 1-cube $\mid v_b = \mathrm{argmax}_{v\in Q} f(v)\} \geq 2$.*

Lemma 5. *Let $(d^\star, b^\star) \in H^\bullet_0$. Then, at least two edges appear with the vertex $v_{d^\star}$. In other words, $\#\{Q$ a 1-cube $\mid v_{d^\star} = \mathrm{argmin}_{v\in Q} f(v)\} \geq 2$.*

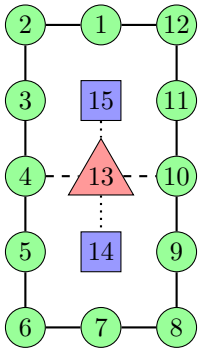

Figure A.1: A function f defined on a grid, for which there exists a point in $H_{1,\bullet} \cap H^\bullet_0$. The vertices in circles and the solid edges represent the cubical complex at K_{12}, while the rectangular ones- K^{14}. The triangular vertex is the saddle point characterised by the lemma 6. In particular, $(13, 14) \in H_{1,\bullet}$, since the v_{13} connects v_4 and v_{10}, and is filled when v_{14} appears. At the same time, v_{13} connects v_{15} with v_{13} in $K^\bullet$, so $(13, 14) \in H^\bullet_0$.

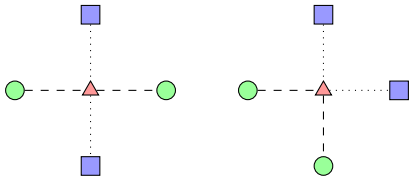

Figure A.2: Up to rotation, there are only two possible situations - the intersection between edges is either transversal (left) or tangential (right).

Proof. Let $0 < \epsilon < 1$. Then, $\iota_{d^\star, d^\star+\epsilon} : H_0^{d^\star+\epsilon} \to H_0^{d^\star}$ is not injective - there exists $b_0^\star > b^\star$ such that $\widehat{[v_{b_0^\star}]}$ and $\widehat{[v_{b_0^\star}]} = \widehat{[v_{b^\star}]}$ in $H_0^{d^\star+\epsilon}$. There exists γ a path in $K^{d^\star}$ that joins vertices $v_{b_0^\star}$ and $v_{b^\star}$. Since $\gamma \notin K^{d^\star+\epsilon}$, it passes through $v_{d^\star}$. To be more precise, if we denote by $u_1, \ldots, u_4$ the four neighbours of $v_{d^\star}$, and treat gamma as a chain in $C_1^{d^\star}$, we can write $\gamma = \gamma' + \sum_{l=1}^4 \alpha_l [u_l, v_{d^\star}]$, for some $\gamma' \in C_1^{d^\star+\epsilon}$ with $\sum_{l=1}^4 \alpha_l = 0$. Since γ passes through $v_{d^\star}$, not all are trivial. Hence, there is l_1, l_2 such that $\alpha_{l_i} \neq 0$, what implies $\widehat{[u_{l_i}, v_{d^\star}]} \in C_1^{d^\star}$, for $i = 1, 2$ and concludes the proof. □

In fact, the second Lemma state only that in order to connect two previously-disconnected components in this filtration, we need to add one vertex and at least two edges - each of the edges connecting the vertex to one of the two previously-existing components. The first lemma is similar, but referring to cycles. Given that edges are born as soon as both of their endpoints appear, the birth of a cycle at b implies that, in addition to a vertex, two edges need to appear.

Recall that, in a two-dimensional cubical complex on the grid, each vertex has at most four edges. Hence, we have established that if a vertex v_b is such that $(b, d) \in H_{1,\bullet}$ and $(b, b^\star) \in H_0^\bullet$ for some $d, b^\star \in \mathbb{Z}$, two of its neighbouring edges have f-values strictly greater than b, and two strictly less than b. The possible situations, up to rotation, are illustrated in fig. A.2 - the intersection between the edges can be either crossing or transversal.

Lemma 6. *Let $(b, d) \in H_{1,\bullet} \cap H_0^\bullet$. Then, the intersection of edges is transversal.*

Proof. A cycle c born at b dies at d. There exists a two-chain $\square_d \in C_{d,2}$, containing the vertex v_d, such that $\partial \square_d = c + c'$, where $c' \in C_{b-\epsilon,1}$. Let us consider the path given by the proof of Lemma 5. In particular, $d^\star = b$, $b^\star = d$. We see γ it as a sequence of vertices, starting at v_d and ending at $v_{b_0^\star}$ - the first vertex on the path which was born earlier than $v_{b^\star} = v_d$. As $d^\star = b$, the vertex v_b is still traversed, before reaching $v_{b_0^\star}$. Since $b_0^\star > b^\star = d$, the vertex $v_{b_0^\star} \notin K_{b_0^\star} \supseteq \square_d$ (here, we confuse the notion of algebraic and geometric cubes). In particular, this means that $\gamma \cap \square_d \neq 0$ and $\gamma \cap (\square_d)^c \neq 0$, where by $(\square_d)^c$, we mean $Q(\mathrm{Vert}(K) \setminus \mathrm{Vert}(\square_d))$ - the maximal cubical complex on the vertices not in $\square_d$. Hence, $0 \neq \gamma \cap \partial(\square_d) = \gamma \cap (c + c')$. Since $\gamma \in C_1^b$, we have $\gamma \cap c' = 0$. Hence, $\gamma \cap c \neq 0$. It is v_b and we get that the cycle and the path intersect transversally at v_b. □

Remark. Professor Ulrike Tillmann provides an alternative reasoning, based on the Jordan curve theorem. This result, applied to the cycle c, states that the set $\mathrm{Vert}(K) \setminus \mathrm{Vert}(c)$ is divided into two connected components. Given that $v_{b_0^\star}$ and $v_{b^\star}$ lie in different ones, we can infer that the path γ must cross between them as well. As the path and the cycle intersect only at v_b, its intersection is transversal.

The above lemma 6 provides a characterisation of the filter function around v_b. We could provide a characterisation of the filter at v_d as well - v_d is a local maximum for f. Note however that the characterisation of v_b is specific to points (b, d) in the intersection of the groups, as in the statement of the lemma. On the other hand, the fact that v_d is a local maximum can be deduced from $(b, d) \in H_{1,\bullet} \cup H_0^\bullet$ alone.

Remark. This result, in a more abstract setting, could be related to Intersection Theory, using Poincare duality. The proof that we provide relies on the planar setting, but the result may generalise to oriented, two-dimensional manifolds.

Remark. This result rises questions about the necessary conditions for a saddle point to give points that lie in both persistent homology groups. In particular, it is not true in general, as a saddle point does not imply the existence of a corresponding one-cycle. In addition, $(b, d) \in H_0^\bullet$ and v_b being a saddle point, do not guarantee that $(b, d) \in H_{1,\bullet}$. Similarly, for $(b, d) \in H_{1,\bullet}$ and v_b a saddle point do not imply that $(b, d) \in H_0^\bullet$.

Appendix B

Stability of spectral representations

In chap.2, we have defined spectrograms and we now examine their stability with respect to the input, in particular in the context of obfuscations that we apply.

While this would make for an inefficient implementation, computing the DFT can be seen as matrix multiplication. In this representation, the DFT matrix U is unitary, and hence, its eigenvalue are all of norm one. If we denote by $W \in \mathbb{R}^{N_{FT} \times N}$ the matrix with the window vector w on the diagonal,

$$\|UWs - UWs'\| \leq \|U\|\|W\|\|s - s'\| = \max_{i=1,\dots,N_{FT}} |w_{i,i}|\|s - s'\| = |s - s'\|.$$

The above is true regardless of the norm. The first inequality holds by definition of the norm of a linear operator, while the latter rely U being unitary and W having coefficients only on the diagonal. By repeating this for sliding windows that define the time-dimension of the spectrogram, we get

$$\|S - S'\| \coloneqq \sum_{k=1}^{N_T} \|S_{:,k} - S'_{:,k}\| \leq N_T |s - s'|. \tag{B.1}$$

An equal-loudness mel spectrogram is obtained by column-wise operations on an STFT, therefore it does not compromise the uniform continuity of the spectrogram, with respect to the input.

Regarding the obfuscations that we apply, noise addition and pitch-shifting can be expressed as $s = s' + \epsilon$, $\epsilon = (\epsilon)_{k=1}^{N}$ with small ϵ_k. However, time-stretching and window extraction both change the length of the song (and thus of the spectrogram).

Appendix C

Invariance of filtrations

In the present section, we examine the invariants of the filtrations proposed in sec. 4.1. We start by extracting a patch from a spectrogram and applying a range of transformations

1. rotation by $\pi/2$,
2. frequency inversion,
3. time inversion,
4. scaling of intensity values.

For each of the filtrations introduced in sec. 4.1: intensity (fig.C.1), frequency (fig.C.2), time (fig.C.3), intensity level-set (fig.C.4), edge (fig.C.5), we verify which invariants it respects and we summarize the results in Table 4.6. While we only picture the filtrations and persistence diagrams, we have confirmed the invariance by calculating the bottleneck distance between the corresponding persistent homology groups.

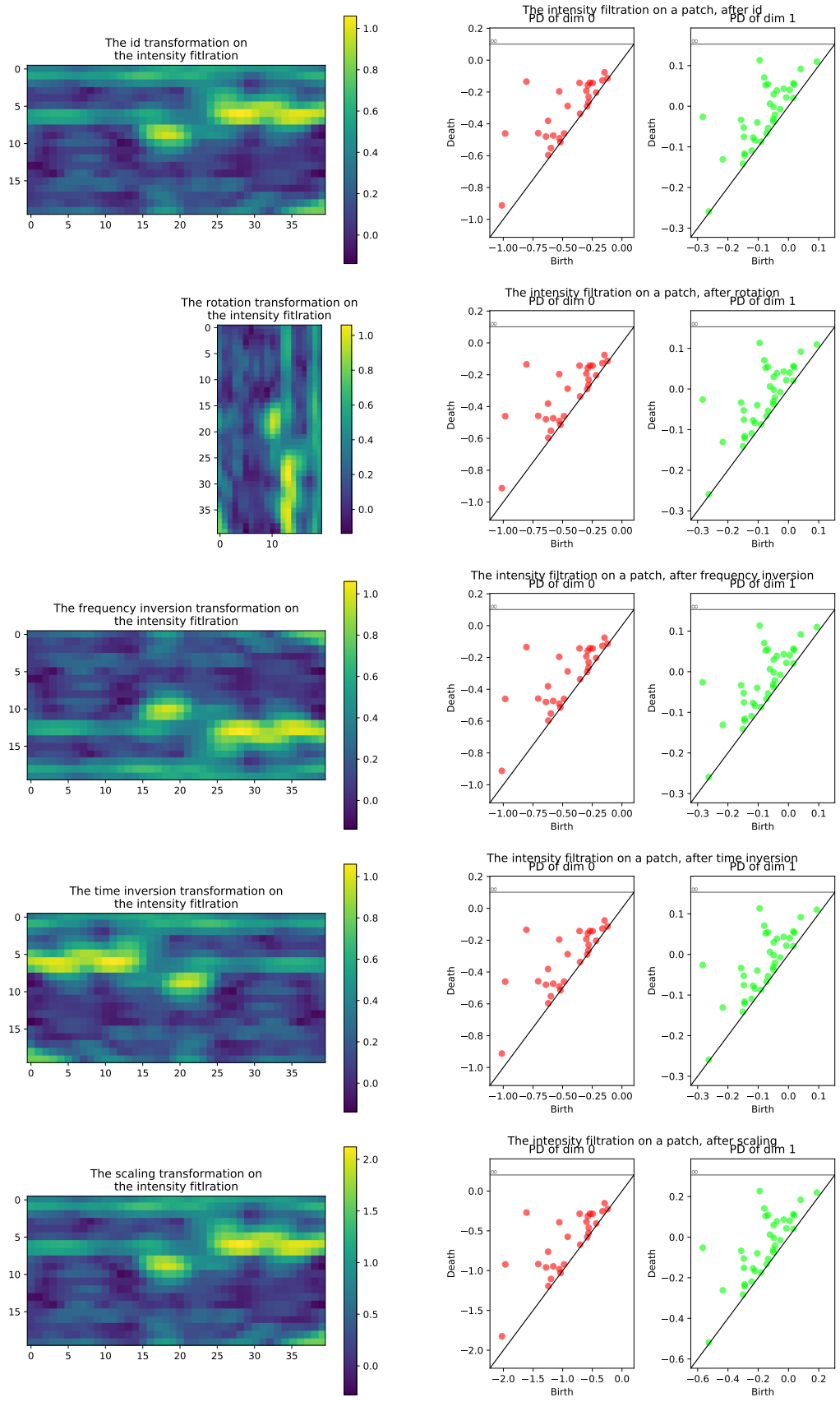

Figure C.1: Invariance of the intensity filtration on a patch to various obfuscations: rotation, frequency and time inversion, and scaling. In the top row, the filtration on the patch and on the right, its persistence diagrams. In the rows that follow, the filter functions on the transformed spectrograms and the corresponding persistent homology groups.

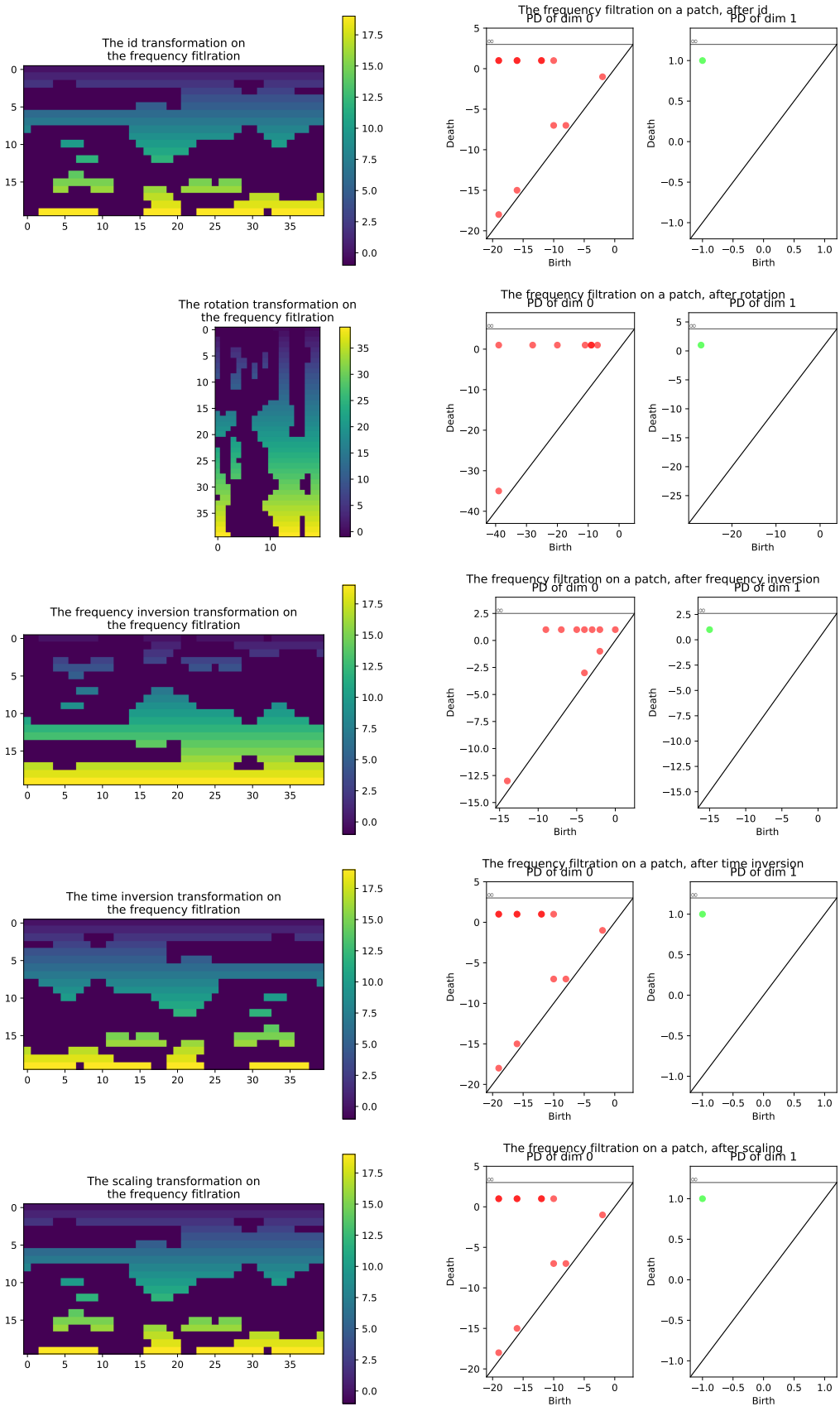

Figure C.2: Invariance of the frequency filtration on a patch to various obfuscations: rotation, frequency and time inversion, and scaling. In the top row, the filtration on the patch and on the right, its persistence diagrams. In the rows that follow, the filter functions on the transformed spectrograms and the corresponding persistent homology groups.

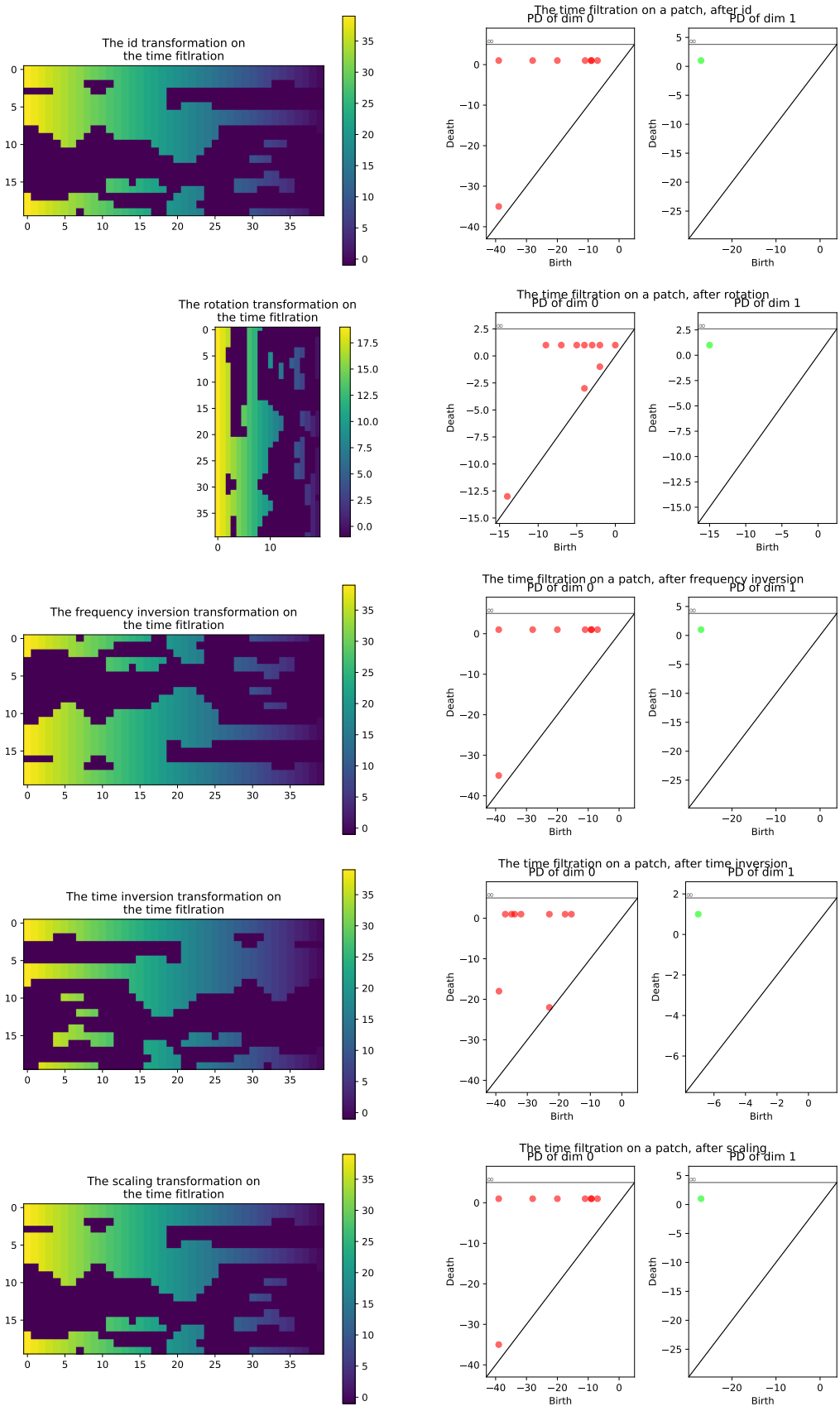

Figure C.3: Invariance of the time filtration on a patch to various obfuscations: rotation, frequency and time inversion, and scaling. In the top row, the filtration on the patch and on the right, its persistence diagrams. In the rows that follow, the filter functions on the transformed spectrograms and the corresponding persistent homology groups.

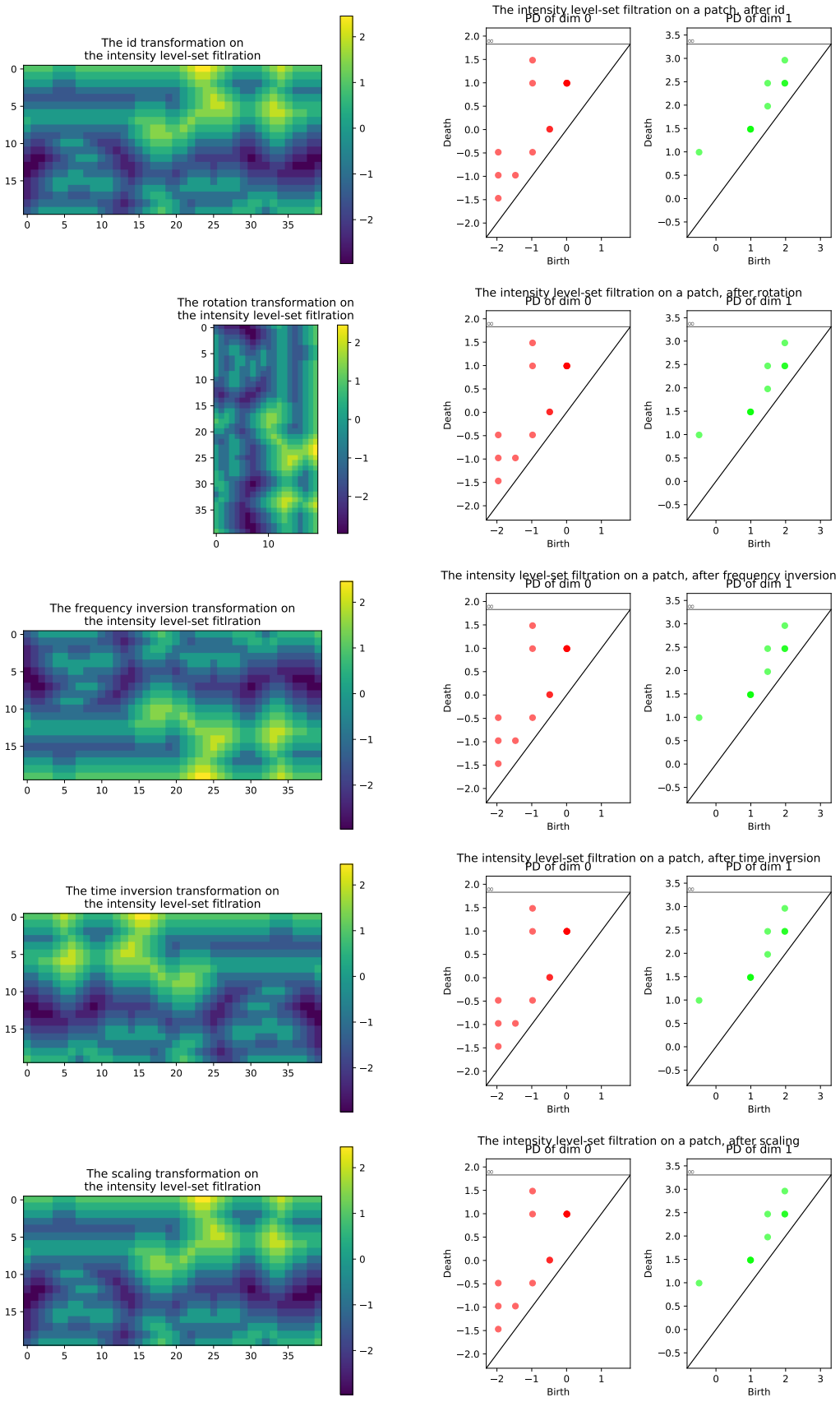

Figure C.4: Invariance of the intensity level-set filtration on a patch to various obfuscations: rotation, frequency and time inversion, and scaling. In the top row, the filtration on the patch and on the right, its persistence diagrams. In the rows that follow, the filter functions on the transformed spectrograms and the corresponding persistent homology groups.

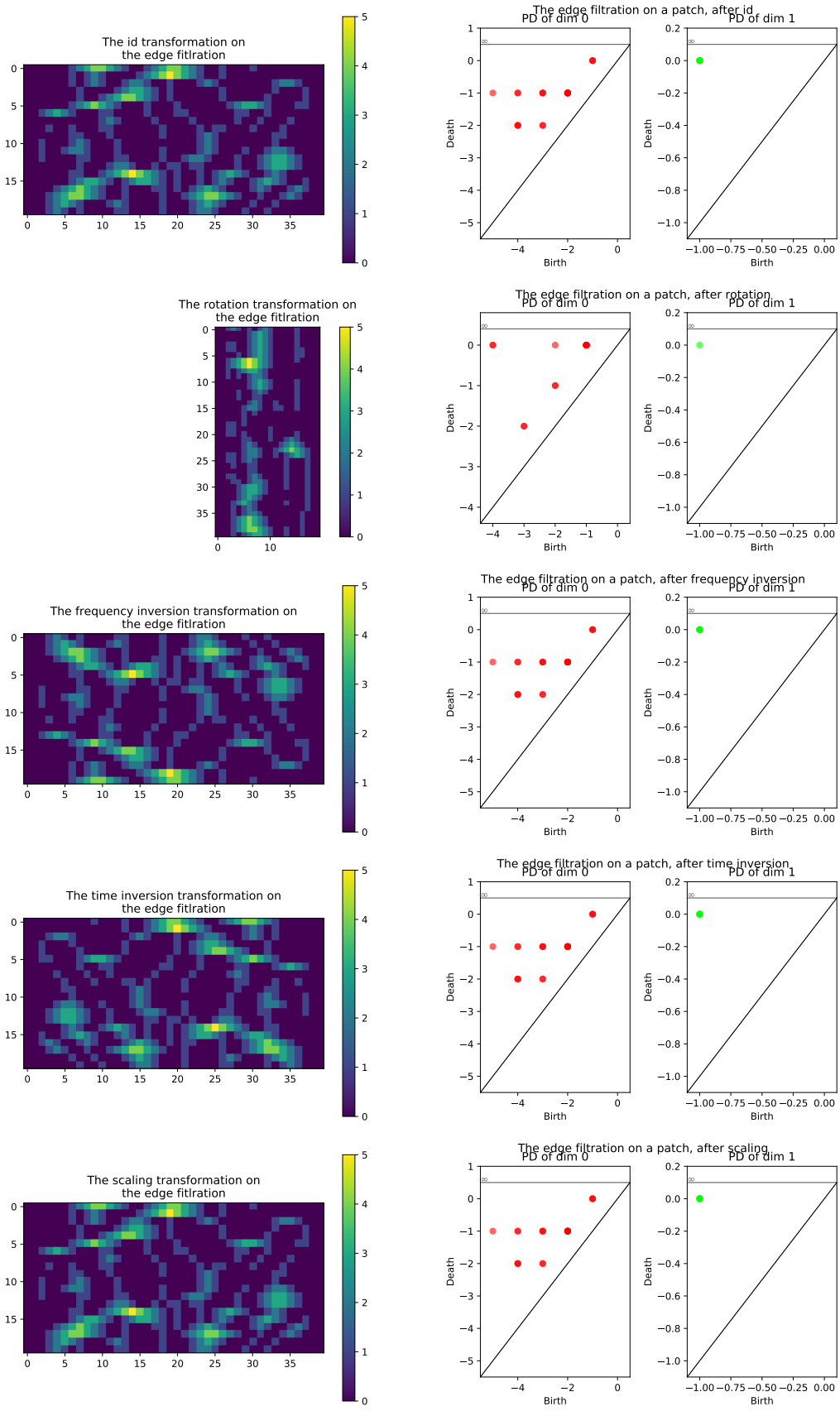

Figure C.5: Invariance of the edge filtration on a patch to various obfuscations: rotation, frequency and time inversion, and scaling. In the top row, the filtration on the patch and on the right, its persistence diagrams. In the rows that follow, the filter functions on the transformed spectrograms and the corresponding persistent homology groups.