University of Padova

Department of Information Engineering
Master's degree in Telecommunications Engineering

Master's thesis

# Digital Audio Phylogeny

## A Computer Vision Analysis on Spectrograms

Candidate:
**Sebastiano Verde**

Thesis advisor:
**Prof. Simone Milani**

Co-advisor:
**Prof. Paolo Bestagini**

**Academic Year 2016–2017**

## Abstract

The possibility of sharing multimedia contents on the Web, such as images, audio tracks and videos, results in the existence of multiple similar but not identical copies of the same document. Multimedia phylogeny deals with the analysis of dependencies among these objects, in order to lead back to the original document and make sense of the evolution of its content over time. In this thesis we propose an innovative approach to the phylogenetic analysis of digital audio tracks, where time-frequency representations of audio tracks are used to convert audio contents into images. In this way, computer vision techniques may be employed as in the framework of image phylogeny. The proposed method was validated and compared to the state-of-the-art solution, providing better results in terms of both accuracy and computational complexity. Preliminary considerations on future work suggest the application of this approach to the analysis of different kinds of signals, ranging from biomedical devices or distributed wireless sensors.

## Sommario

La possibilità di condividere in rete contenuti multimediali, quali immagini, audio e video, comporta l'esistenza di più copie simili ma non identiche dello stesso documento. La filogenesi multimediale si occupa dell'analisi delle dipendenze tra questi oggetti, con lo scopo di risalire al documento originale e ricostruire la sua evoluzione nel tempo. In questa tesi viene proposto un approccio innovativo per l'analisi filogenetica di tracce audio digitali, nel quale viene sfruttata la rappresentazione tempo-frequenza delle tracce convertendole dal dominio dell'audio a quello delle immagini. In tal modo è possibile applicare le tecniche di computer vision utilizzate nell'ambito della filogenesi delle immagini. Il metodo proposto è stato convalidato e confrontato con l'attuale soluzione di riferimento, dimostrando maggiore accuratezza e minore complessità computazionale. Considerazioni preliminari sugli sviluppi futuri suggeriscono l'applicazione di questo approccio all'analisi di diversi tipi di segnali, come quelli da dispositivi biomedici o sensori wireless distribuiti.

# Contents

# List of Figures

Chapter 1

# Introduction

Within the past two decades, the technological progress has significantly affected our life on two parallel fronts: on one hand, increasingly powerful and portable hardware (smartphone, tablet, etc.) have been made available for personal use; on the other hand, the Internet and social networks have massively and ubiquitously developed supporting a wide variety of applications. The convergence of these two factors allows users to easily upload and share a large amount of digital multimedia content everywhere, at anytime, through whatever device. To give an idea of the size of this phenomenon, statistics carried out in 2011 have revealed that the Flickr photo hosting service receives 5.2 thousand high-resolution images per minute and the YouTube video hosting service receives a staggering 24 minutes of video content per second [1]. In this scenario, it is not uncommon to assist to the emergence of *viral* contents that are disseminated all across the Web within a short time from the first upload.

In addition, the increased amateur usage of editing software implies the presence of a further phenomenon: digital documents mutate as they spread through the network. Considering images, for instance, every user has the possibility of downloading a picture from the Internet and republishing it after having altered its content. Such modifications may vary from quality enhancements for artistic purposes (brightness and contrast corrections, filtering, etc.) to the necessity of conforming to a given publication format (resizing, cropping, compression, logos insertion, etc.). The iteration of this process, together with the dissemination of documents, gives origin to a large number of *similar but not identical* copies of the same document, which are commonly known in literature as *near-duplicates* [2].

There are several situations where it is necessary to identify all the copies and/or alternate versions of a given content. A simple image search, if left unchecked, may return thousands of similar copies of the same picture. In legal environments, instead, it is crucial to identify an illegal content that has been shared on the Internet, regardless of possible modifications it could have undergone. However, while the identification of an exact copy is a trivial task, the same cannot be said of a near-duplicate. In particular, we identify two categories of problems: i) the *detection* problem, which consists in verifying whether two documents are near-duplicates of each other; ii) the *recognition* problem, which aims at finding all the near-duplicates of a given query in a large collection [3]. These two issues are commonly merged in a single field, known as *near-duplicate detection and recognition* (NDDR).

A more ambitious objective, compared to NDDR problems, is to determine the causal relationships interlying among a population of near-duplicate objects and to identify the *original* one. The idea is to determine the history of the progressive transformations which, starting from the original document, have generated its different versions. This approach draws inspiration from biology, where it is observed how living organisms inherit characteristics from their ancestors and may undergo genetic mutations resulting in the generation of new species. This progressive branching process is called *phylogeny* [4] and can be represented with a relational *tree*. In the same way of living organisms, a document obtained by a series of transformations inherits the characteristics of its predecessors and, in case it is further modified, can give rise itself to a new tree of descendants. The original document represents the *root* of the phylogenetic tree.

In the case of ancient written documents, this phenomenon is mainly due to accidental transcription errors [5]. A famous case concerns the *Book of Soyga*, a 16th century Latin manuscript which is available in two slightly different versions, both kept by the British Library. Recent studies [6] have shown that the two manuscripts present a set of errors common to both, besides those that differentiate them, suggesting that they share a common ancestor. On the other hand, in the context of digital documents, transformations are in most cases voluntary. As a consequence, the need to develop techniques that go beyond the simple recognition of a copy is undeniable, independently from the fact that modifications were done for entertainment purposes or to forge digital evidences in criminal investigations [7]. The knowledge of the history of transformations may provides clues about the original creator of the document and a better understanding of the evolution of its content over time [8]. These demands have given rise to the research field called *multimedia phylogeny*, whose name is based on the aforementioned analogy with biology since the time evolution of documents indeed forms a phylogenetic tree.

From its very first formulation [1,3], the main interest of multimedia phylogeny has so far been mainly related to images. Several studies were performed in the field of image phylogeny, which have provided a large number of solutions that tackle the problem with different kinds of approach while maintaining a common pipeline. In addition to the classic cases, some of the studies published in the literature have also considered datasets where multiple roots can be found (*phylogenetic forest reconstruction*) [9], as well as the one in which descendants inherit characteristics from multiple ancestors [10]. Similarly, a few studies were performed in the context of *video phylogeny* [11, 12]. Considering the whole range of multimedia documents, *audio phylogeny* is still rather unexplored.

In this work we propose a solution for the phylogenetic analysis of digital audio tracks, relying on a computer vision approach in a similar way to what is commonly done for images. Our idea is based on the possibility of transforming audio files into images exploiting time-frequency diagrams, also known as *spectrograms*. From the resulting images it is possible to extract visual features by which we are able to perform quantitative comparisons between pairs of audio tracks [13, 14].

Chapter 2 defines the problem of multimedia phylogeny and presents related work in the literature, ranging from image and video to audio phylogeny. Chapter 3 presents an overview on time-frequency audio analysis and motivates the choice to address audio phylogeny by using spectrograms. Chapter 4 formalizes our solution to the audio phylogeny problem. Chapter 5 presents the performed experiments and comments on the results. Finally, Chapter 6 concludes the thesis and discuss possible further improvements and future work.

Chapter 2

# Multimedia phylogeny

The phylogenetic analysis of multimedia documents, namely the study of transformations and causal relationships in a population of duplicates, has been developed over the last decade, motivated by the occurrence of a number of problems due to the continuous growth and evolution of the digital world.

Information forensics and digital security are indeed areas where phylogeny represents a major investigative tool. Every time a forensic examination has to be performed on a digital document, results may be strongly altered if we are unknowingly considering a duplicate instead of the original [15]. This motivates the importance of being able to correctly identify the source.

Copyright enforcement also benefits from multimedia phylogeny. In this context, the most commonly used strategy is *traitor tracing* [16], which involves the embedding of a signature within the original document before its dissemination. In the event the document leaks into the Internet, potentially with multiple variations of itself, we are able to detect the originally-marked one and to recover its history by checking the presence of the signature and its modification patterns. Signatures are usually implemented with *watermarking* and *fingerprinting* techniques [17]. However, there is a number of issues for which traitor tracing strategies are not always feasible: i) some transformations on the document may destroy the marking; ii) the awareness of markings leads to the adoption of attempts to circumvent them; iii) watermarks only work for documents reproduced after their embedding, leaving earlier copies unidentifiable; iv) it is not always possible to have knowledge about the ownership of the source [3]. Multimedia phylogeny overcome these problems as it relies on a *content-based* approach: documents are analysed in order to extract some significant features, which can then be exploited to compare duplicates and infer the relationships among them.

Nowadays, a widely adopted archetype of phylogenetic analysis consists of a two-steps pipeline: the computation of a dissimilarity metric, which quantifies how two objects are related, between all pairs of documents; a phylogenetic tree estimation algorithm, which reconstructs a tree structure relying on the dissimilarity values [1, 3]. Before we go into details, though, we need to start from the formal definition of a near-duplicate.

## 2.1 Near-duplicates

A near-duplicate is a transformed version of a document that keeps the original information. This is a simple and yet widely adopted definition, which makes it clear that we are not addressing documents with similar content, but altered versions of the same original source. However, establishing objectively to what extent a modified document remains recognizable is not a trivial task. For this reason, Joly *et al.* [2] formally defined near-duplicates by introducing the notion of *tolerated transformations.*

**Definition 1.** *A document $\mathcal{D}_1$ is a near-duplicate of a document $\mathcal{D}$, if $\mathcal{D}_1 = T(\mathcal{D})$, $T \in \mathcal{T}$, where $\mathcal{T}$ is a set of tolerated transformations. $\mathcal{D}$ is called the original document or the root, $\mathcal{D}_1$ is a descendant.*

A generic transformation $T$ may be a composition of several transformations in $\mathcal{T}$, for instance $\mathcal{D}_3 = T_3 \circ T_2 \circ T_1(\mathcal{D})$ is a near-duplicate of $\mathcal{D}$ as are the intermediate documents $\mathcal{D}_2 = T_2 \circ T_1(\mathcal{D})$ and $\mathcal{D}_1 = T_1(\mathcal{D})$. In these cases, sometimes is preferred to use the notation $T_{\vec{\beta}}$, where $\vec{\beta}$ is a vector of indices.

$$T_{\vec{\beta}} = T_{[\beta_1,\beta_2,\ldots,\beta_t]} = T_{\beta_t} \circ \ldots \circ T_{\beta_2} \circ T_{\beta_1}, \qquad T_{\beta_i} \in \mathcal{T}, \quad \forall i = 1 \ldots t \tag{2.1}$$

This property of the transformation composition also implies that the relationship between near-duplicates is transitive. If a document $\mathcal{D}$ has a direct duplicate $\mathcal{D}_1$ and $\mathcal{D}_1$ has a direct duplicate $\mathcal{D}_2$, then $\mathcal{D}_2$ is, in turn, a duplicate of $\mathcal{D}$. Each document $\mathcal{D}_n$ can, however, generate multiple different duplicates of itself, resulting in multiple ramifications, which is what we call a *phylogenetic tree.* All documents forming the tree are copies of the root, but they are *not* necessarily copies of each other. An example of near-duplicate tree and its related transformations is shown in Figure 2.1. As explained above, we can see that if we are in possession of a reference marked document, $\mathcal{D}_{DB}$, then we are able to trace its transformations easily. On the other hand, when no markings are available, as in the case of documents $\mathcal{D}_{1,\ldots,6}$, we can only rely on content-based methods.

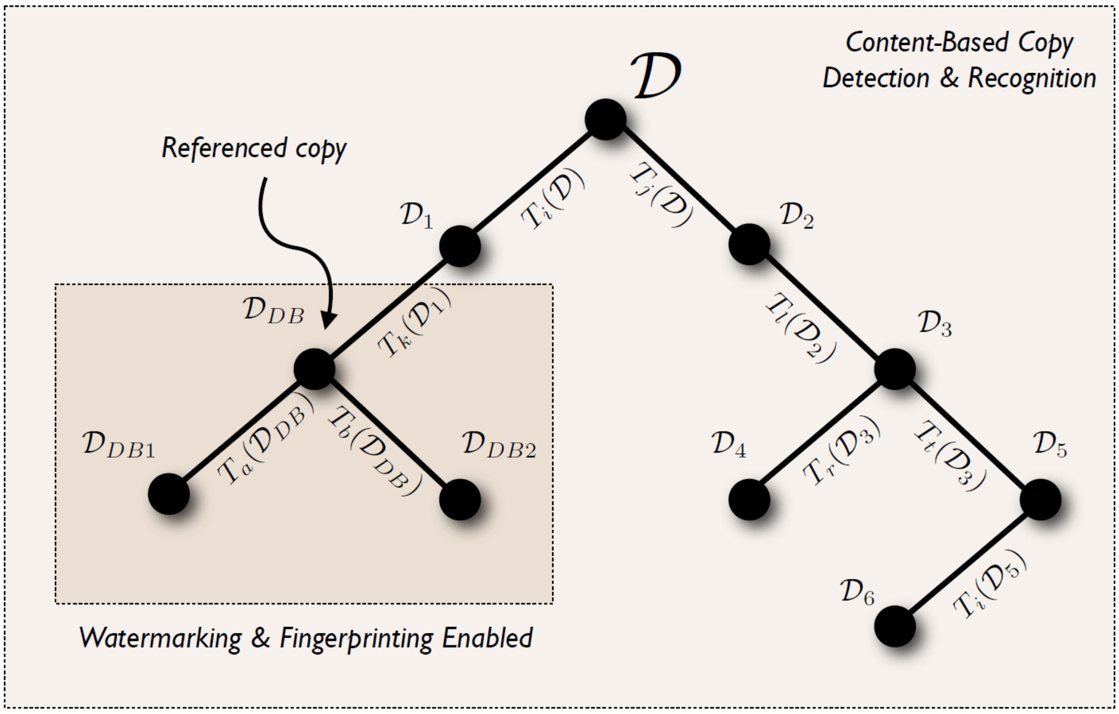
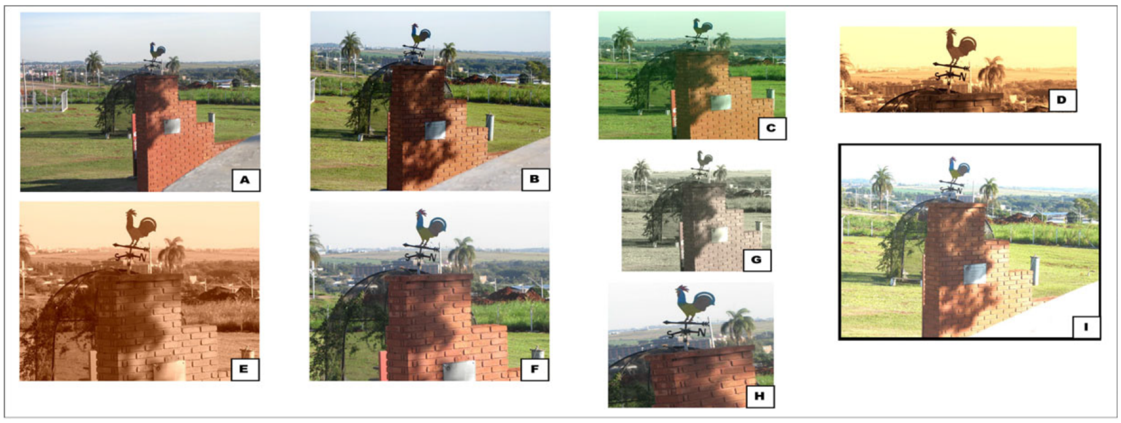


Figure 2.1: Near-duplicate tree of a document $\mathcal{D}$ and its transformations [3].

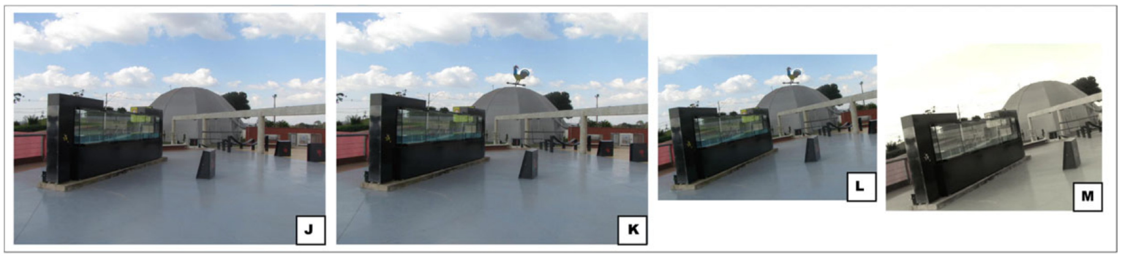
**Definition 2.** *A phylogenetic tree is a directed acyclic graph, where every node represents a document and every edge a transformation or a combination of transformations. Edges point in the direction of the transformation from the parent to its child [8].*

Phylogenetic analyses can also be extended to the scenario in which a multitude of roots is present. Let us consider, for example, a collection of images in which there is a number of pictures of the same scene, but taken with different cameras, using different settings, or from slightly different points of view. Such images are actually *different* and cannot be considered as duplicates. In fact, while sharing the same semantic content, they do not have any relationship with each other, because they do not share any common ancestor. Such documents are referred to as *semantically-similar*. A set of semantically-similar documents has $m$-independent subsets of near duplicates, each one forming a separate tree, resulting in a structure called a *phylogenetic forest* [9, 18]. An example of this scenario is shown in Figure 2.2a.

Furthermore, a situation that cannot be ruled out is the one where an object inherits content from multiple sources. For instance, image montages (the content of each source image is placed in a larger frame, with minimum or no overlapping among them), splicing (a small region of one image is pasted onto another image), blending of images (the contents of two or more images are blended through their



(a) Set of semantically-similar images. Three subsets of near-duplicates are present: i) D, E, F and H, with root F; ii) B and I, with root B; iii) A, C and G, with root A.



(b) Image composition and their near-duplicates. K consists of J with a splicing of the rooster from A. L and M are near-duplicates of K and J, respectively.

Figure 2.2: Set of related pictures [8].

alpha channels) or combinations of different videos, commonly found in the Internet [8]. An example is shown in Figure 2.2b. These scenarios represent an extension of phylogenetic forests and are referred to as *multiple parenting phylogeny* [10, 19].

## 2.2 Dissimilarity

The first step in the typical multimedia phylogeny pipeline consists of using a suitable dissimilarity function to determine how two objects are related.

**Definition 3.** *Given a set $X$ of digital objects, a dissimilarity function is a function*

$$\begin{array}{rccl} d: & X \times X & \rightarrow & \mathbb{R} \\ & (i,j) & \mapsto & d(i,j) = d_{i,j} \end{array} \tag{2.2}$$

*where $(i,j)$ is a couple of objects, $i, j \in X$, such that the more two objects are distinct, the larger the values yielded by $d_{i,j}$.*

In multimedia phylogeny, we usually take two similar objects as being likely in parent-child relationship. Conversely, two dissimilar objects are those that underwent a higher number of significant transformations, therefore occupying positions further apart in the phylogenetic tree.

In order to reconstruct the tree, first, we need to compute the pairwise dissimilarities between all objects, creating a matrix.

**Definition 4.** *Given a set $X$ of $n$ digital objects and a dissimilarity function $d(\cdot, \cdot)$, a dissimilarity matrix $M$ is an $n \times n$ real-valued matrix, such that*

$$M_{i,j} = d_{i,j}, \qquad \forall i, j \in X, \quad i \neq j \tag{2.3}$$

A dissimilarity matrix $M$ can then be taken as the adjacency matrix of a complete weighted directed graph, where every node represents an object, and the phylogenetic tree can be reconstructed by running a minimum spanning tree algorithm (see Section 2.3).

An interesting point is whether dissimilarity meets the criteria of being a distance in a metric space. Given a set $S$ and a function $f : S \times S \to \mathbb{R}$, a metric space is an ordered pair $(S, f)$ such that $\forall x, y, z \in S$, the following holds [20]:

1. $f(x, y) = 0 \iff x = y$ (identity);
2. $f(x, y) = f(y, x)$ (symmetry);
3. $f(x, y) \leq f(x, z) + f(z, y)$ (triangle inequality).

For what concerns multimedia phylogeny, however, we can see that dissimilarity function is *not* a distance. In fact, while we can reasonably expect the identity property to hold, the same cannot be said of the symmetry and triangle inequality properties. This is largely due to the fact that many digital signal processing transformations are fundamentally asymmetrical. Consider for instance the family of rescaling operations, namely upsampling and downsampling. When two images $\mathcal{I}_A$ and $\mathcal{I}_B$ have different dimensions, $d_{\mathcal{I}_A,\mathcal{I}_B}$ would calculate the metric of the residual on $\mathcal{I}_B$'s dimensions, while $d_{\mathcal{I}_B,\mathcal{I}_A}$ would calculate the metric of the residual on $\mathcal{I}_A$'s dimensions [1]. However, this asymmetry is essential for phylogenetic analyses, as it allows the reconstruction algorithms to determine the direction of the modifications (parent–child relationship) [8].

Most of the work in multimedia phylogeny involves implementing the dissimilarity function properly. In the following paragraphs we will give an overview of some of the proposed solutions, addressing the cases of image, video and audio phylogeny separately.

### 2.2.1 Image phylogeny

The most common approach to the image phylogeny problem is the one proposed by Dias *et al.* [1,3], where the dissimilarity function maps the domain of one image into that of another (image registration) and considers the cost of such operation. Given a family of image transformations $\mathcal{T}$, the dissimilarity between two images $\mathcal{I}_A$ and $\mathcal{I}_B$ is defined as

$$d_{\mathcal{I}_A,\mathcal{I}_B} = \min_{T_{\vec{\beta}} \in \mathcal{T}} \left| \mathcal{I}_B - T_{\vec{\beta}}\left(\mathcal{I}_A\right) \right|_{\mathcal{L}} \tag{2.4}$$

where $\mathcal{L}$ is any point-wise comparison metric (e.g. mean squared error). Therefore, equation (2.4) measures the amount of residual between the best transformation of $\mathcal{I}_A$ to $\mathcal{I}_B$, according to the family of operations $\mathcal{T}$, and $\mathcal{I}_B$ itself. There are countless possible transformations an image can undergo to create a near-duplicate of itself. The family $\mathcal{T}$ of transformations considered by Dias *et al.* is composed by the following operations.

- ◇ **Resampling**: the image is resized through upsampling or downsampling.
- ◇ **Cropping**: the lateral regions of an image are removed.
- ◇ **Affine warping**: the image is rotated, translated or even slightly distorted (sheared).
- ◇ **Brightness and contrast**: the colors of image pixels are adjusted through brightness and contrast operations.
- ◇ **Lossy compression**: the image is compressed using the standard lossy JPEG algorithm.

In order to estimate the transformation that best approximates image $\mathcal{I}_A$ into $\mathcal{I}_B$ we proceed in four steps.

1. Calculate the corresponding points between images $\mathcal{I}_A$ and $\mathcal{I}_B$ using the speeded-up robust features (SURF) algorithm [21]. This first step gives candidate points to estimate the resampling and cropping operations for $\mathcal{I}_A$ with respect to $\mathcal{I}_B$.
2. Robustly estimate the affine warping transformation parameters which includes translation, rotation, off-diagonal correction, and resampling operations for image $\mathcal{I}_A$ with respect to $\mathcal{I}_B$, taking the corresponding points into consideration and filtering them using Random Sampling Consensus (RANSAC) algorithm [22].

3. Calculate the mean and variance of each color channel of $\mathcal{I}_B$ and normalize color channels of $\mathcal{I}_A$ using such measures.

4. Compress the result of steps 2 and 3 according to the quantization table of $\mathcal{I}_B$, then uncompress both of them and calculate their point-wise dissimilarity using the standard mean squared error (MSE) as metric $\mathcal{L}$.

The final product of the transformation estimations for every pair of images is the dissimilarity matrix $M$, upon which we can run the tree reconstruction algorithm.

**Combined dissimilarity**

A different approach, proposed by Melloni *et al.* [23], adopts an additional dissimilarity matrix using the *content-independent* part of the image. Any image $\mathcal{I}$, in fact, can be described as the composition of two separable parts: one conveying the semantic information of the real scene, $\mathcal{I}^c$, and a mostly random content-independent part, $\mathcal{I}^r$ [24].

$$\mathcal{I} \leftrightarrow [\mathcal{I}^c, \mathcal{I}^r] \tag{2.5}$$

The content-independent part can be obtained by denoising the input image with a wavelet-based approach [25] and then subtracting the resulting image from the input one. Using both dissimilarity matrices obtained by analysing the two components separately, it is possible to reduce the ambiguity during the tree reconstruction phase in those cases where the matrix $M$ presents multiple low dissimilarity values along the same row, which means that an image is very similar to more than one of the other duplicates.

### 2.2.2 Video phylogeny

Dealing with video phylogeny reconstruction brings additional challenges compared to images. In fact, we need to deal with both spatial and temporal alignment of videos.

Dias *et al.* [11] initially proposed a simplified approach considering only temporally aligned videos of the same length and encoded with the same compression scheme. Similarly to image phylogeny, considering a family $\mathcal{T}$ of video transformations, equation (2.6) defines the dissimilarity between two videos $\mathcal{V}_A$ and $\mathcal{V}_B$ by measuring the residual between the best transformation of $\mathcal{V}_A$ to $\mathcal{V}_B$ and $\mathcal{V}_B$ itself.

$$d_{\mathcal{V}_A,\mathcal{V}_B} = \min_{T_{\vec{\beta}} \in \mathcal{T}} \left| \mathcal{V}_B - T_{\vec{\beta}}\left(\mathcal{V}_A\right) \right|_{\mathcal{L}} \tag{2.6}$$

The transformation estimation is performed by first selecting a subset of temporally coherent frames from both videos. Then, for each pair of frames, $f_A^k \in \mathcal{V}_A$ and $f_B^k \in \mathcal{V}_B$, the same four steps described for image phylogeny are followed, and the frame point-wise MSE is calculated.

A more recent approach, proposed by Costa *et al.* [26], extends the aforementioned one taking into account the case of time clipped, misaligned and compressed videos, although some constraints are imposed: a video $\mathcal{V}_A$ cannot be an ancestor of $\mathcal{V}_B$ if $|frames\,(\mathcal{V}_A)| < |frames\,(\mathcal{V}_B)|$; all duplicates were generated using the same frame rate; no multiple-parenting compositions are present.

### 2.2.3 Audio phylogeny

A possible approach to the analysis of near-duplicate audio tracks was proposed by Nucci *et al.* [27], relying on a similarity metric. Unlike the commonly used dissimilarity function, they define the similarity between two audio tracks $\mathcal{A}_A$ and $\mathcal{A}_B$, for a given family of audio transformation $\mathcal{T}$, as

$$s_{\mathcal{A}_A,\mathcal{A}_B} = \max_{T_{\vec{\beta}} \in \mathcal{T}} \mathcal{L}\left(\mathcal{A}_B, T_{\vec{\beta}}\left(\mathcal{A}_A\right)\right) \tag{2.7}$$

In this case, the comparison metric $\mathcal{L}$ is the SNR (*signal-to-noise ratio*) of the transformation of $\mathcal{A}_A$ with respect to its difference with $\mathcal{A}_B$.

$$\mathcal{L}\left(\mathcal{A}_B, T_{\vec{\beta}}\left(\mathcal{A}_A\right)\right) = 20\log_{10}\left(\frac{\|\mathcal{T}_{\vec{\beta}}(\mathcal{A}_A)\|^2}{\|\mathcal{A}_B - \mathcal{T}_{\vec{\beta}}(\mathcal{A}_A)\|^2}\right) \tag{2.8}$$

The considered family of transformations $\mathcal{T}$ is as follows.

- ◇ **Trim**: removal of the leading/trailing parts of the track.
- ◇ **Fade**: amplitude modulation of audio samples, characterised by a gradual increase/decrease in the level of the track, commonly applied to the leading (fade-in) and trailing (fade-out) ends.
- ◇ **Perceptual coding**: lossy compression of the audio track performed by taking advantage of perceptual limitations of human hearing (e.g. mp3 coding format).

The algorithm enumerates all possible transformations parametrized by a set of control values. To speed up processing the iteration is stopped whenever the calculated SNR is above a given threshold (tuned manually to 65 dB).

## 2.3 Tree reconstruction

Once the dissimilarity is calculated for all pairs of objects, the dissimilarity matrix $M$ can be seen as the adjacency matrix of a complete weighted directed graph. The problem of determining the phylogenetic tree is therefore reducible to the one of finding a directed spanning tree of minimum total weight. This problem is called *minimum spanning arborescence.*

### 2.3.1 Minimum spanning arborescence

In graph theory, an $r$-arborescence (or simply arborescence) is a directed graph in which, for a given vertex $r$ called the root, there is exactly one directed path from $r$ to $v$, $\forall v \neq r$ [28]. Equivalently, given a directed graph $D = (V, E)$ and a special root vertex $r$, an arborescence is a spanning tree (when viewed as an undirected graph) directed away from $r$ [29].

The minimum spanning $r$-arborescence is the problem of, given a directed graph $D = (V, E)$, a root vertex $r \in V$ and a cost $c_e$ for every directed edge $e \in E$, finding an $r$-arborescence in $D$ of minimum total cost.

As an integer program, the problem can be formulated as follows [29]. Letting $x_e$ be 1 for the edges of an $r$-arborescence, we have the formulation:

$$\begin{aligned}
\min \quad & \sum_{e\in E} c_e x_e \\
& \sum_{e\in\delta^-(S)} x_e \geq 1 \qquad \forall S \subseteq V \setminus \{r\} \\
& \sum_{e\in\delta^-(v)} x_e = 1 \qquad \forall v \in V \setminus \{r\} \\
& x_e \in \{0,1\} \qquad e \in E
\end{aligned} \tag{2.9}$$

In this formulation $\delta^-(S)$ represents the set of arcs $\{(u, v) \in E : u \notin S, v \in S\}$. Although it is possible to show that we can relax the integrality restrictions to $x_e \geq 0$ and also remove the equality constraints, this optimization problem still has a number of constraints that depends exponentially on the number of vertices.

However, many efficient algorithms have been proposed to solve the arborescence problem. In particular, within the framework of multimedia phylogeny, the following approaches have been tested.

- ◇ **Oriented Kruskal (OK)** [1,3]: an adaptation for directed graphs of the classic Kruskal's minimum spanning tree algorithm [30]. The OK algorithm does not require the knowledge of the root beforehand: it finds the root and determines the minimum arborescence in the same run. As its undirected version, the OK follows a greedy approach, choosing the lowest cost edge at each stage. An extension for phylogenetic forest reconstruction has also been proposed, called “automatic OK” (AOK) [9].

- ◇ **Best Prim (BP)** [31]: a heuristic based on the Prim’s minimum spanning tree algorithm [32]. It builds $n$ different directed trees, using an auxiliary method called “oriented Prim”, considering each node as a possible root once. Then, the algorithm chooses the one with the lowest total cost.

- ◇ **Optimum branching (OB)** [31]: based on the namesake algorithm proposed in different versions by Chu and Liu [33], Edmonds [34] and Bock [35]. Differently from the OK, the OB considers the whole dissimilarity matrix, always finding one optimum global solution. The OB algorithm has been as well extended to the case of forest reconstruction with the “automatic OB” (AOB) and the “extended AOB” (E-AOB) [18].

A detailed description of the OB algorithm is given below, as it has repeatedly proven to be the most effective, in addition to being the one we adopted in our work.

### 2.3.2 Optimum branching algorithm

The implementation of the algorithm in the version proposed by Edmonds [34] relies on a recursive description. Let $G_i = (V_i, E_i)$ be an input graph in a certain recursive step of the algorithm and let $r$ be the root of the tree.

First, for each node $v$ in the graph other than $r$, we select the edge arriving at $v$ with the lowest cost (Figure 2.3a). If no loops are present in the obtained graph, the algorithm stops and the tree is returned. If there is a loop, we deal with it by creating a dummy node, $v_{L_i}$, representing it. Let $L_i \subset V_i$ be the set of nodes forming the loop and $w_{\min}$ the lowest edge weight in the loop. We create a new graph $G_{i+1} = (V_{i+1}, E_{i+1})$, such that $V_{i+1} = V_i \setminus L_i \cup \{v_{L_i}\}$, i.e. $G_{i+1}$ has the same nodes of $G_i$ except for those forming the loop which are merged in the node $v_{L_i}$, and $E_{i+1}$ is defined as follows.

1. For each edge $(v_x, v_y)$ connecting a node $v_x$ outside the loop to a node $v_y$ in the loop, we add in its place the edge $(v_x, v_{L_i})$ with a weight equal to

$$w(v_x, v_{L_i}) = w(v_x, v_y) + w_{\min} - w(\cdot, v_y) \tag{2.10}$$

where $w(\cdot, v_y)$ is the weight of the edge arriving at $v_y$ in the loop.

2. For each edge $(v_y, v_x)$ connecting a node $v_y$ in the loop to a node $v_x$ outside the loop, we add in its place the edge $(v_{L_i}, v_x)$ with the weight of $(v_y, v_x)$.

$$w(v_{L_i}, v_x) = w(v_y, v_x) \tag{2.11}$$

3. Edges connecting nodes outside the loop are kept unchanged.

After processing all nodes within and outside the loop, the algorithm recursively finds the optimum branching for the new graph $G_{i+1}$ (Figure 2.3b). Next, to cope with the loop itself, the algorithm replaces the edge connecting a node $v_x$ to the dummy node $v_{L_i}$ in the current optimum solution with the correct one in the original graph, $(v_x, v_y)$. In addition, all edges within the loop that do not arrive at $v_y$ are restored, while $(\cdot, v_y)$ is removed. Finally, the algorithm updates possible edges from nodes within the loop to outside that are part of the optimum branching (Figure 2.3c).

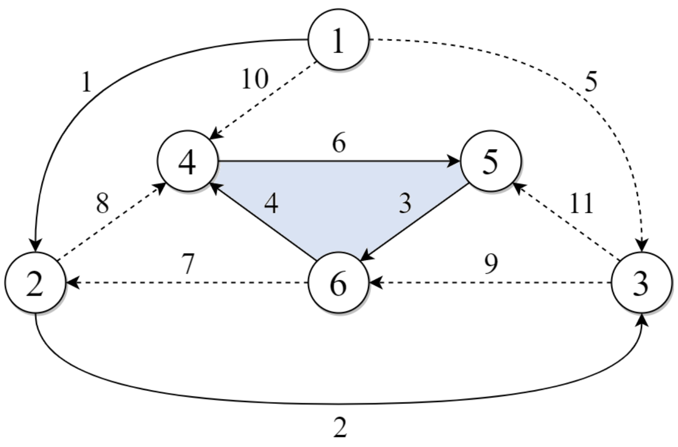
However, within the multimedia phylogeny framework, the root of the tree is unknown. In order to deal with this we have at least two possibilities. The first one is used for the BP algorithm and consists in running the algorithm $|V|$ times, each time with a different node as a possible root, and choosing the solution with the minimum cost. The second one consists in inserting a dummy node $v_0$ to be used as root and changing the edge structure so that $v_0$ can be removed later without affecting the reconstruction (i.e., ensuring the edges between $v_0$ and any other will never be selected as part of the optimum branching). In our case it is sufficient to set

$$\begin{aligned} V_0 &= V \cup \{v_0\}, \\ E_0 &= E \cup \{(v_0, v)\}, \quad \forall v \in V, \end{aligned}$$

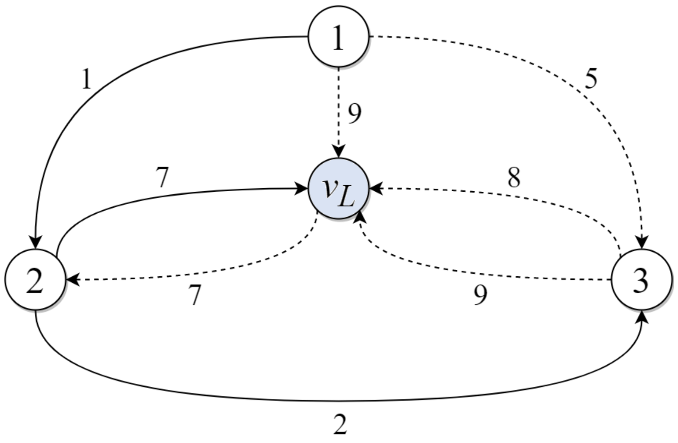
with $w(v_0, v) = K$, where $K$ is a large enough number (e.g., $K = \sum_{e \in E} w_e$), and run the algorithm using $G_0 = (V_0, E_0)$ as input and $v_0$ as root. Then, when we obtain the optimum branching $B_0$, we just need to remove the only edge $(v_0, v)$ involving $v_0$. The branching $B = B_0 \setminus \{(v_0, v)\}$ is the final optimum branching for graph $G$.

### Complexity

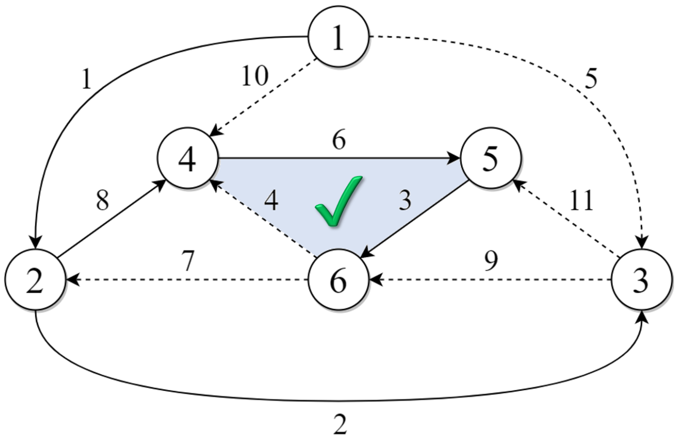
Let $G_i = (V_i, E_i)$ be an input graph at the i-th recursive call, $n_i = |V_i|$ and $m_i = |E_i|$. Finding the optimum branching $B_i$ takes time $O(n_i + m_i)$. If we are given an optimum branching $B_{i+1}$, we can obtain $B'_i$ in time $O(n_i)$. Therefore, the



(a) Find the minimum-cost edge entering each node other than 1 (solid lines). If no loops are found, return.



(b) Contract all nodes in the loop in a new node, $v_L$, and update edge costs. Recursively apply steps (a) and (b) until an arborescence is found.



(c) Uncontract $v_L$ and take all the edges of the loop but one. Repeat until there are no more edges to uncontract.

Figure 2.3: Optimum branching algorithm.

complexity of the OB algorithm is given by the recurrence

$$t(G_i) = t(G_{i+1}) + O(n_i + m_i) \tag{2.12}$$

Since the number of nodes is reduced by at least one unit at each recursive call ($|V_{i+1}| < |V_i|$) we have that

$$t(G) = O\left(n(n+m)\right) \tag{2.13}$$

Within the multimedia phylogeny framework, the input graph $G$ is complete, i.e. $m = n(n-1) = O(n^2)$, therefore

$$t(G) = O(n^3) \tag{2.14}$$

There exists a faster implementation for dense graphs, running in $O(n^2)$ [36].

Chapter 3

# Time-frequency audio analysis

In signal processing, time-frequency analysis is a generalization and refinement of Fourier analysis that provides a set of techniques used for characterizing and manipulating signals whose statistics vary in time [37], such as speech, music, images and medical signals. These techniques study the signal in time and frequency domains simultaneously. Instead of representing the signal as a one-dimensional function (where the domain is the set of real or integer numbers and the co-domain is the set of real or complex numbers), time-frequency analysis represents the signal as a two-dimensional function (where the domain is either the Cartesian plane $\mathbb{R}^2$ or a lattice) obtained via a time-frequency transform [38].

Time-frequency analysis is at the heart of our work on audio phylogeny. The possibility of mapping one-dimensional audio signals into two-dimensional visual signals allows us to study audio files and their transformations from an image processing point of view. Within this framework, many powerful tools become available, as computer vision techniques. The advantages resulting from the employment of this kind of approach are several. The detection of some specific audio features which are problematic in the audio domain (e.g. pitch modifications) becomes much more affordable in the image domain after a time-frequency transformation. In addition, feature extraction techniques adopted in computer vision allow to detect and describe some relevant keypoints in an image, thereby making it possible to analyse data by focusing only on a limited subset of features, instead of having to process the entire document. This results in lowering the required computation time dramatically.

In this chapter, we review some basic Fourier analysis definitions and notations, which are used to generate the time-frequency representations employed in the phylogenetic analysis. Then, we discuss how audio transformations are mapped into geometric transformations in the image domain, and how we can detect them.

## 3.1 Short-time Fourier transform

The *short-time Fourier transform* (STFT), or alternatively *short-term Fourier transform,* is a Fourier-related transform used to determine the sinusoidal magnitude and phase content of local sections of a signal as it changes over time. In the discrete time case, as for audio files, the procedure for computing STFTs consists of dividing a signal into segments of equal length and then computing the Fourier transform separately on each one. The fundamental building block of the discrete STFT is therefore the discrete Fourier transform.

Let $x_n$ be a discrete-time signal with sampling rate $T_s$. First, we define its *discrete-time Fourier transform* (DTFT) as

$$\mathcal{F}\{x\}(\omega) = X(\omega) = \sum_{n=-\infty}^{\infty} x_n e^{-j\omega n} \tag{3.1}$$

where $\omega = 2\pi f T_s$. The DTFT is a continuous function of frequency $f$. By sampling $X(\omega)$ at the points $f = k/NT_s$, $k = 0, \ldots, N-1$, we obtain the *discrete Fourier transform* (DFT), which is defined by the coefficients $X_k = X(k/NT_s)$ and represent the most important method of modern Fourier analysis [39].

For a discrete-time signal $x_n$ of finite length $N$, its DFT is defined as [40]

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{kn}, \qquad k = 0, \ldots, N-1 \tag{3.2}$$

where $W_N = e^{-j\frac{2\pi}{N}}$ is the $N$-th root of unity. The inverse transform (IDFT) is given by Equation (3.3).

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k W_N^{-kn}, \qquad n = 0, \ldots, N-1 \tag{3.3}$$

The DFT converts a finite sequence of samples of a signal into an equivalent-length sequence of samples of the signal DTFT, and it is therefore said to be a frequency domain representation of the input sequence.

Since the DFT deals with a finite amount of data, it can be easily implemented by numerical algorithms. Simply applying the definition in (3.2) the complexity of the implementation of the DFT is $O(N^2)$. However, modern implementations usually employ an efficient *fast Fourier transform* (FFT) algorithm [41], which manages to reduce the complexity down to $O(N \log N)$. The most common FFT algorithm is the Cooley-Tukey algorithm, which employs the *divide and conquer* paradigm recursively breaking down a $N$-long DFT into two $N/2$-long DFTs, as shown in (3.4).

$$\begin{aligned} X_k &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n}\, W_N^{2kn} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1}\, W_N^{(2n+1)k} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n}\, W_N^{2kn} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1}\, W_N^{2nk} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n}\, W_{N/2}^{kn} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1}\, W_{N/2}^{nk} \end{aligned} \tag{3.4}$$

The application of this algorithm is therefore limited to power-of-two sizes.

The short-time Fourier Transform (STFT) is usually employed whenever we need to compute the DTFT on consecutive and usually overlapping segments of the input signal $x_n$. This is performed applying a sliding window on $x_n$. The usual mathematical definition of the STFT is [42]

$$X_m(\omega) = \sum_{n=-\infty}^{\infty} x_n\, w_{n-mR}\, e^{-j\omega n} \tag{3.5}$$

where

$x_n$ = input signal at time $n$;

$w_n$ = window function (e.g. Hamming) with length $M$;

$X_m(\omega)$ = DTFT of windowed signal centred at time $mR$;

$R$ = hop size, in samples, between successive DTFTs.

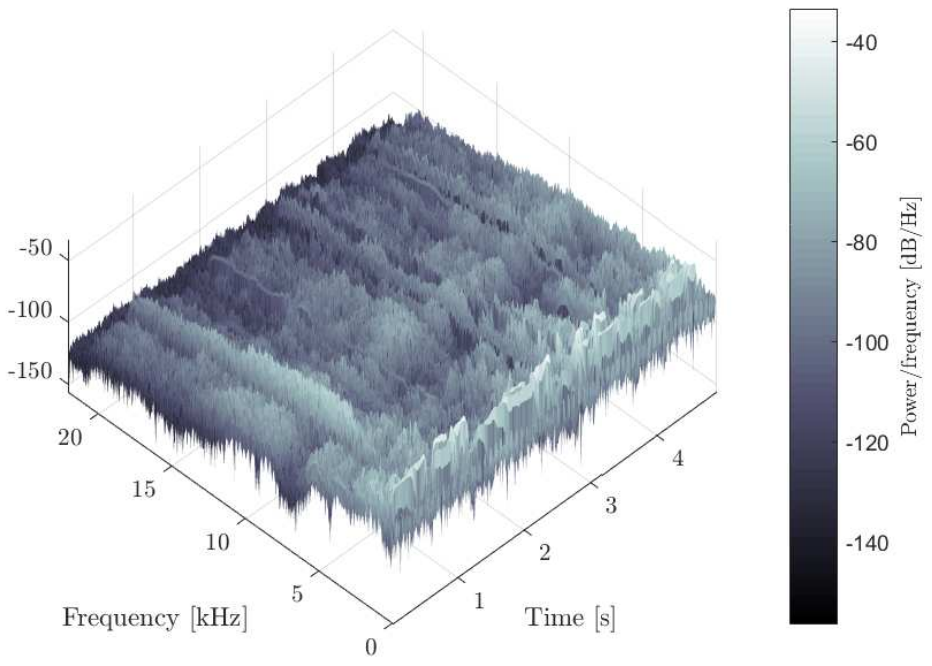
However, in practical applications the STFT is performed on a computer using the FFT, so both variables are discrete and quantized, yielding a complex-valued matrix, $X_{m,k}$, that stores magnitude and phase for each point in time and frequency.

## 3.2 Spectrograms

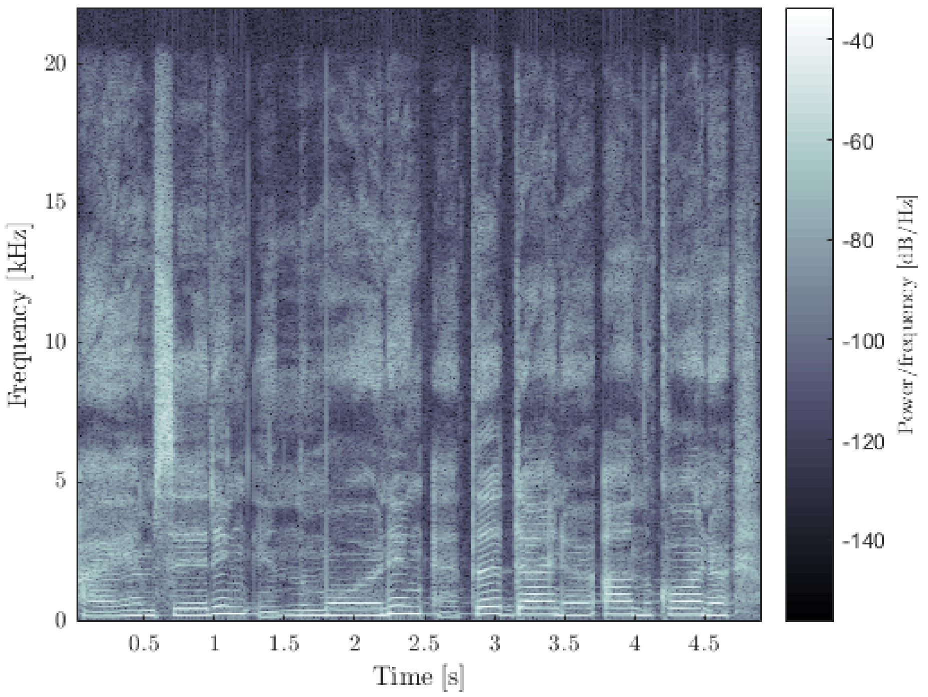
A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies over time. Let $X_{m,k}$ be the complex-valued matrix storing the discrete STFT coefficients of our input signal, $x_n$. The spectrogram $\mathcal{S}\{x\}$ can be obtained by computing the squared magnitude of the STFT.

$$\mathcal{S}\{x\}(m,k) = |X_{m,k}|^2 \tag{3.6}$$

A spectrogram is therefore a non-negative real-valued function that can be visualized as a three-dimensional surface, as shown in Figure 3.1a, where time and



(a) Three-dimensional waterfall spectrogram.



(b) Two-dimensional image spectrogram.

Figure 3.1: 3D and 2D visualization of Suzanne Vega's "Tom's Diner" spectrogram.

frequency are on the horizontal plane and magnitude on the vertical axis. This representation actually corresponds to the magnitude plots of the Fourier transform of each window laid side by side.

However, a more widely-used representation of spectrograms, which is the one we adopted for our work, is a two-dimensional image, where the horizontal axis represents time or the number of samples and the vertical axis reports frequency (in some applications the two axes are switched). The magnitude for a certain time and frequency is associated to the intensity or color of the corresponding pixel. Since we are considering audio signals, the magnitude values, as is customary, are reported in logarithmic scale.

In Figure 3.1 we can see the spectrogram of the first seconds of Suzanne Vega's "Tom's Diner", visualized in both three-dimensional (Figure 3.1a) and two-dimensional (Figure 3.1b) frames. The image representation is particularly useful for sound and music analysts, as it allows to visually inspect an audio track and highlight some specific features that would be otherwise invisible with the waveform view. For example, in Figure 3.1b we can see a high-energy spike at around 0.6 seconds, rather uniform at all frequencies. This is due to the presence of an "S" consonant in the singing, whose sound indeed resembles a white noise. In the lower part of the spectrogram (below 5 kHz) instead, we can see the harmonics of voiced sounds (vowels), visualized as parallel lines at regular frequency intervals.

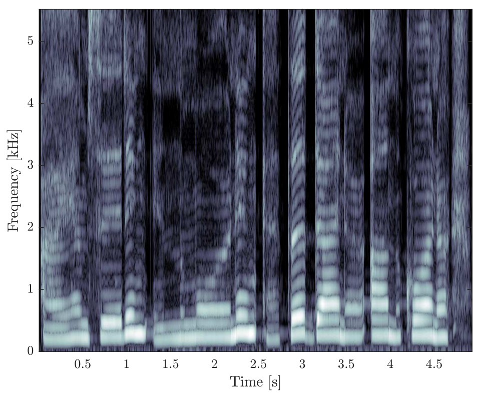
## 3.2.1 Generation parameters

When a spectrogram is created, its appearance strongly depends on a set of parameters, which must be tuned according to the needs of the specific application.

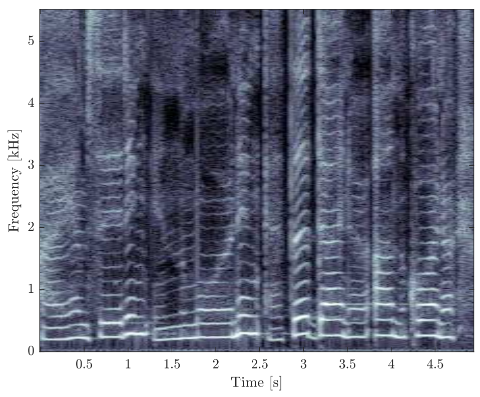
### Time windowing

The length of the sliding window employed for the STFT computation affects the time and frequency resolution of the resulting spectrogram. A wide window gives better frequency resolution but poor time resolution, and vice versa. This uncertainty is known in signal processing as the *Gabor limit* and it represents a property of signals themselves, in the same way as the Heisenberg's uncertainty principle for quantum systems. In fact, if the standard deviations of the time and frequency estimates are $\sigma_t$ and $\sigma_f$ respectively, then we can write the Gabor's uncertainty principle as follows [43].

$$\sigma_t \sigma_f \geq \frac{1}{4\pi} \tag{3.7}$$



(a) 512 samples window (12 ms).



(b) 4096 samples window (93 ms).

Figure 3.2: Effects of different time windows on spectrograms.

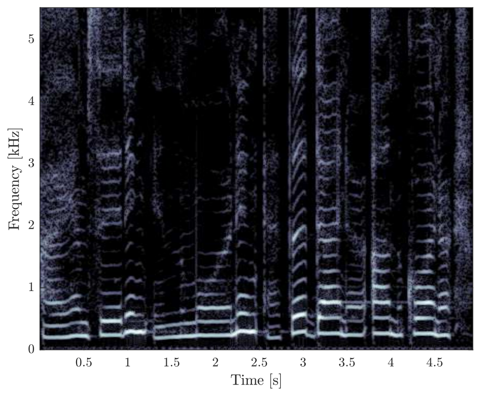
A representation of the differences between two spectrograms obtained with windows of different width, while keeping all other parameters fixed, is given in Figure 3.2. As we can see, harmonics are very clearly defined in Figure 3.2a (the one obtained with a wide window), but at the same time we lose precision on the localization of temporal features, such as note attacks and silences, which are better separated in Figure 3.2b.

In addition to the window width, the employed window function may significantly affect spectrogram appearances. Despite the best time and frequency resolution are reached with a Gaussian window function, different functions may provide better results depending on the application.

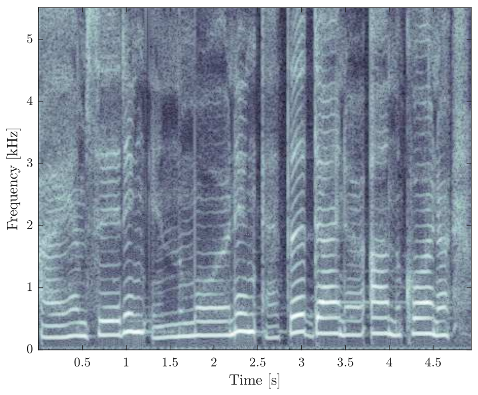
In the end, windows usually overlap each other, in order to reduce the evidence of visual artifacts due to the windowing operation. Common values for the overlap rate are around 0.5, but they can as well be either very close to zero (no overlap at all) or to one.

## Magnitude thresholding

In many practical applications, it is useful to apply a threshold on the magnitude values of a spectrogram. If an audio track is very noisy or spectrally rich, some spectral features could not be highlighted enough. This may represent an issue for visual analyses, especially if they need to compute some robust visual features (as it will be described later). Therefore, we select a minimum value of magnitude below which every pixel of the spectrogram is set to zero. We can see the effects of the thresholding operation in Figure 3.3. Only the high-energy harmonics and few other relevant features are kept in Figure 3.3a, while most of the uniform noisy background is discarded. In this example, as well as in the rest of our work, we



(a) -90 dB threshold.



(b) -150 dB threshold.

Figure 3.3: Effects of different magnitude thresholds on spectrograms.

apply the threshold to the logarithmic magnitude value of each pixel, that is

$$\mathcal{S}'(m,k) = \begin{cases} 0, & \text{if } 10\log_{10}\mathcal{S}(m,k) < \delta_L \\ \mathcal{S}(m,k), & \text{otherwise} \end{cases} \tag{3.8}$$

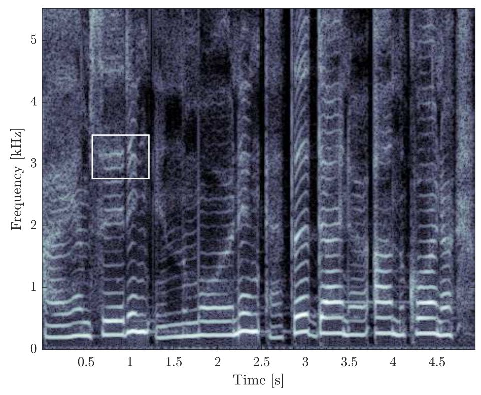
where $\delta_L$ is the selected low threshold, in dB.

### 3.2.2 Visual effects of audio transformations

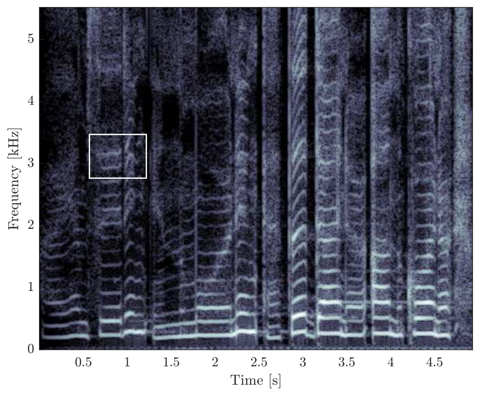
From a forensic point of view, one of the most distinctive characteristics of spectrograms is the property of mapping certain audio transformations into well-defined geometric transformations in the image domain. This is crucial for this work as it allows to shift our focus from the audio domain, where very few tools are available to identify transformations, to the framework of image geometric analysis, which can be effectively tackled by using computer vision methods. We will go into details of computer vision spectrogram analysis in Section 3.3. In this section we overview some of the most common transformations performed in the audio domain discussing the related visual effects that appears in their spectrograms.

#### Trim and fade

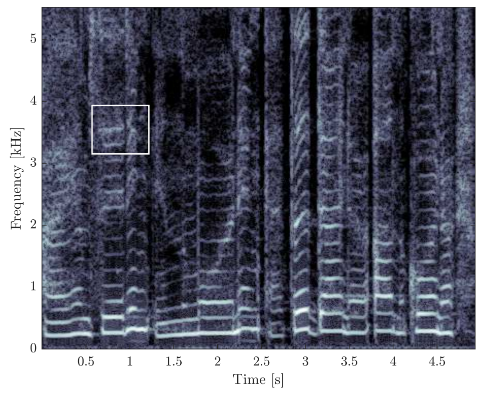
One of the most common audio transformations is *trim*, i.e. the removal of a certain number of samples from the leading and/or trailing part of the waveform, in order to extract a fragment of interest from the whole track (e.g. the chorus of a song). If an audio track is trimmed, the corresponding spectrogram results cropped, missing a set of columns in its left and/or right side. Cropping is one



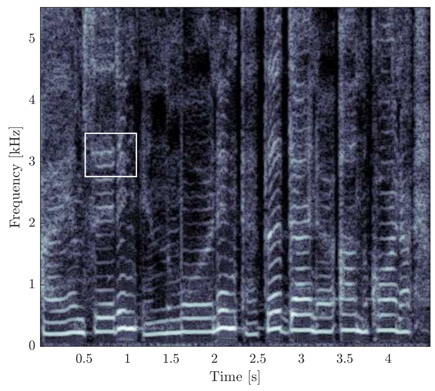
(a) Original.



(b) Fade-in (3 s).



(c) Pitch shift (two semitones up).



(d) Time stretch (10% speed up).

Figure 3.4: Geometric parallels of audio transformations.

of the typical geometric transformations considered in the framework of image phylogeny.

A somewhat similar transformation is *fade*, where the leading (trailing) samples are not completely removed but rather modulated by a given increasing (decreasing) function, creating a smooth opening (closing) effect. The spectrogram of a track where a fade effect is present has the same dimensions of the original one (no cropping) but the leftmost (rightmost) side of the image appears darker, because of the lower energy content of the faded samples (see Figure 3.4b). Most of the visual features are lost in the faded area (see Section 3.3.1).

### Time and frequency scaling

The family of time and frequency scaling transformations consists of pitch shifting and time stretching operations. These processes are commonly used to match the pitches and tempos of two tracks for mixing, or to create effects such as increasing the range of a musical instrument.

Pitch shifting is the process of changing the pitch of an audio signal without affecting the speed. The geometric transformation on the spectrogram is a scaling along the frequency axis. The scale factor can be any positive value in general, however in the field of music the smallest significant step is the *semitone* [44]. An example is shown in Figure 3.4c. The visual feature highlighted by the white box is located around 3.5 kHz, while in the original spectrogram (Figure 3.4a) it is found around 3 kHz. The feature itself is slightly stretched along the vertical axis, occupying a wider range of frequencies.

The dual operation is time stretching, which consists in changing the speed or duration of a signal without affecting its pitch. The geometric transformation on the spectrogram is itself symmetric with respect to pitch shifting, which is a scaling along the time axis. An example is shown in Figure 3.4d. The spectrogram is 10% shorter along the horizontal axis because the audio track has been speeded up, resulting in a shorter duration. The highlighted feature occupies the same frequency location as it does in the original spectrogram, but it starts earlier in time and has itself a shorter duration.

## 3.3 Computer vision spectral analysis

As we have seen, a wide range of audio transformations can be mapped into well-determined geometric transformations in the image domain. In order to exploit this property in the phylogenetic analysis, we need tools to automatically extract relevant features and estimate the geometric transformation that registers a spectrogram into another. This is where computer vision libraries come in handy.

Given a pair of spectrograms, the idea is first to apply a robust feature extraction technique that computes a set of keypoints and their associated descriptors for each of the two images. Then, we match the two point sets by comparing feature descriptors and, finally, we estimate the best geometric transformation that maps one set into the other.

### 3.3.1 Feature extraction and matching

A feature is usually defined as an distinctive part of an image, as there is no universal definition of what exactly constitutes one and it often depends on the type of problem or application we are considering. Feature extraction and matching are low-level image processing operations that are employed in many computer vision algorithms and deeply affects their performance in terms of both accuracy and

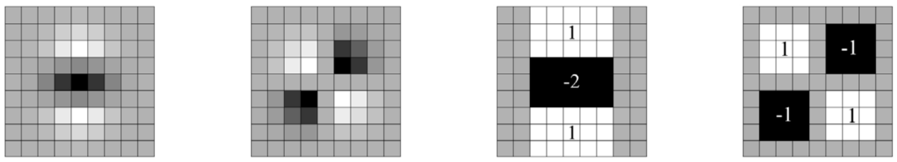


Figure 3.5: Left to right: the (sampled and cropped) Gaussian second order partial derivatives in $y$-direction and $xy$-direction, and SURF approximations thereof using box filters. The grey regions are equal to zero [21].

computational complexity. It follows that the overall algorithm will often only be as good as its feature extractor. In our work, we used *speeded up robust features* (SURF) [21] as feature detector and descriptor.

Detection is the first step of feature extraction. As a pre-requisite, the input image is usually smoothed by a certain kernel in a *scale-space representation* [45]. In fact, interest points can be found at different scales, partly because the search for correspondences often requires to compare images where they are seen at different scales. Scale-space representations are generated by repeated convolutions of an image with a smoothing kernel filter followed by sub-samplings, resulting in a structure called *image pyramid.* Gaussian filters are optimal for scale-space analysis, as shown in [46]. In practice, however, they need to be sampled and cropped. As a consequence, comparable performance can be obtained by using approximations of Gaussian kernels, allowing to reduce the computational complexity. SURF employs square-shaped (box) filters as an approximation of Gaussian filters. In fact, filtering an image with a square is a very fast operation if the *integral image* is used:

$$\mathcal{I}_{\Sigma}(x,y) = \sum_{i=0}^{x}\sum_{j=0}^{y}\mathcal{I}(i,j) \tag{3.9}$$

Features are then typically detected in terms of local image derivative operations in the scale-space representation. SURF uses a detector based on the Hessian matrix, because of its good performance in complexity and accuracy. Given a point $\boldsymbol{p} = (x, y)$ in an image $\mathcal{I}$, the Hessian matrix $H(\boldsymbol{p}, \sigma)$ in point $\boldsymbol{p}$, at scale $\sigma$ is

$$H(\boldsymbol{p},\sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{p},\sigma) & L_{xy}(\boldsymbol{p},\sigma) \\ L_{xy}(\boldsymbol{p},\sigma) & L_{yy}(\boldsymbol{p},\sigma) \end{bmatrix} \tag{3.10}$$

where $L_{xx}(\boldsymbol{p}, \sigma)$ is the convolution of the second-order derivative of Gaussian with the image $\mathcal{I}$ in point $\boldsymbol{p}$, at scale $\sigma$, and similarly for $L_{xy}(\boldsymbol{p}, \sigma)$ and $L_{yy}(\boldsymbol{p}, \sigma)$. Interest points are localised in the image and over scales by applying a non-maximum suppression to the determinant of the Hessian in a $3 \times 3 \times 3$ neighbourhood. The maxima are then interpolated in scale and image space with the method proposed

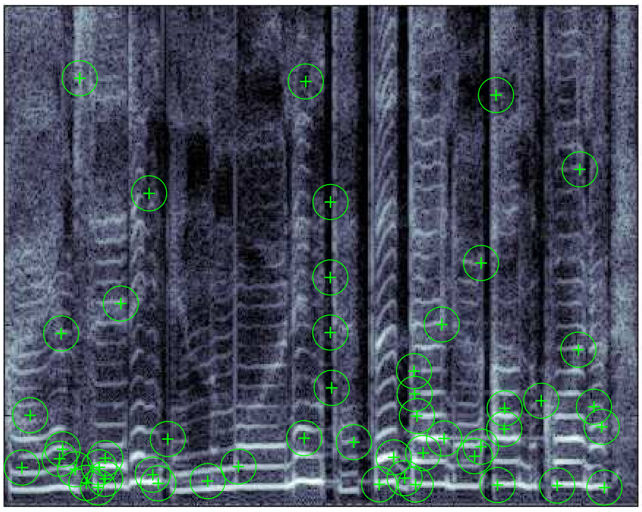


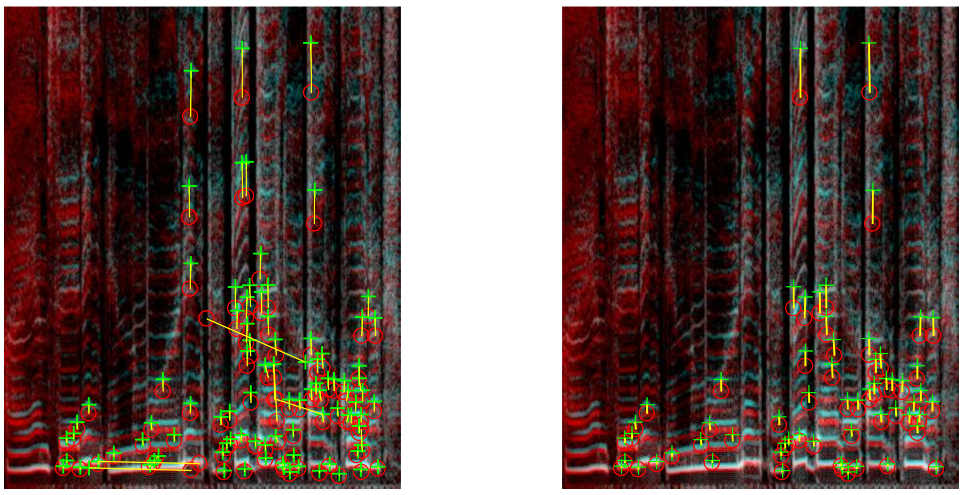
Figure 3.6: Interest points detected by SURF on a spectrogram image.

by Brown and Lowe [47].

Once features have been detected, we need to assign a descriptor for each one of them. The goal is to provide a unique and robust description of a feature, which allows us to track it over multiple images. Most descriptors are computed locally, e.g. by describing the intensity distribution of the pixels within the feature neighbourhood. SURF feature descriptor is based on the sum of the Haar wavelet response [48] around the keypoint.

Finally, by comparing descriptors obtained from two different images, matching pairs can be found. A visual representation is shown in Figure 3.7a. Two spectrograms have been superimposed: the original one and a modified version, which is pitch shifted (note the misalignment of harmonics) and faded on its leading samples. The two feature sets are represented with circle and cross marks, for the original and modified spectrogram, respectively. Note how distances between feature pairs are higher at high frequencies. This is due to the fact that pitch shifting causes a vertical rescaling on spectrogram pixels, and its effect is therefore less pronounced at low frequency. In addition, it is interesting to note how features are almost completely absent in the leftmost part of the spectrogram, where fade was applied.

However, some wrong matches may be present, as we can see in Figure 3.7a where some feature pairs are linked by non-vertical lines. Keypoints generating wrong matches are called *outliers*, and we will see how it is possible to get rid of them in Section 3.3.2, by using RANSAC algorithm.



(a) Before RANSAC. (b) After RANSAC.

Figure 3.7: Feature matching in two misaligned spectrograms. Note the absence of wrong matches after the application of RANSAC algorithm.

### 3.3.2 Geometric transformation estimation

A geometric transformation is a bijective function between sets of points, and it can be classified according to the geometric properties it preserves (see Appendix A.3). Listed in descending order of generality, we have [49]:

1. *Projectivity*, preserving collinearity;
2. *Affinity*, preserving parallelism;
3. *Similarity*, preserving angles and ratios between distances;
4. *Isometry*, preserving angles and distances.

In the case of spectrograms, one can easily conclude that the geometric transformations resulting from the mapping of audio transformations in the image domain are *affinities*. In fact, we have already seen how time and pitch scaling operations can rescale the spectrogram by different scaling factors along the two axis, which rules out similarities. On the other hand, the generality of projectivities is not required. Formally, an *affine transformation* [49] is a linear map

$$f : \mathbf{m} \mapsto H\mathbf{m} \tag{3.11}$$

where $H$ is a $3 \times 3$ non-singular matrix with the form

$$H = \begin{bmatrix} A_{2\times 2} & \boldsymbol{b} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.12}$$

and $\mathbf{m}$ is a point in *homogeneous coordinates*, which means a two-dimensional point $(x, y)$ represented as a three-dimensional vector $(x, y, 1)$ (see Appendix A.2).

However, in the specific context of spectrogram analysis, we can assume $H_{1,2} = H_{2,1} = H_{2,3} = 0$, since there are no common audio transformation that result in a shear effect or a translation along the frequency axis, in the spectrogram domain. Therefore, the transformation matrix is simplified as

$$H = \begin{bmatrix} s_T & 0 & t \\ 0 & s_P & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

where $s_T$ and $s_F$ are the time and pitch scaling factors, respectively, and $t$ represents the temporal misalignment between the two audio tracks, in number of spectrogram pixels.

Given two sets of corresponding points, the geometric transformation between them can be estimated by using a least squares optimization (see Appendix A.4). This approach is optimal when data are reliable or affected by Gaussian noise. In practice, however, the problem of outliers cannot be ignored, since the estimated transformation may be corrupted to the point of being useless, even if just a single wrong match is present. Therefore, we need a robust method that can compute the underlying geometric transformation and, at the same time, identify and remove outliers [49]. In computer vision, the most used algorithm for this purpose is called *Random Sample Consensus* (RANSAC) [22].

RANSAC is a non-deterministic algorithm, in the sense that it produces a reasonable result with an increasing probability as more iterations are allowed. A classification strategy is employed in order to distinguish *inliers* from outliers: given a transformation matrix $H$ and a threshold $\epsilon$, the pair of matched points $(\mathbf{m}_i, \mathbf{m}'_i)$ is an inlier if

$$\|\mathbf{m}'_i - H\mathbf{m}\|_2 < \epsilon \tag{3.14}$$

where $\|\cdot\|_2$ denotes the Euclidean norm.

Let us consider a set of $K$ matches, $\mathcal{M} = \{(\mathbf{m}_i, \mathbf{m}'_i) \;:\; i = 1, \ldots, K\}$. RANSAC algorithm performs the following steps.

1. Select a random subset $\mathcal{M}_0 \subset \mathcal{M}$ of $K_0 < K$ matched points.

2. Estimate a transformation $H_0$ for this subset.
3. Determine the *consensus set*, which is the set of matched points in $\mathcal{M}$ that satisfies (3.14), with respect to $H_0$.
4. Repeat steps 1–3 until the consensus set is larger than a given threshold.
5. Estimate the final model using all members of the consensus set.

In Figure 3.7b we can see the effect of RANSAC algorithm. All wrong matches have been correctly removed, together with a number of keypoints that, even though they were correctly matched, have not been included in the consensus set.

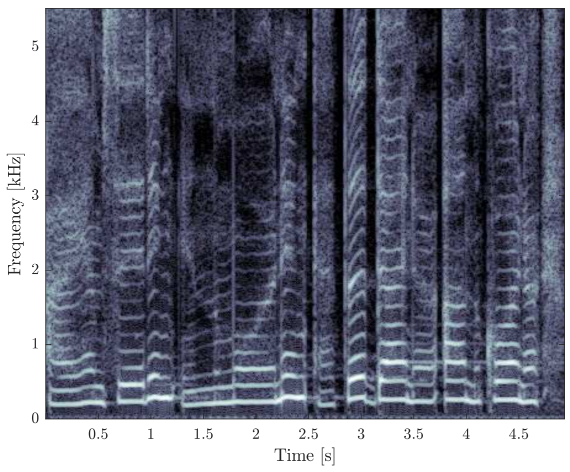
## 3.4 Non-reversible audio transformations

In conclusion to this chapter, we take in consideration some other effects of audio transformations in the spectrogram domain, which cannot be modelled with geometric transformations but are essential for our phylogeny work because of their non-reversible nature.

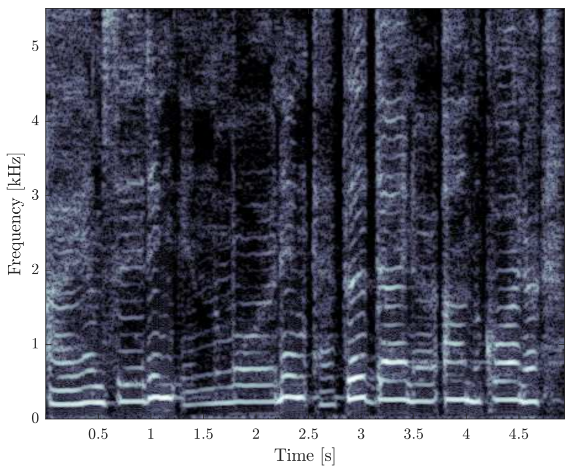
### 3.4.1 Quality decay due to scaling operations

We have seen that in the spectrogram domain time and pitch scaling operations are mapped into horizontal and vertical rescaling, respectively. If this were the only effect, then these operations would be perfectly reversible. In other words, applying a pitch shift of $p$ semitones up followed by a shift of $p$ semitones down would result in the original track; similarly, the same would happen for time stretches. In phylogenetic terms, this means that the dissimilarity is symmetric (see Section 2.2).

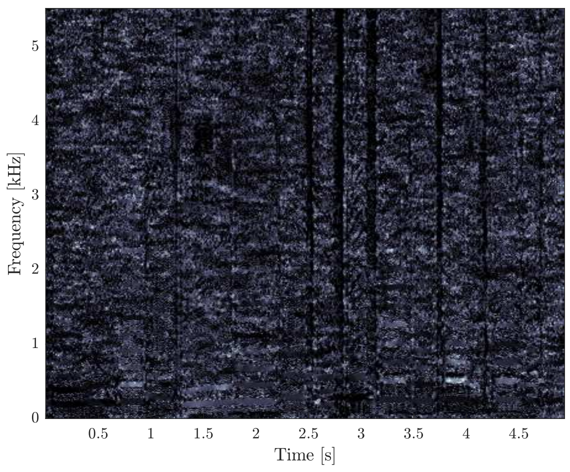
Actually, in practical situations this is not verified. These operations involve in fact a progressive decay of the track quality, as they are implemented by using intrinsically lossy operations, such as the STFT [50]. This is vital for a phylogenetic analysis, as it brings a certain asymmetry in the dissimilarity measurements which permits a correct estimation of a parent-child relationship. In Figure 3.8 we can see a comparison between an original track (Figure 3.8a) and a modified version of it, subject to two equal and opposite pitch shifts (Figure 3.8b). Note how the second spectrogram appears overall darker, due to the irreversible loss of spectral components. The effect is more evident in Figure 3.8c, where we report the absolute difference of the two spectrograms. As we can see, the residual is rather uniformly distributed all over the image, except for silences (dark vertical lines).



(a) Original.



(b) Two equal and opposite pitch shifts.



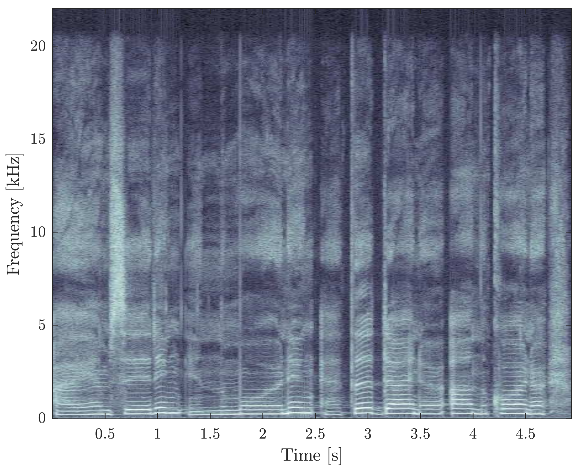
(c) Residual.

Figure 3.8: Spectral decay due to scaling operations (in this case, two equal and opposite pitch shifts).

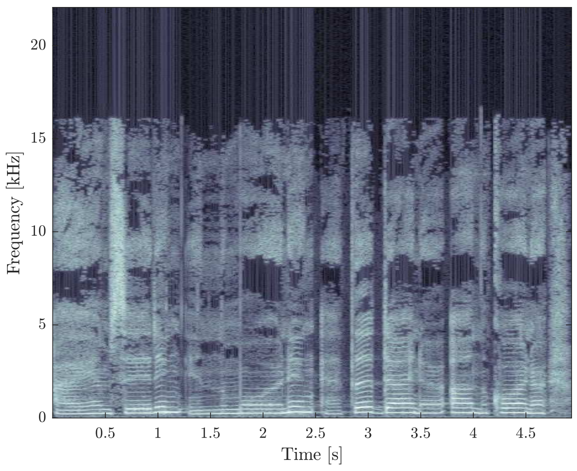
### 3.4.2 Perceptual coding artifacts

Perceptual coding is used to compress digital audio signals by removing redundant information that is not perceived in most cases by human hearing. The MPEG-1/2 Audio Layer III, more commonly referred to as MP3, is nowadays one of the most popular audio coder and therefore we must take it into consideration in an audio phylogeny framework. Once again, time-frequency representation is extremely useful since MP3 compression is designed to be perceptually almost indistinguishable from the uncompressed track, but it introduces some obvious artifacts in the spectrogram.

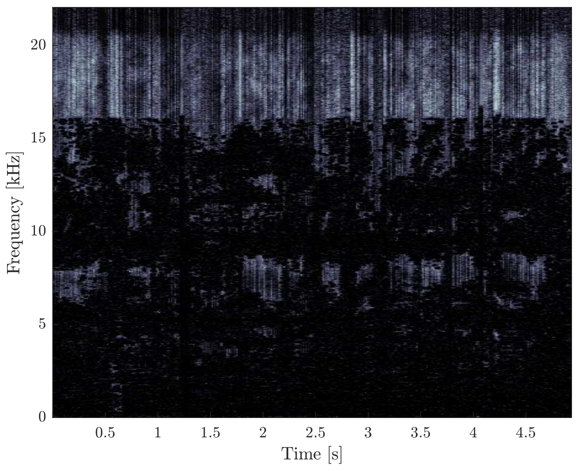
The most typical and visible effect of mp3 compression is a low-pass filtering, with a steep transition band at around 15-16 kHz. At low frequency though, the spectrogram appears rather unaltered. An example is shown in Figure 3.9. Another frequent effect we noticed is the appearance of horizontal lines, mostly at high frequency, which is probably due to the frequency quantization performed by the mp3 encoder. Even though these artifacts are often difficult to detect by visual inspection, it turned out that they can be easily identified by simple gradient-based algorithm, as we are going to discuss in Chapter 4 where we present our audio phylogeny solution.



(a) Original.



(b) MP3 compression.



(c) Residual.

Figure 3.9: MP3 compression artifacts. LAME encoder has been used, with quality factor $q = 4$.

Chapter 4

# Audio phylogeny algorithm

In this chapter we formalize our solution to the audio phylogeny problem, providing a detailed description of the devised algorithm. From now on, let us assume we are given a set of $N$ near-duplicate audio tracks, which constitute the input of the algorithm. Our goal is to output a $N$-nodes tree representation of the input set, depicting the phylogenetic relationships among the audio tracks.

We divide the algorithm into three functional blocks (Figure 4.1): the *spectrogram generation* block, in which the input track set is analysed in order to calibrate the algorithm parameters and the spectrograms are generated; the *dissimilarity computation* block, in which each pair of spectrograms is analysed in order to compute its dissimilarity value; the *graph pruning and tree reconstruction* block, in which some edges are eliminated from the dissimilarity graph in order to improve the robustness and the tree reconstruction algorithm is performed.

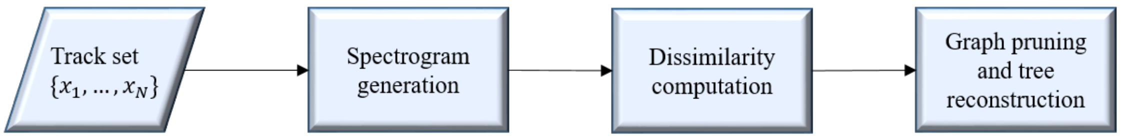

Track set
$\{x_1, \dots, x_N\}$
Spectrogram generation
Dissimilarity computation
Graph pruning and tree reconstruction

Figure 4.1: Audio phylogeny algorithm: main blocks.

## 4.1 Spectrogram generation and parameters

The first block in the algorithm pipeline performs a preprocessing of the input for the dissimilarity computation block. This consists in converting each audio track into a 2-D time-frequency representation. As we have seen in Section 3.2.1, however, the visual appearance of a spectrogram depends on its generation parameters, which must therefore be accurately tuned and constant for all the tracks in the input set. A wrong configuration would make spectrograms hardly comparable and the related dissimilarities would be meaningless. Note also that these parameters can be adapted to the input dataset, instead of being fixed.

### 4.1.1 Time windowing

For what concerns the window parameters, a fixed approach has been adopted. The window length is set to 4096 samples, corresponding to about 93 ms for audio tracks sampled at 44.1 kHz. We came to this decision by evaluating a trade-off between the number of features detected and the overall computational complexity. In fact, narrower windows result in a higher resolution along the horizontal dimension of the image, usually allowing the feature detector to identify a greater number of keypoints. On the other hand, the computational time required to process large images and additional features is proportionally higher. Moreover, having an excessively high number of keypoints available is not necessarily beneficial, since the robustness of the estimated geometric transformation may be affected. However, different window sizes may provide better results for certain kinds of input audio tracks. For instance, there might be some specific spectral features which are detectable only above a minimum resolution value. For this reason, future developments could employ an adaptive choice of the window size or even adopt a *multi-resolution* approach, performing a parallel analysis over multiple spectrogram pairs realized with different window sizes.

In Table 4.1 we report all fixed parameters used for spectrograms realization.

| Parameter | Value |
|---|---:|
| Window length | 4096 samples |
| Window function | Hamming [13] |
| Overlap | 0.75 [13] |
| Number of DFT points | 4096 |

Table 4.1: Spectrogram fixed parameters.

### 4.1.2 Magnitude thresholding

The approach we adopted about the spectrogram magnitude threshold is adaptive, in the sense that we perform a preprocessing of the input tracks in order to find the optimal parameter value. This choice is due to the fact that the threshold value does not affect the computational complexity, since the spectrogram dimension remains the same, but it determines the number of detected spectral features.

Preprocessing for threshold calibration consists in the following procedure. A set of magnitude threshold values is taken into consideration, ranging from $\tau_{L_{\min}}$ to

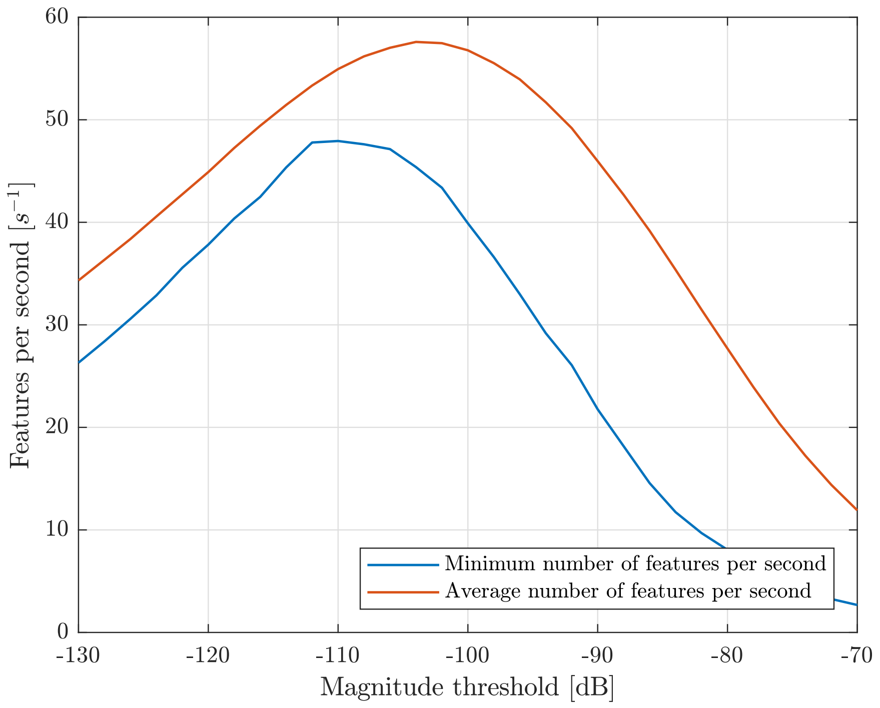


Figure 4.2: Magnitude threshold versus the number of detected features for each second of audio track.

$\tau_{L_{\max}}$ with step $\Delta\delta_L$. In our experiments, we used $\tau_{L_{\min}} = -130$ dB, $\tau_{L_{\min}} = -70$ dB and $\Delta\delta_L = 3$ dB. Then, for each $\delta_L$, spectrograms of the $N$ input tracks are generated with the current threshold value and a feature detection is performed. The number of feature detected is normalized to the track length, yielding the *feature-per-second* measure. For each threshold value, we store the average and minimum number of feature per second detected in the $N$ spectrograms. An example of results is shown in Figure 4.2, where $N = 50$ near-duplicate audio tracks have been processed. Finally, the threshold calibration function selects $\delta_L$ maximizing the number of detected features. It is possible to choose to maximise either the average or minimum value. In our experiments we chose to maximise the *minimum* value, since we prefer to guarantee the highest possible number of features detected in the worst case. However, as we can see in Figure 4.2, the two approaches provide similar results for the threshold value.

### 4.1.3 Spectrogram generation

Once the minimum magnitude threshold is set, the algorithm is ready to generate the time-frequency representations. Given a track $x$ from the input set[1], the related

[1]If an input track is stereo, then it is first converted to mono.

spectrogram $\mathcal{S}$ is computed through the STFT as in Equation (3.6). The result is then expressed in logarithmic scale.

$$\mathcal{S}_{\text{dB}} = 10 \log_{10} (\mathcal{S}) \tag{4.1}$$

At this point, the minimum magnitude threshold $\delta_L$ is applied. Since we want to format spectrograms as gray-scale images, though, we also need a *maximum* threshold, $\delta_H$. If its value is sufficiently high (0 dB in our experiments), however, the number of saturated pixel is usually very low and feature detection is not affected.

$$\mathcal{S}'(u,v) = \begin{cases} \delta_L, & \text{if } \mathcal{S}(u,v) < \delta_L \\ \delta_H, & \text{if } \mathcal{S}(u,v) > \delta_H \\ \mathcal{S}(u,v), & \text{otherwise} \end{cases} \tag{4.2}$$

Finally, the image $\mathcal{I}$ is generated as

$$\mathcal{I}(u,v) = \left\lfloor \frac{\mathcal{S}'(u,v) - \delta_L}{\delta_H - \delta_L} \cdot \left(2^b - 1\right) \right\rfloor \tag{4.3}$$

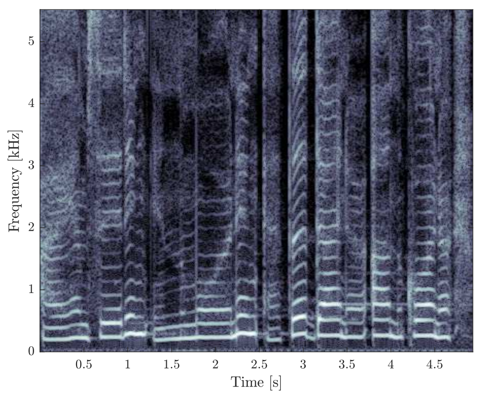
where $b$ is the number of bits per pixel (8 in our experiments). Once the $N$ spectrogram images have been generated, every image pair $(\mathcal{I}_i, \mathcal{I}_j)$, $i = 1, \ldots, N$, $j = 1, \ldots, N$, $i \neq j$, is passed to the dissimilarity computation block.

## 4.2 Dissimilarity computation

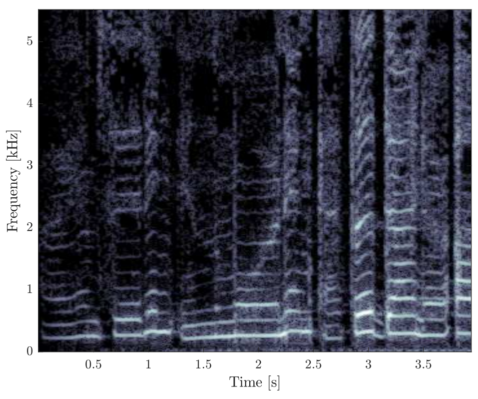
Given a spectrogram pair, $(\mathcal{I}_i, \mathcal{I}_j)$, the dissimilarity block has the task of computing the related dissimilarity value, $d_{i,j}$, which is then memorized in an $N \times N$ dissimilarity matrix, $M$. This is done by estimating the best transformation that maps $\mathcal{I}_i$ into $\mathcal{I}_j$'s domain and evaluating the difference between $\mathcal{I}_j$ and the transformed version of $\mathcal{I}_i$, as discussed in Section 2.2.1.

Let us take, for example, the spectrogram pair reported in Figure 4.3. In this case, $\mathcal{I}_i$ (Figure 4.3a) is the spectrogram of the original track and $\mathcal{I}_j$ (Figure 4.3b) is a near-duplicate, obtained by MP3 compression, pitch shifting, fade-in and trim operations.

First, the algorithm extracts keypoints from both images, $(\mathcal{K}_i, \mathcal{K}_j)$, and matches them by using the related descriptors (Section 3.3.1). The result is shown in Figure 4.4a, where the spectrogram aspect ratios have been changed in order to better visualize matches. Then, the geometric transformation mapping keypoints from $\mathcal{I}_i$ to $\mathcal{I}_j$ is robustly estimated by using the RANSAC algorithm (Section 3.3.2),



(a) $\mathcal{I}_i$, original.



(b) $\mathcal{I}_j$, near-duplicate.

Figure 4.3: Spectrogram pair in input to the dissimilarity block. $\mathcal{I}_i$ is the original one, $\mathcal{I}_j$ is a near-duplicate (compression, pitch shift, fade-in, trim).

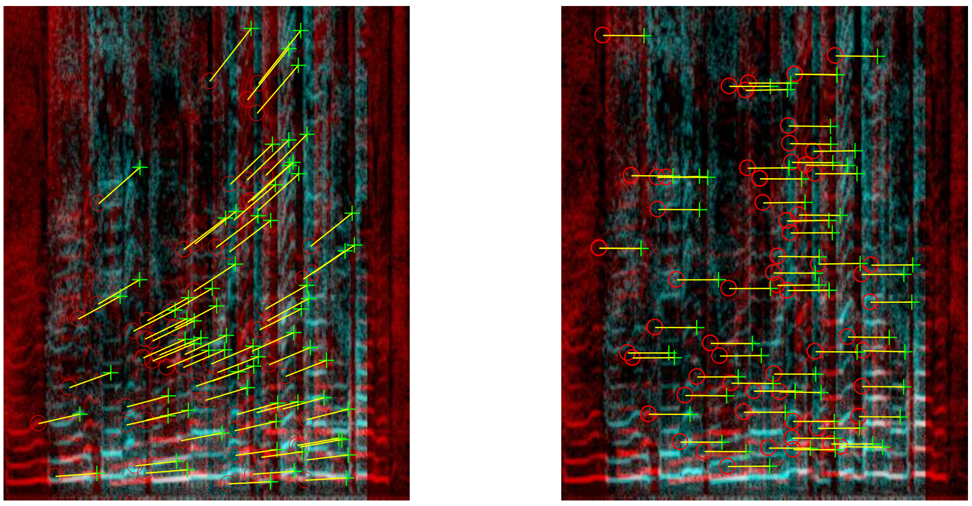
obtaining the transformation matrix $H$ defined in (3.13).

At this point, the algorithm observes the estimated values of $s_T$ and $s_P$. If at least one of them is different from one (with a suitable threshold) then a time or pitch scaling is detected. If that happens, the algorithm applies to the audio track related to spectrogram $\mathcal{I}_i$ the scaling transformation specified by $s_T$ and $s_P$, then it recomputes the spectrogram. We call this operation *scale compensation* and indicate the new spectrogram as $\mathcal{I}_i^{(s)}$. The reasons for this compensation are discussed in Section 3.4. In fact, if we simply register $\mathcal{I}_i$ to $\mathcal{I}_j$ according to the estimated transformation matrix $H$, the dissimilarity contribution due to the only time and pitch scaling operations will be symmetric ($d_{i,j} = d_{j,i}$) because, for instance, a pitch shift $p$ semitones up from $i$ to $j$ would be undistinguishable from a shift $p$ semitones down from $j$ to $i$. On the contrary, if we apply the scaling transformation in the audio domain and then recompute the spectrogram, we are able to reproduce the artifacts of the operation itself as well. This solves the symmetry problem and makes it possible to infer the correct ancestry relationship.

Once we have the scale-compensated spectrogram $\mathcal{I}_i^{(s)}$, the algorithm goes back to the first steps[2], performing feature extraction and matching them for the second time (Figure 4.4b). This leads to the estimation of a new geometric transformation, $H^{(s)}$. Since $\mathcal{I}_i^{(s)}$ is different from $\mathcal{I}_i$, matched features may also change. In the example of Figure 4.4, some new matches appear in the top left corner of the image, while some others in the bottom left corner are lost.

Given the affine transformation matrix $H^{(s)}$, the algorithm warps $\mathcal{I}_i^{(s)}$ into $\mathcal{I}_j$, providing the registered spectrogram $\mathcal{I}_i^{(r)}$. This registration allows to compensate

[2]If no scaling operation was detected, this phase is not performed.



(a) First matching, $(\mathcal{I}_i, \mathcal{I}_j)$.

(b) Second matching, $(\mathcal{I}_i^{(s)}, \mathcal{I}_j)$.

Figure 4.4: Matched features before and after scale compensation.

any kind of time misalignment, such as trim and/or delays between the two tracks. The superposition of $\mathcal{I}_i^{(r)}$ and $I_j$ is shown in Figure 4.6, where we can see that all features have been correctly aligned. In addition, gray areas indicate a perfect superposition of the two images, while red and blue areas are due to a dominance of $\mathcal{I}_i^{(r)}$ or $\mathcal{I}_j$, respectively.

Finally, the dissimilarity value $d_{i,j}$ is computed as the point-wise mean squared error (MSE) of the two images.

$$d_{i,j} = \text{MSE}\left(\mathcal{I}_i^{(r)}, \mathcal{I}_j\right) \tag{4.4}$$

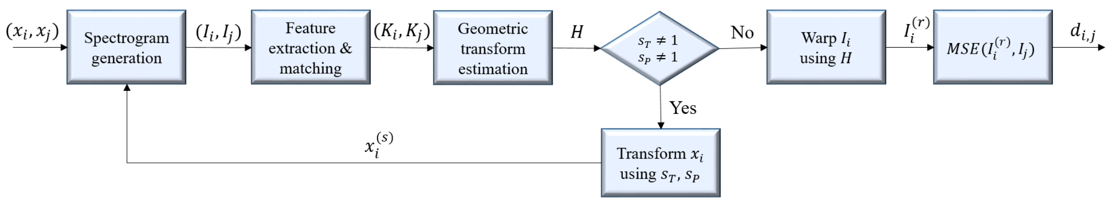


Figure 4.5: Block diagram of spectrogram registration and dissimilarity computation processes.

Once dissimilarities have been calculated for each spectrogram pair $(i, j)$, matrix $M$ is interpreted as a complete graph $G = (V, E)$. This structure is passed to the Edmonds' optimum branching algorithm, which outputs the related minimum spanning arborescence $\hat{\varphi} = (V, E^*)$. Graph $\hat{\varphi}$ represents our estimated audio phylogenetic tree (Figure 4.7). However, matrix $M$ needs some further processing

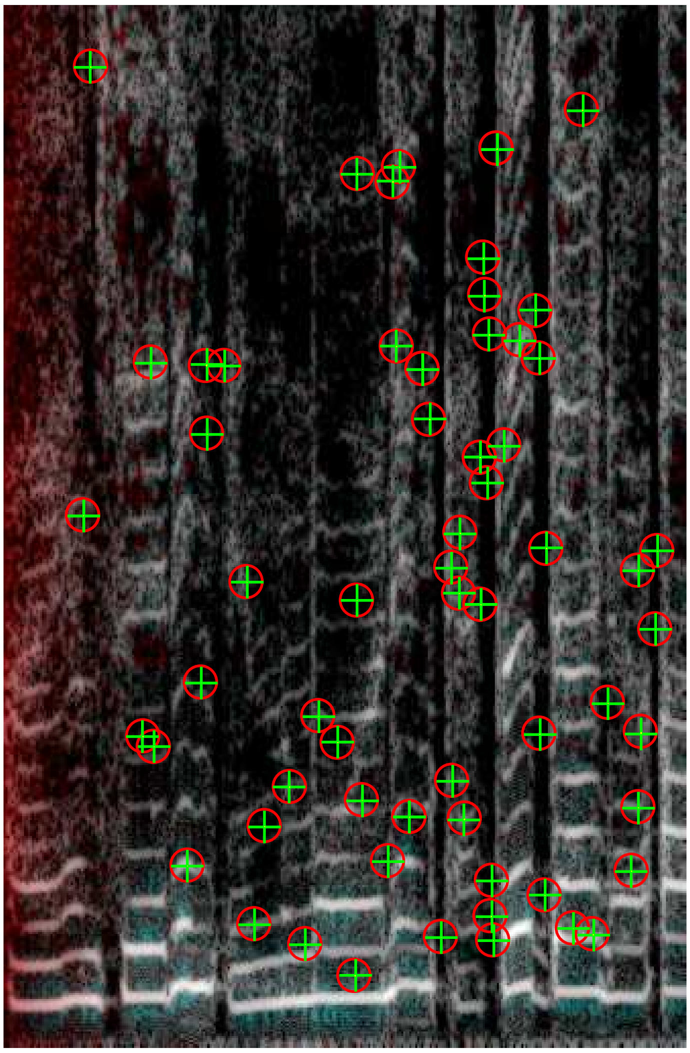


Figure 4.6: Registered spectrogram pair, $(\mathcal{I}_i^{(r)}, \mathcal{I}_j)$. The two sets of features (circles for $\mathcal{I}_i$, crosses for $\mathcal{I}_j$) are aligned. Gray areas depict a perfect superposition. Red and blue areas indicate a dominance of $\mathcal{I}_i^{(r)}$ over $\mathcal{I}_j$ or vice versa, respectively.

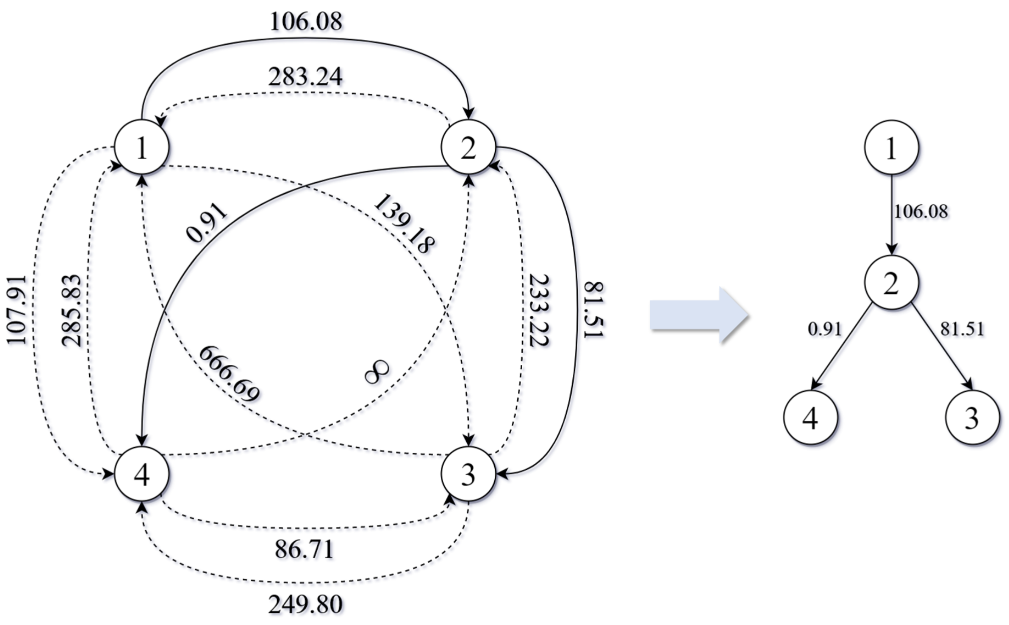


Figure 4.7: Tree estimation from complete dissimilarity graph using optimum branching algorithm.

before being passed to the optimum branching algorithm, and this will be the topic of the next section.

## 4.3 Graph pruning

The last block at the core of our algorithm is needed in order to refine the dissimilarity matrix and consequently improve the performance of the tree reconstruction algorithm. The problem here is that there still could be some pairs $(i, j)$ such that $d_{i,j} = d_{j,i}$. As previously stated, symmetric dissimilarities tend to affect the precision of the reconstructed phylogenetic tree since some parent-child relationships may be reversed. Such situation mainly occurs when a track undergoes only those specific transformations that are not compensated by the dissimilarity block.

Let $(\mathcal{I}_o, \mathcal{I}_f)$ be an image pair, where $\mathcal{I}_o$ is the spectrogram related to the original track and $\mathcal{I}_f$ differs from $\mathcal{I}_o$ only by a fade effect. The estimated geometric transformations that map $\mathcal{I}_o$ into $\mathcal{I}_f$ and vice versa will both be identities, and therefore

$$\mathrm{MSE}\left(\mathcal{I}_o^{(r)}, \mathcal{I}_f\right) = \mathrm{MSE}\left(\mathcal{I}_f^{(r)}, \mathcal{I}_o\right) \tag{4.5}$$

which implies $d_{o,f} = d_{f,o}$. However, assuming that no other transformations are present (or equivalently that all other transformations have been correctly compensated in the previous block) made exception for fade, the identification of the original image between these two is a trivial task at least by human inspection. In fact, the intensity gradient due to the fade effect is clearly visible (Figure 3.4b), and therefore the parent images is the one where such effect is not present. The same arguments also apply to compression. Let us consider the example of Section 4.2, where we have an image pair, $(\mathcal{I}_i, \mathcal{I}_j)$, in which $\mathcal{I}_j$ differs from $\mathcal{I}_i$ by trim, pitch shift, fade and compression. In Figure 4.6, the two spectrograms have been superimposed after the registration process and it is possible to notice the differences due to the non-compensated operations, namely fade and compression. The red band in the left side of the image is related to fade, while the blue artifacts in the bottom part are related to compression. With this in mind, we can disambiguate the symmetric dissimilarities by the following steps.

1. Detect artifacts caused by fade and compression operations.
2. Exploit detected artifacts in order to infer the correct parent-child relationship direction.
3. If there is certainty (with respect to a suitable tolerance threshold) that $i$ is *not* an ancestor of $j$, set
$$d_{i,j} = +\infty \tag{4.6}$$
otherwise, dissimilarity computed in the previous block is left unchanged.

Recalling that the dissimilarity matrix $M$, defined as $M(i,j) = d_{i,j}$, $\forall(i,j)$, is then taken as the adjacency matrix of a directed graph, setting an edge weight to infinity implies the removal of that edge from the graph, which is usually referred to as a *pruning* operation.

The following sections describe the implementation of such detection algorithms for fade and compression, respectively.

### 4.3.1 Fade detection

In order to identify a fade effect, we follow a simple consideration about pixel energy distribution in the residual spectrogram. In particular, given a spectrogram pair, we first compute the point-wise absolute difference between the two images, obtaining the residual image, and then compare the energy content of its left and right side. A high energy concentration on one side of the residual is a good indicator of the presence of fade. If we consider the example of Section 4.3, where we have the original spectrogram, $\mathcal{I}_o$, and the one affected by fade, $\mathcal{I}_f$, and we compute the absolute difference of the two images, we obtain the result shown in Figure 4.8. Clearly, energy is highly concentrated on the left side of the image. Be careful that, while the right side may seem perfectly black by visual inspection, this is

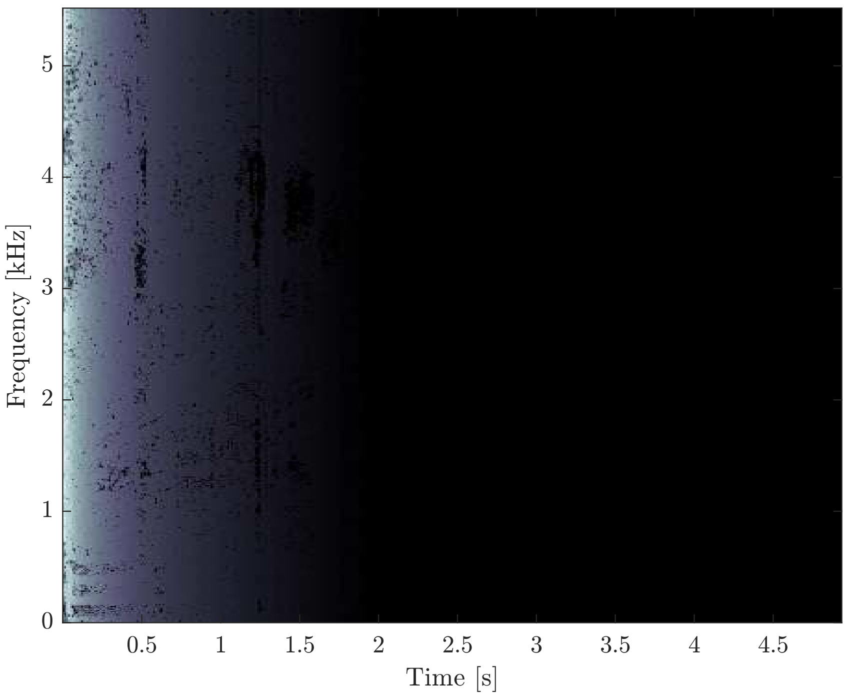


Figure 4.8: Residual spectrogram in presence of a fade effect.

not always true, because of possible registration errors and other non-compensated transformation artifacts. In the current example, however, the right side is actually black, because fade is the only transformation applied, and therefore the two audio tracks are identical from fade expiration onwards.

Let $(\mathcal{I}_i, \mathcal{I}_j)$ be a spectrogram pair and $\mathcal{I}_r = |\mathcal{I}_i - \mathcal{I}_j|$ the residual image. In addition, let $L$ and $R$ be the sets of pixels of the image left and right halves, respectively. The fade detection algorithm proceeds as follows.

1. Increase each pixel of $\mathcal{I}_r$ by $\epsilon = 2/N_{\text{px}}$, where $N_{\text{px}}$ is the total number of pixels. In this way, if one of the two side was totally black, its energy content is now equal to one.

2. Calculate the overall energy of the left and right halves.

$$E_L = \sum_{(u,v)\in L} \mathcal{I}_r(u,v) \tag{4.7}$$

$$E_R = \sum_{(u,v)\in R} \mathcal{I}_r(u,v) \tag{4.8}$$

3. If $E_L$ is sufficiently higher[3] than $E_R$ (or vice versa) we conclude that a fade

[3]At least one order of magnitude.

effect is present in the left (right) side.

4. Let $F \in \{L, R\}$ be the image half in which fade has been detected. We define an indicator $\eta$ as follows.

$$\eta = \operatorname{sgn}\left(\sum_{(u,v)\in F} \mathcal{I}_i(u,v) - \mathcal{I}_j(u,v)\right) \tag{4.9}$$

5. If $\eta = +1$, then $\mathcal{I}_i$ presents a higher overall energy in the spectrogram side where fade was detected. We conclude that $\mathcal{I}_i$ is the parent (or more generally an ancestor) of $\mathcal{I}_j$.

   If $\eta = -1$, then $\mathcal{I}_i$ presents a lower overall energy in the spectrogram side where fade was detected. We conclude that $\mathcal{I}_i$ is a child (or more generally a descendant) of $\mathcal{I}_j$.

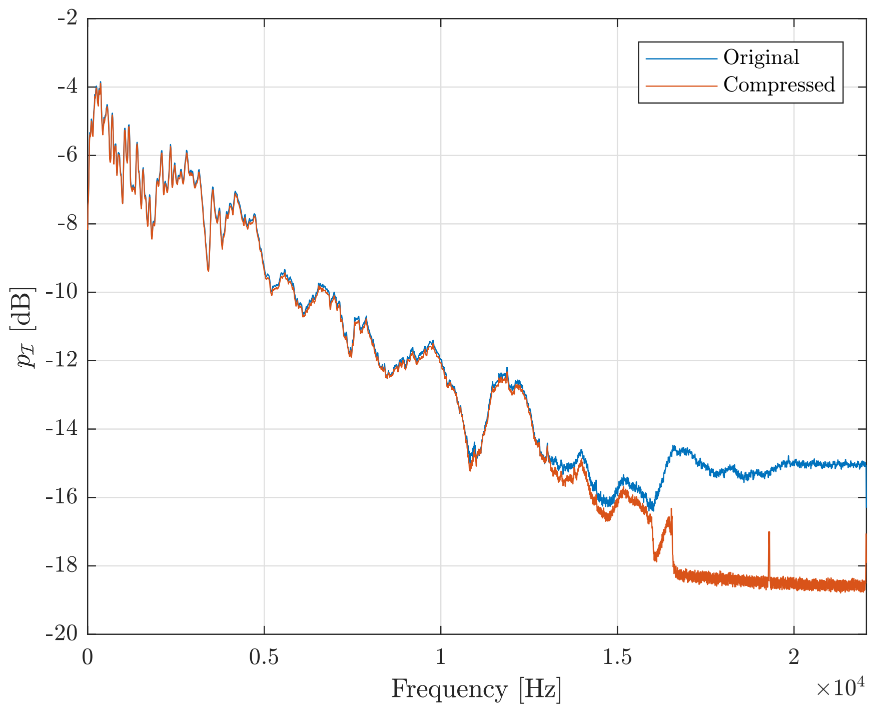
### 4.3.2 MP3 detection

The most evident features that allow us to detect compression are usually found in high frequency. As we have seen in Section 3.4, MP3 artifacts can be characterized as discontinuities in the average image intensity level along the frequency axis, such as low-pass filtering steps or other line-shaped features. Therefore, it is possible to detect them by using gradient-based algorithms. Let us consider a spectrogram image, $\mathcal{I}(u, v)$, with $u = 1, \dots, U$, $v = 1, \dots, V$. We define a function $p_{\mathcal{I}}(u)$ as follows.

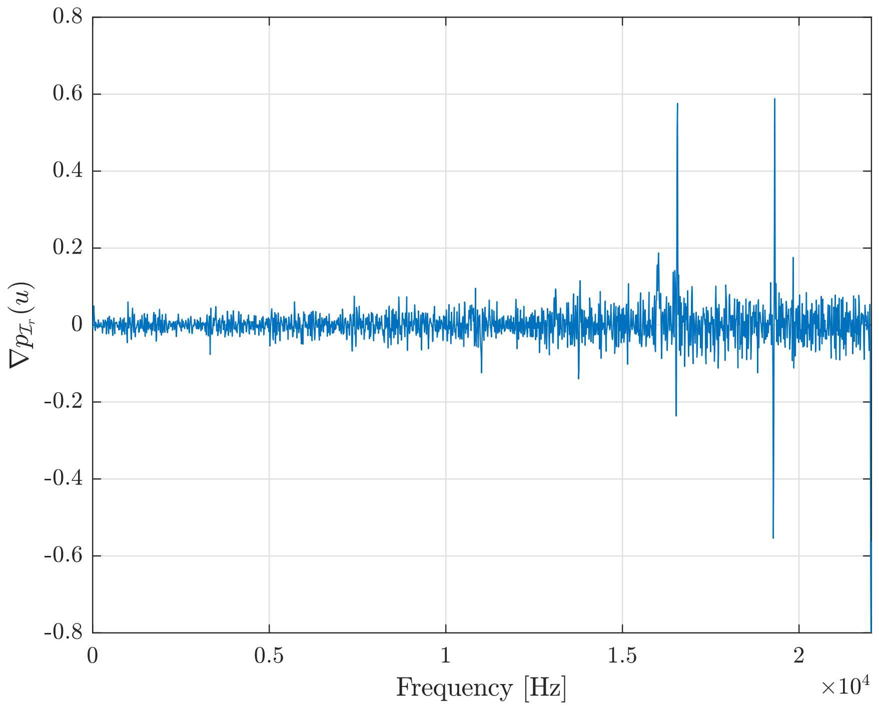
$$p_{\mathcal{I}}(u) = \frac{1}{V} \sum_{v=1,\dots,V} \mathcal{I}(u,v), \qquad \forall u = 1, \dots, U \tag{4.10}$$

Function $p_{\mathcal{I}}(u)$ computes the average pixel intensity value for each spectrogram row $u$, i.e. for each frequency band, and therefore it is possible to interpret it as a power spectral density (PSD) estimator. In Figure 4.9a we plot $p_{\mathcal{I}}(u)$ for spectrograms obtained from an unaltered track and an MP3-encoded near-duplicate of it. As previously stated, the two PSDs are very similar up to 12.5 kHz. For higher frequencies, they start to slowly separate from each other, up to about 16 kHz, where they rapidly diverge. This corresponds to the transition band of the low-pass filtering performed by the MP3 encoder. In addition, we can see a spike at around 19 kHz in the PSD of the compressed track, which indicates the presence of one of those aforementioned line-shaped artifacts.

In our approach, we exploit such spectral artifacts in order to detect MP3 compression and, possibly, the correct parent-child relationship. The employed



(a) PSD of original and compressed track.



(b) PSD gradient of residual spectrogram.

Figure 4.9: Gradient-based MP3 detection method.

method consists of the following steps.

1. Compute the residual spectrogram, $\mathcal{I}_r = |\mathcal{I}_i - \mathcal{I}_j|$, as for fade detection. An example of the visual result is shown in Figure 3.9c.

2. Apply (4.10) to the residual spectrogram, obtaining function $p_{\mathcal{I}_r}$, and compute the gradient, $\nabla p_{\mathcal{I}_r}$. In Figure 4.9b we show the behaviour of $\nabla p_{\mathcal{I}_r}$ for the current example.

3. Compute the standard deviation, $\sigma$, of $\nabla p_{\mathcal{I}_r}$, and look for those frequencies at which the function value is above a given threshold, expressed in terms of multiples of $\sigma$. In our experiments, we required a minimum value of $5\sigma$. Keep in memory the frequency values (spectrogram rows) at which gradient features have been found, $\hat{u}_k$, $k = 1, \ldots, K$, where $K$ is the number of features detected.

4. Apply (4.10) to $\mathcal{I}_i$ and $\mathcal{I}_j$, then compute the gradients $\nabla p_{\mathcal{I}_i}$ and $\nabla p_{\mathcal{I}_j}$, respectively.

5. For each $\hat{u}_k$, perform the $5\sigma$ check on $\nabla p_{\mathcal{I}_i}(\hat{u}_k)$ and $\nabla p_{\mathcal{I}_j}(\hat{u}_k)$, using the related standard deviations. We define a score for each of the two images, $C_i$ and $C_j$, and we increment it every time we detect a gradient feature. Eventually, we will have $0 \leq C_i, C_j \leq K$.

6. If $C_i + \Delta_C < C_j$, where $\Delta_C$ is a constant opportunely tuned, then $\mathcal{I}_i$ presents sufficiently less artifacts than $\mathcal{I}_j$ to state that track $i$ is an ancestor of $j$.

   In our experiments, we set $\Delta_C = 1$, so that a gap of at least two artifacts is required to state a parent-child relationship. In our example, $\mathcal{I}_i$ presents no artifacts, thus $C_i = 0$, and $\mathcal{I}_j$ has two, yielding $C_j = 2$. Therefore we correctly state that $i$ is an ancestor of $j$.

   If $C_i + \Delta_C > C_j$, then $\mathcal{I}_i$ presents sufficiently more artifacts than $\mathcal{I}_j$ to state that track $i$ is a descendant of $j$.

   In all other cases, we do not take any decision and dissimilarity computed in the previous block is left unchanged.

Chapter 5

# Experiments and results

In this chapter we describe our experimental scenario, including all details about the creation of the dataset we used for simulations and the employed evaluation metrics. Then, we comment on the obtained results, also considering performance comparison against the state-of-the-art [27].

## 5.1 Dataset generation

The proposed algorithm has been validated on a dataset built by generating multiple near-duplicate trees from tracks of different genres and length. Specifically, we used five different uncompressed and original-quality audio excerpts as roots for our generated phylogenetic trees.

1. Excerpt from "Ludwig Thuille: *Piano Sextet in B-flat major, Op. 6 - III*"[1] (26 s).
2. Electric guitar blues riff[2] (34 s).
3. Hand-crank music box playing *Amazing Grace*[3] (31 s).
4. Male voice reading verses from Edgar Allan Poe's *The Raven*[4] (41 s).
5. MIDI loop from *Super Mario*[5] (15 s).

Starting from these tracks, near-duplicate trees have been generated by applying random transformations to the root, thus generating the first descendant, and

[1]https://www.jamendo.com/track/146588/
[2]https://www.freesound.org/s/30021/
[3]https://www.freesound.org/s/180384/
[4]https://www.freesound.org/s/189467/
[5]https://www.freesound.org/s/179684/

then iterating the process randomly selecting the parent of the incoming child, until the desired number of nodes is reached. The set of considered audio transformations was implemented in MATLAB language and consists of the following operations.

1. Trim, applied to the leading or trailing samples, with a maximum length of 3 seconds.
2. Fade, applied to the leading (fade-in) or trailing (fade-out) samples, with a maximum length of 3 seconds.
3. Time stretching, speed-up or slow-down up to 10%.
4. Pitch shifting, one semitone up or down.
5. MP3 coding, using the LAME[6] encoder, with quality factor $q \in \{2, 3, 4\}$.

Both time stretching and pitch shifting operations were implemented by using a MATLAB implementation of a phase vocoder [50]. All transformations and related parameters are selected randomly with uniform distribution during the near-duplicate generation process.

From these premises, we constructed different phylogenetic trees according to two strategies:

◇ *single transformation dataset* - for each track, we generated 10 trees of 50 nodes each, with a single audio transformation per edge.

◇ *multiple transformations dataset* - for each track we generated additional 10 trees of 50 nodes each with multiple transformations per edge (a random number up to a maximum of four).

Our dataset thus consists of 100 phylogenetic trees for a total amount of 5,000 audio excerpts.

## 5.2 Evaluation metrics

During the validation phase of a phylogenetic analysis algorithm, a suitable evaluation system is needed in order to quantify how the reconstructed structure is close to the actual one. For this purpose, Dias *et al.* [1, 3] developed four metrics, namely *Root*, *Edges*, *Leaves* and *Ancestry*, for scenarios in which the ground-truth is available. Each of these metrics evaluates a different set of properties of the

[6]http://lame.sourceforge.net/

estimated tree, and together they provide a picture of the overall behaviour of the reconstruction algorithm. This evaluation system has become a standard for most of the studies on multimedia phylogeny, since it allows to produce a set of quantitative results which are both meaningful for the evaluation phase and comparable to those of the related work in the literature.

Given a reconstructed phylogenetic tree $\hat{\varphi} = (V, \hat{E})$ an its correspondent ground truth tree $\varphi = (V, E)$, with $|V| = n$, the metrics are defined as follows.

### Root

The *Root* metric simply compares the root of $\varphi$ to that of $\hat{\varphi}$, returning 1 if they match and 0 otherwise.

$$Root(\varphi, \hat{\varphi}) = \begin{cases} 1, & \text{if } root(\varphi) = root(\hat{\varphi}) \\ 0, & \text{if } root(\varphi) \neq root(\hat{\varphi}) \end{cases} \tag{5.1}$$

### Edges

The *Edges* metric evaluates the accuracy of the estimate of edge connections. If an edge connecting two nodes in $\hat{\varphi}$ also appears in $\varphi$, then it is considered to be correct. The percentage of correct edges is calculated by taking the cardinality of the intersection $E \cap \hat{E}$ and dividing it by the number of edges of the ground-truth tree, $|E| = n - 1$.

$$Edges(\varphi, \hat{\varphi}) = \frac{|E \cap \hat{E}|}{n - 1} \tag{5.2}$$

### Leaves

The *Leaves* metric evaluates the accuracy of the identification of those nodes without descendants (leaves of the tree). Let $L$ and $\hat{L}$ be the leaves of $\varphi$ and $\hat{\varphi}$, respectively. The metric is expressed as the ratio between the cardinalities of the intersection $L \cap \hat{L}$, representing the leaf nodes of $\varphi$ correctly detected in $\hat{\varphi}$, and their union.

$$Leaves(\varphi, \hat{\varphi}) = \frac{|L \cap \hat{L}|}{|L \cup \hat{L}|} \tag{5.3}$$

### Ancestry

The *Ancestry* metric evaluates the accuracy of the estimate of ancestry relationships. Specifically, for each node in the tree the metric counts how many of its ancestors (parent, grandparent, great grandparent, and so on up to the root) in $\varphi$ are correctly found in $\hat{\varphi}$. Formally, given a tree $G = (V, E)$ its ancestry $A$ is defined as

$A = \{(x, y) : \text{`}y \text{ is a descendant of } x\text{'}, \forall x, y \in V\}$. It follows from this that $E \subseteq A$. For example, considering the simple graph $G = (\{v_1, v_2, v_3\}, \{(v_1, v_2), (v_2, v_3)\})$, i.e. a chain of tree nodes, its ancestry would be $\{(v_1, v_2), (v_1, v_3), (v_2, v_3)\}$. Let $A$ and $\hat{A}$ be the ancestries of $\varphi$ and $\hat{\varphi}$, respectively. The expression of the *Ancestry* metric is analogous to that of the *Leaves* metric.

$$Ancestry(\varphi, \hat{\varphi}) = \frac{|A \cap \hat{A}|}{|A \cup \hat{A}|} \tag{5.4}$$

**Forest metrics**

For the evaluation of phylogenetic forests the same metrics are considered, but they are expressed by a more general formulation:

$$Metric(F, \hat{F}) = \frac{|S \cap \hat{S}|}{|S \cup \hat{S}|} \tag{5.5}$$

where $Metric \in \{Root, Edges, Leaves, Ancestry\}$, $F$ is the ground-truth forest, $\hat{F}$ is the estimated forest, and $S \in \{R, E, L, A\}$ are the interest features.

## 5.3 Algorithm validation

The proposed algorithm was implemented in MATLAB language (release R2017a). In particular, we employed the functions provided by the Computer Vision System Toolbox to perform feature detection, description and matching (using SURF), to estimate the geometric transformation with a robust outlier removal, and to warp images according to the estimated transformation. A phase vocoder implementation was used to perform the time and frequency scaling compensation in the dissimilarity block.

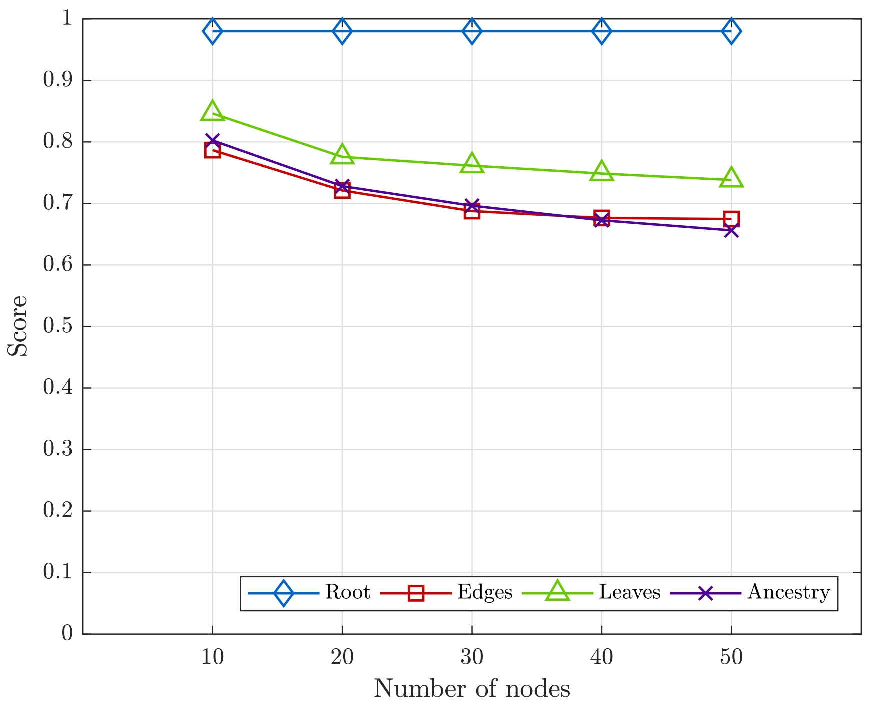
The first experiment we performed consisted in reconstructing trees from our generated dataset and evaluating the accuracy of the estimate with respect to the ground-truth. Multiple simulations were run for different sizes of the tree, with the number of nodes $K$ varying from 10 to 50. This was implemented by randomly pruning a set of $50 - K$ nodes from the dataset trees, starting from the leaves and moving upwards. The algorithm received in input $K$ unordered tracks with no additional information and returned the estimated phylogenetic tree, which was then evaluated by means of the *Root*, *Edges*, *Leaves* and *Ancestry* metrics. This was repeated for each tree in the dataset. In addition, we conducted the experiment separately on the single and multiple transformations datasets, in order to find out whether the algorithm encountered difficulties on one case with respect to the other. In Figure 5.1a and Figure 5.1b we show the results obtained for the single and multiple transformations datasets, respectively. As we can see, except for the smallest tree size ($K = 10$) in which the single-transformation dataset seems to provide slightly better results, the two scenarios are comparable. In particular, the *Root* metric is stable at 98% in the first case, and a bit more noisy in the second one, while remaining at around 96% on average. These results show that the algorithm is not particularly strained by the presence of a higher number of processing operations. Also, such results are quantitatively compatible to those obtained in the framework of image phylogeny [3].

In a practical case, however, it is reasonable to assume that only a subset of the nodes/tracks is available to the analyst. Therefore, the underlying dissimilarity graph will have some empty spaces due to the missing nodes. In order to validate the proposed solution in this kind of context, we devised an additional experiment consisting in testing our algorithm on trees where some nodes/tracks were randomly removed. The ground-truth of these trees was then generated by connecting all the nodes without a parent to their closest ancestor (Figure 5.2). This is the most sensible result the algorithm may provide in cases where some nodes are missing. For instance, if we consider a three-node chain (grandparent, parent and child) and assume we are missing the parent node, the most sensible outcome is to state that the child node is a direct descendant of the grandparent node. In our experiments, however, we always preserved the root node. With reference to the same experiment in image phylogeny [1], performance are expected to decrease as we increase the number of removed nodes. We conducted this test with up to 25 removed nodes (half the size of the whole tree). Results are reported in Figure 5.3. As expected, we observe an overall decrease of about 10% for the *Edges*, *Leaves* and *Ancestry* metrics. *Root* metric, instead, does not present a decreasing trend, remaining approximately constant at around 95%.

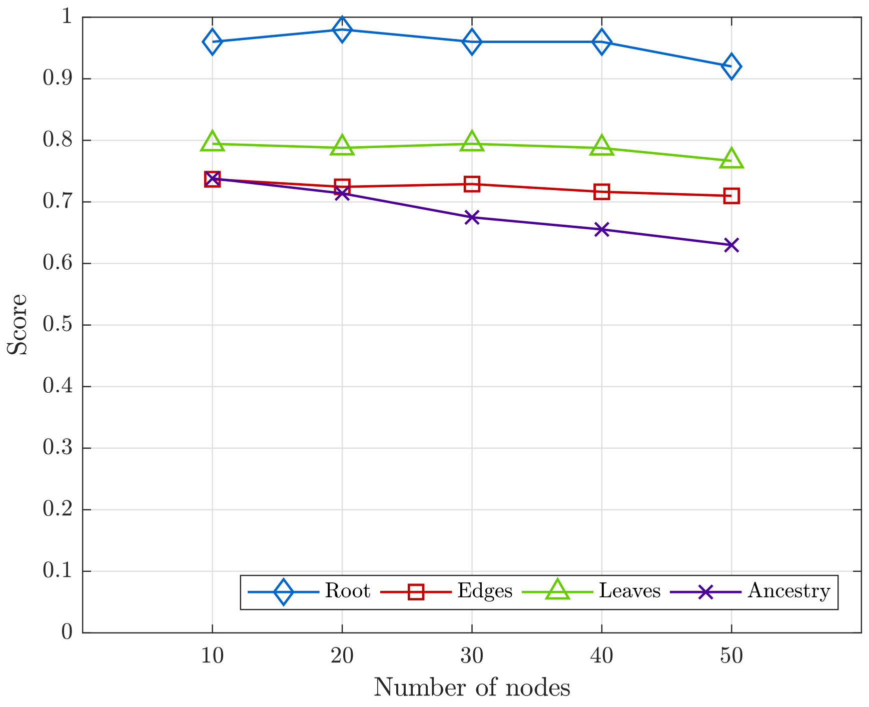
## 5.4 Comparison against state-of-the-art

In addition to assessing accuracy results of our method under different conditions, we also performed a comparison against the baseline method presented in [27]. Specifically, we run both algorithms on 40 trees of 10 nodes each, obtained using the single transformation strategy for dataset generation.

The algorithm in [27] performs a brute-force search on a candidate set of audio transformations and parameters. We included in the candidate set all editing



(a) Single transformation per edge.



(b) Multiple transformatione per edge (up to four).

Figure 5.1: R-E-L-A metrics for different tree sizes, from 10 to 50 nodes.

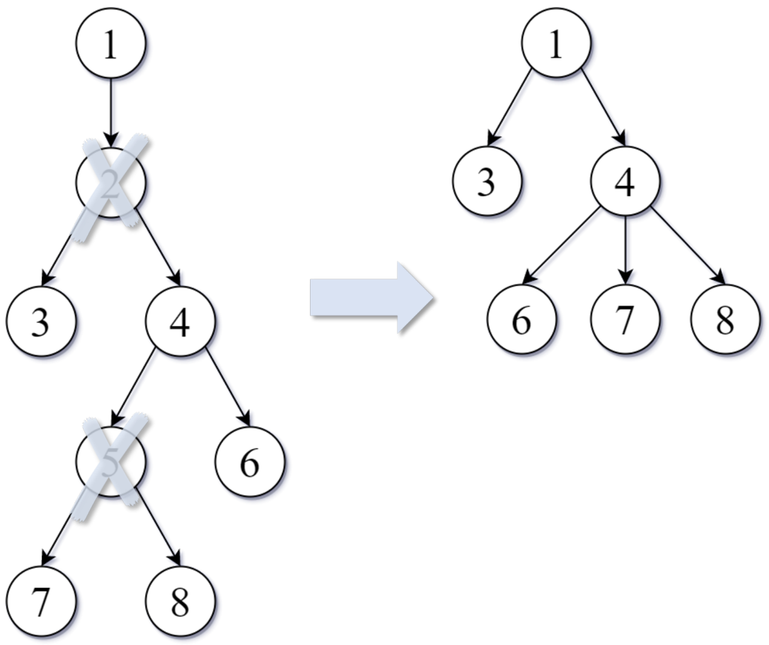


Figure 5.2: Example of correct tree reconstruction in case some nodes are missing in the available near-duplicate set.

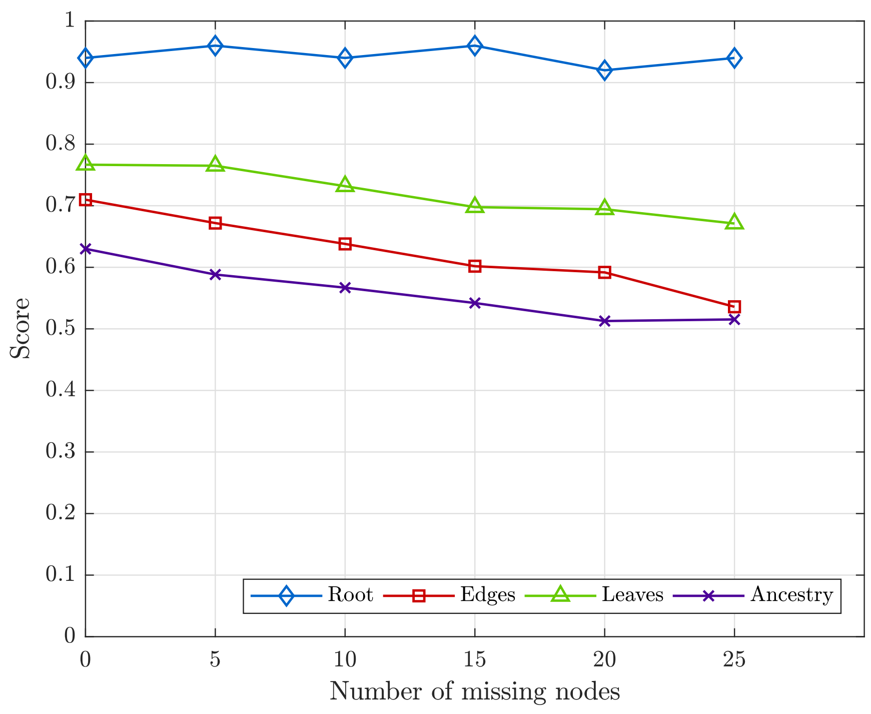


Figure 5.3: R-E-L-A metrics for increasing numbers of removed nodes (root preserving).

| | *Root* | *Edges* | *Leaves* | *Ancestry* | Time |
|---|---|---|---|---|---|
| baseline [27] | **97.5%** | 71.7% | 78.3% | 77.5% | 436 s |
| proposed | **97.5%** | **76.1%** | **81.4%** | **79.3%** | **203 s** |

Table 5.1: Comparison against baseline solution [27]. The proposed solution achieves higher accuracy with average processing time for a 10 nodes tree that is less than a half.

operations actually used to generate the dataset. As parameters for the brute-force grid search, we selected three candidates for compression (i.e., the ones used for dataset generation), 20 for fading (i.e., uniformly sampling 0 to 3 seconds at either heading or tailing samples of the track), 20 for time stretching (uniformly sampling the range used for dataset generation) and 8 for pitch shifting (from -4 to +4 semitones). We implemented the fastest version of [27], which only search for a single transformation from node to node coherently with the used dataset.

Results are reported in Table 5.1. It is possible to notice that the proposed approach always performs slightly better (or on par) with the baseline. Moreover, we also considered the average processing time needed to process a tree with MATLAB implementations of both algorithms run on a MacBook Pro equipped with a 2.2 Ghz Intel Core i7, 8 GB or RAM and SSD disk. This test confirms that our solution is more efficient, being able to process each tree in less than half the time needed by [27].

Chapter 6

# Conclusions and future work

In this thesis we presented an overview on the field of multimedia phylogeny, starting from the basic definitions and then presenting some proposed solutions related to the dissimilarity computation for different types of documents (images, videos and audio tracks) and the reconstruction of the phylogenetic trees associated to the processing history of a set of near-duplicate documents. Then, we discussed the application of computer vision techniques to time-frequency representations of audio tracks, with a special focus on how audio modifications are mapped into geometric transformations in the image domain and how it is possible to detect and classify them. From these considerations, we proposed a solution to the problem of the phylogenetic reconstruction of digital audio tracks based on a computer vision analysis of spectrograms. Given a set of near-duplicate audio tracks, the proposed algorithm is able to reconstruct causal relationships among all audio excerpts, enabling a precise estimation of the underlying structure. Differently from state-of-the-art techniques [27], our approach do not involve a brute-force search of possible audio editing operations applied to audio tracks. Moreover, the algorithm do not require any prior information about the specific set of candidate audio transformations.

The proposed solution was validated on a wide dataset of near-duplicate trees, built starting from audio excerpts of different genres. Performed experiments provided results that are comparable to those obtained in the related work within the framework of image phylogeny [1]. In comparison to the audio phylogeny state-of-the-art solution [27], instead, our approach performs overall better and with a shorter computation time.

Time-frequency analysis with computer vision techniques turned out to be an effective approach for audio phylogeny. However, a number of improvements remain possible. First, better results may be achieved with an improved management of compression artifacts. The proposed approach, in fact, can effectively deal with coding operations applied to an uncompressed track, but it has difficulty in inferring parent-child relationships in presence of multiple compressions. Moreover, future studies may be focused on spectrogram effects due to additional audio modifications, in order to expand the range of transformations taken into account, namely equalization, restoration techniques, ambient effects like reverb and so on.

Furthermore, the applicability of computer vision analysis on spectrograms can potentially extend from audio to any kind of one-dimensional signal. For instance, signal synchronization in distributed wireless sensors or biomedical devices can be as well performed by estimating the geometric transformation in the time-frequency representation domain. Similar considerations apply to signal authentication and restoration, where spectrograms combined with computer vision techniques may provide a robust comparative and analytical tool. Future work, therefore, shall explore those scenarios where image processing tools may provide better results than methods operating in the one-dimensional domain, thus opening the way to a new range of applications for the computer vision field.

Appendix A

# Projective geometry

This appendix outlines a number of concepts of projective geometry useful for understanding the geometric transformation estimation performed in the proposed method.

## A.1 Perspective projection

**Definition 5.** Perspective projection *maps a point $M$ in 3-D space on a plane $\mathcal{I}$ by intersecting the line passing through $M$ and $C$ – center of projection – with $\mathcal{I}$.*

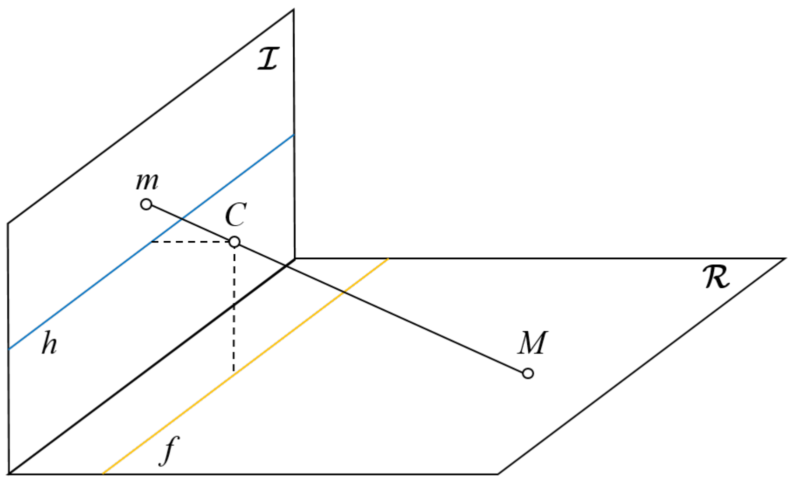


Figure A.1: Perspective projection of a point $M$ on a plane $\mathcal{R}$ into a point $m$ on the image plane $\mathcal{I}$.

Let $\mathcal{R}$ be a plane in 3-D space. All the points of $\mathcal{R}$ are mapped into points on the image plane $\mathcal{I}$, except for a line $f$ that is mapped to the infinite. Symmetrically, points of $\mathcal{R}$ lying at the infinite are mapped to the line $h$ of $\mathcal{I}$. From these considerations it is possible to derive the definition of projective space.

**Definition 6.** *The* projective space $\mathbb{P}^2$ *is defined as the union of the real plane* $\mathbb{R}^2$ *with the line at the infinite* $l_\infty$*:*

$$\mathbb{P}^2 = \mathbb{R}^2 \cup l_\infty \tag{A.1}$$

Similarly, one can define the projective space $\mathbb{P}^n$, where points at the infinite lie on an $n$-dimensional hyperplane.
In projective spaces two parallel lines share a common point at the infinite. Consequently, there is a point-line dualism:

- ◇ two points identify a line (passing through both);
- ◇ two lines identify a point (finite or at the infinite).

## A.2 Homogeneous coordinates

Let us consider the Cartesian space $\mathbb{R}^2$ and two lines defined by the standard form equations

$$\begin{array}{llll} a\,x & +\,b\,y & +\,c & = 0 \\ a'\,x & +\,b'\,y & +\,c' & = 0 \end{array} \tag{A.2}$$

where parameters $a, b, c, a', b', c' \in \mathbb{R}$. The common points between the two lines can be found solving the linear system

$$\begin{cases} a\,x & +\,b\,y & = -c \\ a'\,x & +\,b'\,y & = -c' \end{cases} \tag{A.3}$$

which can be written in matrix notation as

$$\begin{bmatrix} a & b \\ a' & b' \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -c \\ -c' \end{bmatrix} \tag{A.4}$$

$$A \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -c \\ -c' \end{bmatrix} \tag{A.5}$$

Equation (A.5) defines a linear system of two equations in two unknowns: in case a solution exists, it is possible to write it as

$$\begin{bmatrix} x \\ y \end{bmatrix} = A^{-1} \begin{bmatrix} -c \\ -c' \end{bmatrix}. \tag{A.6}$$

The inverse $A^{-1}$ can be written as

$$A^{-1} = \frac{1}{det(A)} adj(A) = \frac{1}{det(A)} \begin{bmatrix} b' & -b \\ -a' & a \end{bmatrix}$$

providing the solution

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{det(A)} \begin{bmatrix} b' & -b \\ -a' & a \end{bmatrix} \begin{bmatrix} -c \\ -c' \end{bmatrix} = \frac{1}{det(A)} \begin{bmatrix} b\ c' - b'\ c \\ a'\ c - a\ c' \end{bmatrix}$$

The resulting coordinates are

$$x = \frac{b\ c' - b'\ c}{det(A)} = \frac{det\left(\begin{bmatrix} b & c \\ b' & c' \end{bmatrix}\right)}{det\left(\begin{bmatrix} a & b \\ a' & b' \end{bmatrix}\right)} = \frac{\alpha}{\gamma} \tag{A.7}$$

and

$$y = \frac{a'\ c - a\ c'}{det(A)} = \frac{det\left(\begin{bmatrix} c & a \\ c' & a' \end{bmatrix}\right)}{det\left(\begin{bmatrix} a & b \\ a' & b' \end{bmatrix}\right)} = \frac{\beta}{\gamma} \tag{A.8}$$

where $\alpha, \beta, \gamma$ are real numbers. One can notice that all the possible outcomes of the initial system can be grouped into three conditions identified by the values of $\alpha$, $\beta$ and $\gamma$.

1. $\gamma \neq 0$ and $\alpha, \beta \in \mathbb{R} \Rightarrow$ the system has a solution, i.e., the two lines has a unique intersection point $(x, y)$.

2. $\gamma = \alpha = \beta = 0 \Rightarrow$ the two lines are the same, and therefore, the number of intersection points is infinite.

3. $\gamma \neq 0$ and $\alpha \neq 0 \vee \beta \neq 0 \Rightarrow$ the two lines are parallel since the slopes $r$ and $r'$ correspond. The intersection point lies at the infinite and the coordinates $x, y$ cannot be expressed by equations (A.7) and (A.8). Therefore, in Cartesian space $\mathbb{R}^2$, the system has no solutions, but in projective space $\mathbb{P}^2$ it is possible to state that the intersection point is identified by parameters $\alpha, \beta$ and $\gamma = 0$.

From these premises, it is possible to represent the point $\mathbf{m} \in \mathbb{P}^2$ using the triplet

$$\mathbf{m} = [\ \alpha\ \beta\ \gamma\ ]^T \tag{A.9}$$

This representation is called *homogeneous coordinates*. From the definition, it is possible to notice that:

$\boxed{\gamma \neq 0}$ $\Rightarrow$ the point is finite and corresponds to $(x, y)$ in $\mathbb{R}^2$;

$\boxed{\gamma = 0}$ $\Rightarrow$ the point lies at the infinite and has no correspondence in $\mathbb{R}^2$;

The triplet $(0, 0, 0)$ is considered not valid since it corresponds to infinite points in $\mathbb{R}^2$. When $\gamma \neq 0$, conversion from homogeneous to Cartesian coordinates is possible.

$$\mathbf{m} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \qquad \Rightarrow \qquad \tilde{\mathbf{m}} = \begin{bmatrix} \alpha/\gamma \\ \beta/\gamma \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{A.10}$$

Note that all the homogeneous coordinate vectors are defined with respect to a scale factor, i.e., $\mathbf{m}$ and $k\mathbf{m}$ are the same point. This make the inverse conversion from Cartesian to homogeneous coordinates quite simple:

$$\tilde{\mathbf{m}} = \begin{bmatrix} x \\ y \end{bmatrix} \qquad \Rightarrow \qquad \mathbf{m} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{A.11}$$

## A.3 Projective transformations

Homogeneous coordinates allow projective transformations (non-linear in Cartesian space) to be easily represented by a matrix.

**Definition 7.** *A* projective transformation *or* projectivity $f$ *is a linear function in homogeneous coordinates*

$$f : \mathbb{P}^n \to \mathbb{P}^n$$

*defined by the non-singular* $(n+1) \times (n+1)$ *matrix* $H$ *such that*

$$f(\mathbf{m}) : \mathbf{m} \mapsto H\mathbf{m}$$

Projectivities have the following properties:

- ◇ they preserve collinearity;
- ◇ they form a group, $f \in \mathcal{G}_P$;
- ◇ similarly for points, $H$ and $\lambda\ H$ (with $\lambda \in \mathbb{R}, \lambda \neq 0$) are the same.

**Definition 8.** *An* affine transformation *or* affinity $f$ *is projectivity that maps real points into real points and ideal points into ideal points, i.e.,*

$$f(\mathbf{m}) : \mathbf{m} \mapsto H\mathbf{m} \qquad \textit{with} \qquad H = \begin{bmatrix} A_{n\times n} & \mathbf{b} \\ \\ \mathbf{0}^T & 1 \end{bmatrix}$$

**Definition 9.** *A* similarity $f$ *is an affine transform that preserves the absolute conic* $\mathbf{\Omega}_\infty = \{\mathbf{x} \in \mathbb{P}^n : x_1^2 + x_2^2 + \ldots + x_n^2 = 0 \wedge x_{n+1} = 0\}$*, i.e.*

$$\forall \mathbf{x} \in \mathbf{\Omega}_\infty, \qquad H\mathbf{x} \in \mathbf{\Omega}_\infty$$

*Similarities can be defined by a matrix* $H$ *such that*

$$H = \begin{bmatrix} sR_{n\times n} & \mathbf{t} \\ \\ \mathbf{0}^T & 1 \end{bmatrix}$$

*where* $R$ *is an orthogonal matrix.*

In Cartesian coordinates, similarity operates as the composition of rescaling, rotation and translation, i.e.,

$$H\mathbf{x} = \begin{bmatrix} sR_{n\times n} & \mathbf{t} \\ \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} \qquad \Rightarrow \qquad sR\,\tilde{\mathbf{x}} + \mathbf{t}$$

In case $s = 1$,the transformation is called *rigid transformation* or *Euclidean.*
The different types of transformations in $\mathbb{P}^2$ and related properties are summarized in Table A.1.

## A.4 Homography estimation

A transformation mapping points in homogeneous coordinates from a plane to another is called an *homography.* Formally, it corresponds to a projective transformation in $\mathbb{P}^2$, i.e., it maps $\mathbf{m}$ into $\mathbf{m}'$ through a $3 \times 3$ non-singular matrix $H$,

$$\mathbf{m}' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \simeq \begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H\mathbf{m} \qquad \text{(A.12)}$$

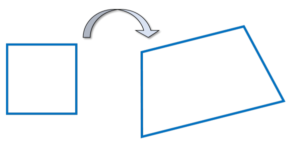
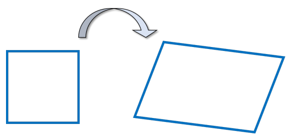
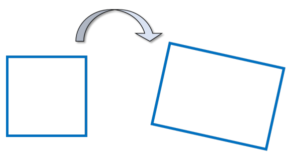
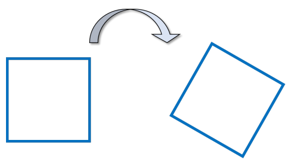
| Transformation | d.o.f. | Matrix | Distortion | Preserves |
|---|---|---|---|---|
| Projectivity | 8 | $\begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{bmatrix}$ |  | collinearity |
| Affinity | 6 | $\begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ 0 & 0 & 1 \end{bmatrix}$ |  | parallelism |
| Similarity | 4 | $\begin{bmatrix} sR_{n\times n} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$ |  | angles |
| Euclidean | 3 | $\begin{bmatrix} R_{n\times n} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$ |  | distances and lengths |

Table A.1: Transforms on $\mathbb{P}^2$

where $\simeq$ denotes here an equality with respect to a scale factor. For this reason the degree of freedom of $H$ is 8, despite having 9 components.

Let us consider the case of two sets of matched points, where we have $\mathbf{m}_i$ matched to $\mathbf{m}'_i$, $\forall i$. In absence of wrong matches, it is possible to write (A.12) for every pair of points, $\mathbf{m}'_i \simeq H\mathbf{m}_i$. Using the properties of the vector product, one can write

$$\mathbf{m}'_i \times H\mathbf{m}_i = [\mathbf{m}'_i]_\times H\mathbf{m}_i = \mathbf{0}_{2\times 1} \tag{A.13}$$

where $[\mathbf{m}'_i]_\times$ is a $3\times 3$ antisymmetric matrix defined as follows.

$$[\mathbf{m}'_i]_\times = \begin{bmatrix} 0 & -1 & v' \\ 1 & 0 & -u' \\ -v' & u' & 0 \end{bmatrix} \tag{A.14}$$

Applying the $vec(\cdot)$ operator and since $vec(\mathbf{x}) = \mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n$ is a column vector, it is possible to write

$$vec\left([\mathbf{m}'_i]_\times H\mathbf{m}_i\right) = \mathbf{0}. \tag{A.15}$$

Let us consider now the following property of the Kronecker product, which is denoted here by $\otimes$.

$$vec\,(ABC) = \left(C^T \otimes A\right) vec(B) \tag{A.16}$$

Applying (A.16) to (A.15) we have

$$vec\left([\mathbf{m}'_i]_\times H \mathbf{m}_i\right) = \left(\mathbf{m}_i^T \otimes [\mathbf{m}'_i]_\times\right) vec(H) = \mathbf{0} \tag{A.17}$$

The matrix $\mathbf{m}_i^T \otimes [\mathbf{m}'_i]_\times$ (which is sized $3 \times 9$) has rank equal to 2. The system in (A.17) therefore includes two equations in 8 unknowns. Since every couple of points brings two equations and we have 8 d.o.f. in $H$, we need at least $n = 4$ matching points to create a system of 8 equations in 8 unknowns

$$\begin{bmatrix} \mathbf{m}_1^T \otimes [\mathbf{m}'_1]_\times \\ \mathbf{m}_1^T \otimes [\mathbf{m}'_2]_\times \\ \vdots \\ \mathbf{m}_n^T \otimes [\mathbf{m}'_n]_\times \end{bmatrix} vec(H) = A\ vec(H) = \mathbf{0} \tag{A.18}$$

In practical cases, however, the system is usually overdetermined for robustness issues, i.e., the number of equations to be satisfied is much higher than the required minimum. An overdetermined system can be solved by looking for vectors of the kernel of $A$. A possible solution consists in applying the singular value decomposition (SVD), obtaining

$$A = UDV^T \tag{A.19}$$

and selecting the last rows of $V^T$, which form a basis of $ker(A)$. In the case of homography estimation, the matrix in (A.18) is $2n \times 9$ and has rank equal to 8. The rank theorem implies that $dim(\ ker(A)\ ) = 1$. Therefore, the solution is provided by the last row of $V^T$.

# Acknowledgements

The drafting of this thesis, and the work behind it, would not have been possible without the contribution of a number of people, to whom I wish to express my gratitude.

Prof. Simone Milani, for allowing me to work in the laboratory for many months, for his valuable help with my PhD application and for being an exceptional advisor overall.

Prof. Paolo Bestagini, for his willingness during the whole thesis period and for his valuable contribution to the experimental results.

My family, for their unconditional and invaluable support, not only economic, but also and above all emotional. In particular, my mother, Roberta, my father, Massimiliano, and my grandparents, Marisa and Lino.

Secondly, but no less important, I feel the need to thank all the people who have helped to make these university years an unforgettable experience. In particular, my friends, classmates and choirmates. A special thanks, for always being there for me, to Caterina, Ester, Federico, Giacomo, Sebastiano and Tommaso, and for still being alive, to Jon Snow.

# Bibliography

[1] Z. Dias, A. Rocha, and S. Goldenstein, "Image phylogeny by minimal spanning trees," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, pp. 774–788, Apr. 2012.

[2] A. Joly, O. Buisson, and C. Frelicot, "Content-based copy retrieval using distortion-based probabilistic similarity search," *IEEE Trans. Multimedia*, vol. 9, pp. 293–306, Feb. 2007.

[3] Z. Dias, A. Rocha, and S. Goldenstein, "First steps toward image phylogeny," in *2010 IEEE International Workshop on Information Forensics and Security*, 2010, pp. 1–6.

[4] B. Lewin, *Genes VI.* London, UK: Oxford University Press, 1997.

[5] C. D. McKinsey, *The Encyclopedia of Biblical Errancy.* New York, NY: Prometheus Books, 1995.

[6] J. Reeds, *John Dee: Interdisciplinary studies in English Renaissance thought.* Dordrecht, NL: Springer, 2006, ch. John Dee and the magic tables in the Book of Soyga, pp. 177–204.

[7] A. Rocha, W. Scheirer, T. Boult, and S. Goldenstein, "Vision of the unseen: current trends and challenges in digital image and video forensics," *ACM Comput. Surv.*, vol. 43, pp. 26:1–26:42, Oct. 2011.

[8] M. A. Oikawa, Z. Dias, A. Rocha, and S. Goldenstein, "Distances in multimedia phylogeny," *International Transactions in Operational Research*, vol. 23, no. 5, pp. 921–946, Sep. 2016.

[9] Z. Dias, S. Goldenstein, and A. Rocha, "Toward image phylogeny forests: Automatically recovering semantically similar image relationships," *Forensic Sci. Int.*, vol. 231, pp. 178–189, 2013.

[10] A. Oliveira, P. Ferrara, A. D. Rosa, A. Piva, M. Barni, S. Goldenstein, Z. Dias, and A. Rocha, "Multiple parenting identification in image phylogeny," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct. 2014, pp. 5347–5351.

[11] Z. Dias, A. Rocha, and S. Goldenstein, "Video phylogeny: recovering near-duplicate video relationships," in *2011 IEEE International Workshop on Information Forensics and Security*, Nov. 2011, pp. 1–6.

[12] F. Costa, S. Lameri, P. Bestagini, Z. Dias, S. Tubaro, and A. Rocha, "Hash-based frame selection for video phylogeny," in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec. 2016, pp. 1–6.

[13] X. Zhang, B. Zhu, L. Li, W. Li, X. Li, W. Wang, P. Lu, and W. Zhang, "Sift-based local spectrogram image descriptor: a novel feature for robust music identification," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, p. 6, 2015.

[14] M. Zanoni, S. Lusardi, P. Bestagini, A. Canclini, A. Sarti, and S. Tubaro, "Efficient music identification approach based on local spectrogram image descriptors," in *Audio Engineering Society Convention 142*, May 2017.

[15] S. Goldenstein and A. Rocha, "High-profile forensic analysis of images," in *3rd International Conference on Imaging for Crime Detection and Prevention (ICDP 2009)*, Dec. 2009, pp. 1–6.

[16] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '94. London, UK: Springer-Verlag, 1994, pp. 257–270.

[17] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker, *Digital watermarking and steganography*, 2nd ed. Boston, MA: Morgan Kaufmann, 2007.

[18] F. Costa, M. A. Oikawa, Z. Dias, S. Goldenstein, and A. R. de Rocha, "Image phylogeny forests reconstruction," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 10, pp. 1533–1546, Oct. 2014.

[19] A. Oliveira, P. Ferrara, A. D. Rosa, A. Piva, M. Barni, S. Goldenstein, Z. Dias, and A. Rocha, "Multiple parenting phylogeny relationships in digital images," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 2, pp. 328–343, Feb. 2016.

[20] M. O. Searcóid, *Metric spaces.* New York, NY: Springer, 2006.

[21] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[22] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[23] A. Melloni, P. Bestagini, S. Milani, M. Tagliasacchi, A. Rocha, and S. Tubaro, "Image phylogeny through dissimilarity metrics fusion," in *2014 5th European Workshop on Visual Information Processing (EUVIP)*, Dec. 2014, pp. 1–6.

[24] A. D. Rosa, F. Uccheddu, A. Piva, M. Barni, and A. Costanzo, "Exploring image dependencies: a new challenge in image forensics," in *Proceedings of SPIE Media Forensics and Security II*, vol. 7541, San Jose, CA, Jan. 2010, pp. X1–X12.

[25] M. K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin, "Low-complexity image denoising based on statistical modeling of wavelet coefficients," *IEEE Signal. Proc. Let.*, vol. 6, no. 12, pp. 300–303, Dec. 1999.

[26] F. O. Costa, S. Lameri, P. Bestagini, Z. Dias, A. Rocha, M. Tagliasacchi, and S. Tubaro, "Phylogeny reconstruction for misaligned and compressed video sequences," in *2015 IEEE International Conference on Image Processing (ICIP)*, Sep. 2015, pp. 301–305.

[27] M. Nucci, M. Tagliasacchi, and S. Tubaro, "A phylogenetic analysis of near-duplicate audio tracks," in *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2013, pp. 99–104.

[28] G. Gordon and E. McMahon, "A greedoid polynomial which distinguishes rooted arborescences," *Proc. Amer. Math. Soc.*, vol. 107, no. 2, pp. 287–298, 1989.

[29] M. X. Goemans, "Lecture notes on the arborescence problem," Massachusetts Institute of Technology, Apr. 2007.

[30] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[31] Z. Dias, S. Goldenstein, and A. Rocha, "Exploring heuristic and optimum branching algorithms for image phylogeny," *J. Vis. Comun. Image Represent.*, vol. 24, no. 7, pp. 1124–1134, Oct. 2013.

[32] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, Nov. 1957.

[33] Y. J. Chu and T. H. Liu, "On the shortest arborescence of a directed graph," *Science Sinica*, vol. 14, p. 1396–1400, 1965.

[34] J. Edmonds, "Optimum branchings," *J. Res. Nat. Bur. Standards*, vol. 71B, p. 233–240, 1967.

[35] F. Bock, "An algorithm to construct a minimum directed spanning tree in a directed network," in *Developments in Operations Research.* New York, NY: Gordon & Breach, 1971, pp. 29–44.

[36] R. E. Tarjan, "Finding optimum branchings," *Networks*, vol. 7, no. 1, pp. 25–35, 1977.

[37] P. Flandrin, *Time–frequency/Time–scale analysis.* San Diego, CA: Academic Press, 1999, vol. 10, ch. Wavelet analysis and its applications.

[38] L. Cohen, *Time–frequency analysis.* New York, NY: Prentice-Hall, 1995.

[39] G. Strang, "Wavelets," *American Scientist*, vol. 82, pp. 256–266, Jun. 1994.

[40] N. Benvenuto and G. Cherubini, *Algorithms for communications systems and their applications.* John Wiley & Sons, 2002.

[41] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Math. Comp.*, vol. 19, pp. 297–301, 1965.

[42] J. B. Allen and L. R. Rabiner, "A unified approach to short-time fourier analysis and synthesis," *Proc. IEEE*, vol. 65, no. 11, pp. 1558–1564, Nov. 1977.

[43] D. Gabor, "Theory of communication," *Journal of the Institute of Electrical Engineering*, vol. 93, pp. 429–457, 1946.

[44] J. Haluska, *The mathematical theory of tone systems.* Chapman & Hall/CRC, 2003.

[45] T. Lindeberg, *Scale-space theory in computer vision.* US: Springer, 1994.

[46] J. J. Koenderink, "The structure of images," *Biological Cybernetics*, vol. 50, no. 5, pp. 363–370, 1984. [Online]. Available: http://dx.doi.org/10.1007/BF00336961

[47] M. Brown and D. Lowe, “Invariant features from interest point groups,” in *Proceedings of the British Machine Vision Conference*, D. Marshall and P. L. Rosin, Eds. BMVA Press, 2002, pp. 23.1–23.10.

[48] A. Haar, “Zur theorie der orthogonalen funktionensysteme,” *Mathematische Annalen*, vol. 69, no. 3, pp. 331–371, 1910.

[49] A. Fusiello, *Visione computazionale. Tecniche di ricostruzione tridimensionale.* FrancoAngeli, 2013.

[50] J. Flanagan and R. Golden, “Phase vocoder,” *Bell System Technical Journal*, vol. 45, pp. 1493–1509, 1966.