University Name

Doctoral Thesis

A content-aware interactive explorer of digital music collections: The Phonos music explorer

Author:
John Smith

Supervisor:
Dr. James Smith

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Research Group Name
Department or School Name

February 2015

“Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.”

Dave Barry

UNIVERSITY NAME (IN BLOCK CAPITALS)

Abstract

Faculty Name
Department or School Name

Doctor of Philosophy

A content-aware interactive explorer of digital music collections: The Phonos music explorer

by John Smith

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

Abbreviations

MFCC	**L**ist **A**bbreviations **H**ere
GUI	**L**ist **A**bbreviations **H**ere
HPCP	**L**ist **A**bbreviations **H**ere
MIR	**L**ist **A**bbreviations **H**ere
RS	**R**ecommender **S**ystems
BPM	**R**ecommender **S**ystems
FFT	**R**ecommender **S**ystems

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 The importance of music analysis

The incredible growth of the Web over the last pair of decades has drastically changed many of our habits. One of the areas that have been highly affected by this fast-paced growth is our consumption of multimedia contents: the use of physically-stored content is seeing itself heavily reduced, as we are more and more getting used to the access of huge databases of multimedia content through the Web.
(it would be nice to cite [3] here to present the subject in a more elegant way) Music is one of these fields that have been revolutionized by this trend: the last decade has seen the rise of several Web services (iTunes, Spotify, Pandora, Google Music just to name a few) that offer their users an easy way to access their enormous catalogue of songs. Statistics show an increasing rate of annual growth for each of these services, in both the amount of users and of revenues: now they are among the most used ways of enjoying and discovering music.
However, the transition to this type of services has brought to some new problems. One of them relies on the vastness of these databases: given that users want to easily discover new music suitable to their tastes through intelligently created playlists, a way to reasonably pick songs and artists among the entire catalogue is needed.
This, among others, has been one reason of the rapid growth of **Music Information Retrieval** (MIR), an interdisciplinary research field whose subject is to provide new ways of finding information in music. Main techniques of describing music can be grouped into two categories:

- Metadata (literally *data describing data*), descriptors of music not directly retrieved from the audio signal but instead from external sources [1]
- Audio content descriptors, automatically computed from audio.

When it comes to choosing one method over the other, it becomes clear that both these categories of tools have their own pros and cons. Regarding metadata, major concerns arise from the questionable consistency of the descriptors among the entire catalogue catalogue of music, given that they may have been extracted from several sources. Other concerns also arise from how well they actually describe the audio track. On the other hand, audio content descriptors (especially the low-level ones) may have no musical meaning and therefore they could be hard to understand. Many efforts have be taken in order to improve the methods of information extraction of both these categories. In general, however, audio content descriptors are thought to be more flexible, since they can be easily and equally computed for any track. One advantage of this technique relies on the fact that these kind of descriptors could easily be computed not just for each kind of song, but also for any segment inside of it. This has for example been exploited by *Shazam*, a widely-used smartphone app for music identification that analyzes peaks in the frequency-time spectrum throughout all song length to build a very robust song identification system [1]. Another popular product that performs audio content analysis just for short segments of a song is The Infinite Jukebox[2], a web-application using *The Echo Nest* library and written by Paul Lamere, that allows users to indefinitely listen to the same song, with the playback automatically jumping to points that sound very similar to the current one. The Infinite Jukebox can be considered an application of the so-called *creative-MIR* [6], an emerging area of activity inner to MIR whose subject is to exploit MIR techniques for creative purposes. Other relevant software that exploits Echo Nest library for similar purposes is Autocanonizer[3] and Wub Machine[4]. However, there aren't many commercial or research-based software tools that exploit this kind of techniques for creative interaction or manipulation of audio tracks at the moment. Probably the most relevant commercial system is Harmonic Mixing Tool[5], that performs audio content analysis on the user's music collection in order to allow a pleasant and harmonic fade when mixing between songs. More recently, the research-based software

[1] There is a lack of agreement on the use of the term metadata, therefore its meaning could be different in other resources. For instance, it may be used to indicate all the data describing an audio file, including the ones derived from some computation on the audio signal itself.

[2] `http://infinitejuke.com`

[3] `http://static.echonest.com/autocanonizer`

[4] `http://thewubmachine.com`

[5] `http://www.idmt.fraunhofer.de/en/Service_Offerings/products_and_technologies/e_h/harmonic_mixing_tool.html`

AutoMashUpper has been developed with the intent of automating generating multi-song mashup[6] while also allowing the user a control over the music generated [7]. WRITE MORE ABOUT AUTOMASH HERE

1.2 Phonos Project

Phonos project[7] is an initiative of the **Music Technology Group** (Universitat Pompeu Fabra, Barcelona) in collaboration with **Phonos Foundation**. Phonos was founded in 1974 by J.M. Mestres Quadreny, Andres Lewin-Richter and Luis Callejo, and for many years it has been the only studio of electroacoustic music in Spain. Many of the electroacoustic musicians in Spain attended the courses of the composer Gabriel Brncic at Phonos. It became Phonos Foundation 1982 and in 1984 it was registered at the Generalitat de Catalunya. In 1994, an agreement of co-operation with Music Technology group was established, with the purpose of promoting cultural activities related to research in the music technology. In 2014, an exhibition at Museum de la Musica has been planned, with the purpose of celebrating the 40th anniversary of Phonos and showing many of the instruments used in the studio, while allowing visitors to listen to the music works produced there during all these years. Given the songs' average length and their complexity, a way for the visitors to quickly and nicely explore a catalogue of songs produced in these 40 years was needed.

Figure 1.1: Phonos logo.

1.3 GiantSteps

GiantSteps[8] is a STREP project coordinated by JCP-Consult SAS in France in collaboration with the MTG funded by the European Commission. The aim of this project

[6] A mashup is a composition made of two or more different songs playing together.

[7] `http://phonos.upf.edu/`

[8] `http://www.giantsteps-project.eu/`

is to create the "seven-league boots" for music production in the next decade and beyond, that is, exploiting the latest fields in the field of MIR to make computer music production easier for anyone. Indeed, despite the increasing amount of software and plugins for computer music creation, it's still considered very hard to master these instruments and producing songs[9] because it requires not only musical knowledge but also familiarity with the tools (both software and hardware) that the artist decide to use, and whose way of usage may greatly vary between each other. The GiantSteps project targets three different directions:

- Developing **musical expert agents**, that could provide suggestions from sample to song level, while guiding users lacking inspiration, technical or musical knowledge
- Developing improved **interfaces**, implementing novel visualisation techniques that provide meaningful feedback to enable fast comprehensibility for novices and improved workflow for professionals.
- Developing **low complexity algorithms**, so that the technologies developed can be accessible through low cost portable devices.

Started on November 2013, GiantSteps will last 36 months and the institutions involved are:

- **Music Technology Group**, Universitat Pompeu Fabra, Barcelona, Spain
- **JCP-Consult SAS**, France
- **Johannes Kepler Universität Linz**, Austria
- **Red Bull Music Academy**, Germany
- **STEIM**, Amsterdam, Netherlands
- **Reactable Systems**, Barcelona, Spain
- **Native Instruments**, Germany

[9] "Computer music today is like piloting a jet with all the lights turned off." (S. Jordà). `http://vimeo.com/28963593`

Figure 1.2: GiantSteps logo.

1.4 Purpose of this work

The purpose of this work is to develop a software to be used by visitors during the exhibition *Phonos, 40 anys de música electrònica a Barcelona* and that allows users to easily explore a medium-sized collection of music. This software is intended to exploit latest MIR findings to create a flow of music, composed of short segments of each song, concatenated in a way that the listener can barely realize of the hops between different songs. The application must also allow users to interact with it in order to have some control over the generation of the playlist; specifically, the user should be able to give a general direction to this flow (through some sliders or others GUI elements) in regards to some relevant music features, in a way that the change in the musical output can be perceived. The application developed is meant to be part of the GiantSteps project and therefore should follow the three guidelines explained in the previous page. In addition to this, given its future use on a public place, the application is required to be easy to use also for non-musicians, as many of the visitors of the exhibition could be.

1.5 Introduction to the problem of Playlist Generation

The problem of playlist generation has already been addressed by many popular music platforms, such as *Last.fm* [10], *Pandora* [11] and *Musicovery* [12]. The main objective of such services is to help users find tracks or artists that are unknown to them and

[10] `http://last.fm`

[11] `http://www.pandora.com`

[12] `http://musicovery.com`

Figure 1.3: *Phonos, 40 anys de música electrònica a Barcelona*, Manifesto.

that they may like, providing *personalized radio playlists*. However, a playlist may be defined, in a broad way, just as a sequence of tracks [16] and therefore its use could be more general. For instance, a common use of the term plylist refers to the broadcasting radio playlists, i.e. playlists made by DJs in a radio stations and often involving popular tracks. We can therefore define the problem of playlist generation as follows [16]:

> Given (1) a pool of tracks, (2) a background knowledge database, and (3) some target characteristics of the playlist, create a sequence of tracks fulfilling the target characteristics in the best possible way.

This task is made particularly challenging by the average size of the music database on which the generation of the playlist is needed: already, personal music libraries can be huge [17], hence the corresponding amount of information to be processed in order to the generate the playlist leads to very heavy computational tasks. Depending on the need of the application, these tasks may also be performed offline, although a real-time user interaction should be supported in many cases in order to allow the user to have some control over this generation process (such as in the case study of this work). As we will see in Chapter 2, extracting information from an audio signal is not a trivial task and many algorithms have considerable time-complexity, and this may lead to very long computational times already for the analysis of small-sized catalogues. Playlist generation is a well-known problem inside MIR [18] [19], since this task can be considered as a retrieval task if its definition is limited to the selection of tracks satisfying a user query [16]. Other major topics of MIR also include extraction of features and similarity analysis, that can be seen as a basis for building a playlist generation system [20].

1.6 Structure of the dissertation

This dissertation is organized as follows:

- The first part will at first give an overview regarding music analysis techniques, explaining *metadata*, audio content analysis and the differences between them. Then, common techniques of music similarity computation will be explained.
- The second part will be about the methodology, explaining the different stages of the development, the problems faced and the techniques used. A presentation of the case study will introduce to an explanation of the reasons that lead to prefer the use of some techniques over others.
- Finally, experimental results will be shown, together with some ideas regarding future development of the application.

Part I

Background

Chapter 2

Music analysis techniques: state of the art

The main subject of MIR regards the *extraction and inference of musically meaningful features, indexing of music* (through these features) and the development of *search and retrieval schemes* [2]. In other terms, the main target of MIR is to make all the music over the world easily accessible to the user [2]. During the last two decades, several approaches have been developed, which mainly differ in the music perception category of the features they deal with. These categories generally are: *music content*, *music context*, *user properties* and *user context* [4]. *Music content* deals with aspects that are directly inferred by the audio signal (such as melody, rhythmic structure, timbre) while *music context* refers to aspects that are not directly extracted from the signal but are strictly related to it (for example label [23], artist and genre information [24] [25], year of release [26], lyrics [27] and semantic labels). Regarding the user, the difference between *user context* and *user properties* lies on the stability of aspects of the user himself. The former deals with aspects that are subject to frequent changes (such as mood or social context), while the latter refers to aspects that may be considered constant or slowly changing, for instance his music taste or education [4].
In this chapter, we will focus on the differences between the categories *music content* and *music context*.

2.1 Metadata

By metadata we mean all the descriptors about a track that are not based on the *music content*. Therefore, they are not directly extracted from the audio signal but rather from external sources. They began to be deeply studied since the early 2000s, when first doubts about an upper threshold of the performance of audio content analysis systems arised [5]. Researchers then started exploring the possibility of performing retrieving tasks on written data that is related to the artist or to the piece.

At first, the techniques were adapted from the Text-IR ones, but it was immediately clear that retrieving music is fairly more complex than retrieving text, because the music retrieved should also satisfy the musical taste of the user who performed the query.

The techniques used in this category may differ both in the sources used for retrieving data and in the way of computing a similarity score, and clearly the performance of a system using metadata for similarity computation is highly affected by both of these factors. Sources may include [12]:

- Manual annotation: description provided by experts; they may be referred to genre, mood, instrumentation, artist relations.
- Collaborative filtering data: data indirectly provided by users of web communities, in the form of user ratings or listening behaviour information.
- Social tags: data directly provided by users of social network of music (such as *Last.fm*[1]) or social games.
- Information automatically mined from the Web. Sources in these cases may include web-pages related to music or microblogs (for instance the very popular Twitter).

The availability of some of them greatly depends on the size of the music collection under consideration; for instance, as manual expert annotations might be very accurate, they would be extremely costly and probably infeasible on large collections [8]. In contrast, collaborative filtering data may be the most studied technique, given that it may be applied to other different fields (such as movies or books recommendation) with just little changes. It is the predominant approach in the field of Recommender Systems (RS) [35] and is mainly focused on user ratings, generally leading to better results [11]. However, some concerns are related to this technique. First, collaborative filtering approaches have not been designed to be used for playlist generation, but mainly for recommending artists or music. Second, the availibility of datasets for user ratings in the field of music is very limited compared to other fields, and research is often based on very small samples [36]. Regarding listening behaviour information, they might be inaccurate since they don't keep track of song durations and of the user activities while listening to music [38]. In addition, there's no way of collecting negative feedback (*dislikes*) through them and, more in general, listening to a specific song doesn't necessarily imply liking that song [12].

[1] `http://last.fm`

Sources are picked also in relation to the subject of the research or of the system, that may be for example a recommendation or a similarity computation system. At this point, it's important to highlight the difference between the two of them: a recommendation system not only has to find similar music, but has also to take into account the personal taste of the user, and therefore it's generally considered as a basic tool to produce recommendation [22]. In any case, the terms "similarity" and "recommendation" cannot be substituted, given that a good similarity computation system doesn't necessarily equate to a good recommendation system [37]. The computation of similarity may happen through a Vector Space Model (a technique adapted from the Text-IR), co-occurence analysis or frequent-pattern mining. In the next subsections we will briefly explain the characteristics and the performance of these techniques.

2.1.1 Vector Space Model

The main idea of this technique lies on building a bag-of-words representation [2] of a retrieved document, and then computing a term weight vector for each document. It's a frequently used technique in Text-IR (and in Computer Vision) which can safely be used when retrieving web pages related to music, in an attempt of computing similarity. One of the first work in this field [39] provided an analysis of this kind on music-related web pages retrieved with the queries (to the *Google* search engine) "artist" `music review` and "artist" `genre style`, where words such as music and review where added to improve the chances of automatically retrieving webpages related to music.

2.1.2 Co-Occurence Analysis

2.1.3 Frequent Pattern Mining

2.2 Audio Content Analysis

The main idea behind this kind of analysis is to directly extract useful information, through some algorithms (or library of algorithms), from the audio signal itself. The type of content information extracted may greatly vary in relation to the need of the research, but we can mainly distinguish four categories [12]:

[2] A bag-of-words can be basically seen as an extension of a programming language "dictionary": it collects words (that sometimes may just be an abstraction of much more complex features, such as computer vision descriptors) from a document, and then computes the frequency with which each of them appears in the document. Two different documents are considered similar if they contain the same or similar words with a comparable frequency.

- *Timbral* information: related to the overall quality and color of the sound.
- *Temporal* information: related to rhythmic aspects of the composition, such as tempo or length of measures.
- *Tonal* information: directly linked to the frequency analysis of the signal and to the pitch. It can describe what notes are being played or the tonality of a given track.
- *Inferred semantic* information: information inferred (usually through machine learning techniques) from the previous categories, in the attempt of giving a more defined and understable shape to the data collected. This kind of information may include descriptors such as genre, valence or arousal.

Information extracted through this family of techniques may also be categorized in the following way:

- Low-level data: information that has no musical meaning and that, more in general, is not interpretable by humans. Examples of this kind of descriptors are Mel Frequency Cepstral Coefficients (MFCCs) and Zero Crossing Rate (ZCR).
- Mid-level data: information that has musical meaning but that is related to low-level music features. This kind of category mainly includes temporal and tonal descriptors.
- High-level data: corresponding to inferred semantic information.

Many of the studies conducted on the computation of music similarity through audio content descriptors have solely focused on low-level and timbral information, because this has been proved to bring alone to acceptable results with proper similarity measures [13]. However, more recent studies have shown some evidence of advantages in using high-level descriptors [14] [15] and, more in general, the most performant systems use data from all of these categories. When computing low and mid-level descriptors, the procedure requires the following operations:

- Conversion of the signal from stereo to mono, in order to compute all the descriptors for just one signal
- Down-sampling of the signal to improve the performance while computing the descriptors

- Segmentation of the signal into frames, short segments (usually from 512 to 2048 audio samples). Consecutive frames are usually not disjoint: the so-called *hop-size* determines the hop of samples between the beginning of a frame and the next one, and is normally half or a quarter as big as the *frame size*.
- Computation of Fast Fourier Transform, with an appropriate prior windowing technique [3].

The computation of descriptors is then performed on each frame, and finally a single value for each descriptor is computed by the means of some statistical analysis. Mean, median, variance and covariance are the most used statistical tools for calculating representative global values out of the enormous *pool* of values computed in each frame.
Some more operations may sometimes be needed, such as de-noising [4] of time-scaling of the signal.

In the next sections, a more detailed look among most important descriptors will be given.

2.2.1 Low-level Descriptors

2.2.1.1 MFCC

2.2.2 Mid-level Descriptors

2.2.2.1 Rhythm

In traditional music notation, there are several notations for tempo. It may be expressed in BPM (beats per minute), MPM (measures per minute; commonly used in ballroom dance music) or by semantic notations indicating a range of BPM; an example of this last category of notations may be the popular system of Italian markings, such as *presto* (168-200 BPM), *andante* (84-90 BPM) or *allegro* (120-128 BPM).
In the field of MIR, accurate notations are needed, therefore semantic annotations are disregarded in favour of more precise notation such as BPM and Onset Rate (OR).

Onset Rate

IDEAS: image of ADSR, some flowchart for a standard onset detection algorithm (look

[3] Although this last step may not be strictly seen as a necessary operation, many descriptors rely on frequency analysis of the signal and therefore they require the computation of the Fourier Transform.

[4] A set of operations which purpose is to reduce the amount of background noise in a signal, therefore incrementing the signal-to-noise ratio (*SNR* or sometimes *S/N*).

at figures folder) Onsets are generally defined as the beginning of a new musical note, and onset rate is therefore defined as the number of onsets in a time interval. This definition however hides several difficulties: in polyphonic music, nominally simultaneous notes may be spread over tens of seconds, making this definition more blurred [40]. Moreover, several instruments have a long attack time and this makes the task of defining an onset time even harder.

Several ways of computing an onset detection function have been proposed, according to what aspects are taken into account for defining an onset. Actually, onset detection may be performed in time domain (when looking for significant changes in the overall energy), frequency domain (if looking for events regarding just a specific range of frequencies), phase domain or complex domain. Important algorithms for this task are:

- *HFC*, the High Frequency Content detection function that looks for important changes on highest frequencies. It is very useful for detecting percussive events.
- Spectral Flux, that decomposes the entire audible range of frequencies (approxitamely the interval 20-20000 Hz) into bins, measures changes in magnitude in each bin, and then sums all the positive changes across all the bins.
- the Complex-Domain spectral difference function [41] taking into account changes in magnitude and phase. It emphasizes note onsets either as a result of significant change in energy in the magnitude spectrum, and/or a deviation from the expected phase values in the phase spectrum, caused by a change in pitch.

HFC was proposed by Masri in [42]. Let us consider the short-time Fourier transform (STFT) of the signal $x(n)$:

$$X_k(n) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} x(nh+m)w(m)e^{-\frac{2j\pi mk}{N}} \tag{2.1}$$

where $w(m)$ is again an N-point window, and h is the hop size, or time shift, between adjacent windows. The idea behind HFC is to give more weight to higher frequencies, by defining a onset function whose values are computed in the follwing way:

$$HFC(n) = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} k|X_k(n)|^2 \tag{2.2}$$

The HFC function produces sharp peaks during attack transients and is notably successful when faced with percussive onsets, where transients are well modeled as bursts of white noise [43].

On the other hand, the Spectral Flux SF function is defined as follows:

$$SF(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X(n,k)| - H(|X(n-1,k)|) \tag{2.3}$$

where $H = \frac{x+|x|}{2}$ is the half-wave rectifier function. This algorithm greatly characterizes changes in magnitude spectrum but it quite weak to frequency-modulation phenomena (such as vibrato). To this end, the recently proposed variant SuperFlux [44] seems to achieve much better results.
Another interesting onset function is the *Complex Domain*, that calculates expected the expected amplitude and phase of the current bin $X(n,k)$ based on the previous two bins $X(n-1,k)$ and $X(n-2,k)$. By assuming constant amplitude the expected value $X_T(n,k)$ is then computed:

$$X_T(n,k) = |X(n-1,k)|e^{\psi(n-1,k)+\psi'(n-1,k)} \tag{2.4}$$

and therefore a complex domain onset detection function CD can be defined as the sum of absolute deviations from the target values [40]:

$$CD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n,k) - X_T(n,k)| \tag{2.5}$$

Given an onset function (for instance one of the already cited $HFC(n)$, $SF(n)$ or $CD(n)$), onsets are then extracted by a peak-picking algorithm which finds local maxima in the detection function, subject to various constraints. Threshold and constraints used in the peak-picking algorithm has a large impact on the results, specifically on the ratio of false positives[5] to false negatives[6]. For instance, a higher threshold may lead to a lower number of false negatives but to a higher number of false positive, while a lower threshold may have the opposite effect. A compromise, mostly specific to the application, has to been found.

BPM

The algorithms for detecting the beats-per-minute (generally called *beat tracking algorithms*) greatly rely on onset detection funcions. The basic idea is to look for some time-pattern that may explain the distribution of onsets over time, and hence derive BPM. They usually require more than one onset detection function to achieve good

[5] True onsets that are not detected by the algorithm.

[6] Points that are classified as onsets by the algorithm, while they are actually not.

results. One of the most performant beat tracking algorithm is TempoTapDegara, presented by N. Degara et al. in [45].
EXPLAIN ALGORITHM HERE

2.2.2.2 Tonality

Many efforts have been taken in order to improve the techniques for detecting tonality or harmonic content of a song, as this is one of the most main aspects of western music (a direct consequence of tonality is the detection of predominant melody; to understand why this is so important just ask yourself how many times you whistled or sang a song to let other people recognize it). Many studies have focused on this aspect of music were not oriented toward the computation of similarity between tracks, but instead toward different tasks, such as *automatic trascription of a polyphonic audio signal* (mainly into a MIDI representation) and *source separation*, that is the task of isolating a single and specific instrument among many playing together.
From a music point of view, in western music, an octave is made of 12 different pitches, and seven differents notes take place in this discrete range. According to the pitch assigned to each note, we may have different *keys*, that are a combination of a *tonic* (the central pitch) and the mode. *Major* and *minor* are the most popular modes. (ADD IMAGE OF THESE TWO MODES)
Harmony is a term that denotes the simultaneous combination of notes, called *chords*, and over time, *chord progressions*.
One of the most important descriptor for extracting information related to tonality is called Harmonic Pitch Content Profile (***HPCP***, also called chromagram). This is directly related to tonality and chord detection: chords can be recognized from from the *HPCP* without even precisely detecting what notes are being played, and tonality can also be inferred by *HPCP* (and in this case a previous estimation of chords is not necessary).
An *HPCP* is a $12k$ size vector indicating the level energy for each profile class. If $k = 1$, the *HPCP* represents the intensities of the twelve semitone pitch classes, otherwise of subdivision of these[7]. In [47], Gómez proposes to distinguish tonality features on temporal scales:

- Instantaneous: features attached to an analysis frame.
- Global: features related to a wider audio segment, for instance a phrase, a chorus or the whole song.

[7] It may be extremely useful to study subdivision of semitone pitch classes when dealing with non-chromatic scales, that are very popular in eastern music.

Furthermore, Gómez proposes to split tonal descriptors in both low-level and high-level descriptors. We hence obtain the representation of tonal descriptors shown in Table 2.1.

Name	Temporal Scale	Level of abstraction
HPCP	Instantaneous	Low
Chord	Instantaneous	High
Average HPCP	Global	Low
Key	Global	High

Table 2.1: Main tonal descriptors.

The general approach for computing *HPCP* is indicated in figure PUT BLOCK DIAGRAM HERE and can be summarized as follows:

- At first, some pre-processing of the audio signal may be performed. For instance, a transient detection algorithm may be used to detect and eliminate regions where the harmonic structure is noisy. This step is usually performed to decrease the computational cost of the *HPCP* without affecting its output [48].
- At this point, spectral analysis is required: once the signal is segmented into frames of a proper size and a windowing function is applied, the Fast Fourier Transform (*FFT*) is computed to get the frequency spectrum. Frames should not be too short, in order to have a better frequency resolution.
- A peak-picking algorithm is run to find those frequencies corresponding to local maxima in the spectrum. Usually, these algorithms are not run only on the interval [100, 5000] Hz: this has shown much better results, because outside this range the signal is predominantly noisy, due to some percussion and instrumental noise [47].
- The *HPCP* is finally computed; many approaches have been developed for this task, all based on the pitch content profile algorithm presented by Fujishima in [46]. At first, a mapping of each frequency bin of the *FFT* to a pitch class is needed (for instance, *FFT* bins corresponding to frequencies like 430Hz, 432Hz or 444Hz are mapped to the A at 440Hz). Then, the amplitudes inside each region are summed up and divided by the number of bins inside that region. Finally, the bins obtained are collapsed, so that bins referring to the same note but in a different octave (for example A4 and A5) are collapsed in a single bin for that note, indicating the overall energy of it in the frame. The *HPCP* is different from the PCP in the sense that a higher resolution may be used for *HPCP* bins (decreasing the quantization level to less than a semitone) and a weight function is introduced

into the feature computation. The *HPCP* value of the n-th *HPCP* bin is calculated as:

$$HPCP(n) = \sum_{i=1}^{nPeaks} w(n, f_i) a_i^2 \tag{2.6}$$

where a_i and f_i are respectively the magnitude and the frequency of the ith peak, $nPeaks$ is the number of spectral peaks considered, and $w(n, f_i)$ is the weight of the frequency bin f_i when considering the *HPCP* bin n.

The performance of the *HPCP* builder strongly relies on the weight function [49]. Note that, for how the common procedure of building *HPCP* is defined, *HPCP* are usually considered robust to noise and different tuning references.

HPCP values are usually normalized in order to store the relative importance of the nth *HPCP* bin:

$$HPCP_{normalized}(n) = \frac{HPCP(n)}{Max_n(HPCP(n))} \tag{2.7}$$

Once the *HPCP* are computed, additional tonal features may be computed, such as tonality or chords. Regarding tonality estimation, this is generally computed through a correlation analysis between the *HPCP* obtained and a matrix of *HPCP* profiles corresponding to different keys.

2.2.3 High-level Descriptors

2.2.4 Main Tools For Extracting Audio Content

Many tools are available for the extraction of audio content descriptors from an audio signal. Many of them have been developed by researchers following the research necessities of MIR. This great variety of tools offers support to several operative systems (mainly Linux, Mac OS X and Windows) and programming languages (Java, C++, C, Python, Matlab). Some of this tools may be offered as standalone software or as a Vamp plugin. Not all the tools for extracting audio content are open-source, therefore not being particularly useful for the research community. In the following paragraphs, we'll briefly show the features of the tools taken into account on the development of this work.

Essentia

Essentia[8] is an open-source C++ library of algorithms for audio analysis and audio-based music information retrieval. It has been developed at Music Technology Group[9], Universitat Pompeu Fabra, and has released under the Affero GPL license[10]. In its current version 2.0.1, it contains a large collection of spectral, temporal, tonal, and high-level music descriptors, algorithms for audio input/output functionality, standard digital signal processing blocks and statistical tools. The library can be complemented with Gaia [11], a C++ library to apply similarity measures and classifications on the results of audio analysis. Both these libraries include Python 2.* bindings and support Linux, Mac OS X and Windows. Essentia has been in developed for over 7 years, incorporating the work of more than 20 researchers and developers through its history. It offers two different modes: standard and streaming, the first being imperative while the latter being declarative. Each processing block is offered as an algorithm, and has three different types of attributes: inputs, outputs and parameters. Different blocks may be linked in order to perform the required processing task. In figure INSERT FIGURE a block diagram of a processing task is shown, composed of several different algorithms linked together. See Appendix A for consulting the full list of descriptors provided by Essentia 2.0.1.

The Echo Nest

The Echo Nest[12] is a company that provides access, through Web API, to a collection of audio descriptors for a catalogue of over 36 million songs and almost 3 million artists. In order to access to this database, an API key is required, and some rate limits are imposed to the use of a free license (for instance, the maximum number of HTTP calls for minute is subject to a limit, generally 20). Users can decide to upload their collection into this database, so that descriptors will be computed for new songs and made available to other users. The performance of this library greatly depends on the chance that a song that is about to be analyzed has already been uploaded into this service. If this is not the case, the upload time has to be taken into account for performing the analysis task.
The Echo Nest provides a great amount of descriptors for each track (see appendix B for the entire list), ranging from very accurate audio content information to metadata, and

[8] `http://essentia.upf.edu/`

[9] `http://mtg.upf.edu/`

[10] `http://www.gnu.org/licenses/agpl.html`

[11] `https://github.com/MTG/gaia`

[12] `http://the.echonest.com/`

has been used by several commercial solutions, developed by *Spotify*, *Rdio*, *Warner Music Group* and many others. Many official and unofficial libraries provide access to *The Echo Nest* service; among these, the most important one is probably the official Python library *Pyechonest*[13], that provides full access to all of the Echo Nest methods including artist search, news, reviews, blogs, similar artists as well as methods for retrieving detailed analysis information about an uploaded track. Furthermore, the library *Echo Nest Remix*[14] worths mentioning, as it is a library for audio manipulation and mixing and has been used by many *web-applications*, including The Infinite Jukebox.
However, the source code of *The Echo Nest* API service is not provided, therefore it has little usefulness to the research community.
The Echo Nest has been aquired by *Spotify* on March 2014.

jMIR

jMir[15] is an open-source software suite implemented in Java and intended for use in music information retrieval research. Its development has been guided by Cory McKay (Marianopolis College), with many researchers from several institutions contributing to it. *jMir* is composed of several components differentiated in their scope, spacing from audio content analysis (performed by *jAudio*), to web mining of metadata and machine learning algorithms for classification.
The most relevant components of this suite are as follows:

- *ACE*: Pattern recognition software that utilizes meta-learning.
- *jAudio*: Software for extracting low and high-level features from audio recordings.
- *jSymbolic*: Software for extracting high-level features from MIDI recordings.
- *jWebMiner*: Software for extracting cultural features from web text
- *jSongMiner*: Software for identifying unknown audio and extracting metadata about songs, artists and albums from various web services.

MIRtoolbox

MIRtoolbox[16] is a set of functions for Matlab, dedicated to the extraction of audio content features from audio files. The design is based on a modular framework:

[13] `http://echonest.github.io/pyechonest/`
[14] `http://echonest.github.io/remix/`
[15] `http://jmir.sourceforge.net/`
[16] `https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox`

algorithms are decomposed into stages, formalized using a minimal set of elementary mechanisms, with the objective of offering an overview of computational approaches in the MIR research field. MIRtoolbox has been developed at the Jyväskylän Yliopisto (University of Jyväskylä, Finland), by Olivier Lartillot, Petri Toiviainen and Tuomas Eerola.

2.3 Computing Music Similarity with Audio Content Descriptors

2.3.1 Notable studies on large databases

When large music collections are used, performance of similarity computation algorithms become critic. Although the collection to be used by the system during its public use can't be considered large, the necessary decomposition of it into hundred of thousands excerpt to be analyzed just in few seconds makes performance a critical factor when designing and implementing an algorithm. Therefore, a deep look into studies where large collections were used was needed.

The first content-based music recommendation system working on large collections (over 200,000 songs) was published by Cano et al. in [52], 2005. The system presented on this work relies on rhythmic and timbre features, combined to form a music similarity feature vector. No special indexing technique was used.

The first music recommendation system for large databases using Gaussian timbre features was proposed some months later by Roy et al. in [53]. In this work, they propose to use a Monte-Carlo approximation of the Kullback-Leibler (KL) divergence to measure music similarity. The method proposed is, in principle, similar to the one proposed by Schnitzer et al. in [51], which has been used on the development of this thesis work (see 5.2). To pre-filter their results, Roy et al. increase the sampling rate of the Monte-Carlo approximation. As the divergence values converge, they are able to reduce the number of possible nearest neighbors. This method has shown good performance, both in query time and results.

A different attempt of improving performance was proposed by Levy and Sandler in [54] where they use only diagonal covariance matrix instead of a full one to compute music similarity. While this has shown a ten-fold speedup compared to the full Kullback Leibler divergence, the quality of this simpler similarity measure results in worse genre classification rates.

2.4 Conceptual Differences Between Metadata and Audio Content Information

The performance of content-based approaches is considerably lower [9]. It is challenging to try to make the so-called *semantic gap* smaller [10]

The advantage of relying on the audio signal over, say, expert annotations, is that the process is objective and can be automated to a large extent. However, extracting the features can be computationally costly [21]. Another limitation is that there might be features like the release date, the "freshness," or popularity of a track, which can be relevant in the playlist generation process but that cannot be extracted from the audio signal [22].

When used in an automated process, data completeness and consistency are crucial. Another potential problem is that not all types of metadata are objective, and annotations regarding, for example, the mood or the genre of a track can be imprecise or inconsistent [28].

(speaking of tags) Although such annotations can be rich and diverse, the perception of music is again subjective and can even be influenced by the perception of other people [29]; tags only for popular songs [28]

When dealing with track ratings: grabbed from a wall posting on Facebook [30] or a tweet on twitter [31], 1-to-5 rating scales like on iTunes. Challenges: problem of data sparsity (especially for the tracks from the long tail), a positivity bias (the phenomenon that most of the ratings are highly positive and negative feedback is rare [28]).

Chapter 3

Assessing the performance of a music similarity computation system

3.1 Literature Review

The coherence of the tracks is a typical quality criterion for playlists [32]. Therefore, selecting and ordering tracks based on their similarities is an obvious strategy to generate playlists. The core of any similarity-based approach is its distance function, which characterizes the closeness of two tracks. How the distance function is actually designed depends on the available data, which could include the raw audio signal along with the features that can be derived from it, but also metadata, such as the artists, the genres, playcounts, or ratings [Slaney and White 2007]. In many cases, a signature or model of each track is determined first, in which the distance function is then applied. Typical examples for such functions applied on more abstract models of a track's features are the earth-mover's distance [32], the Kullback-Leibler (KL) divergence [33], or the Euclidean distance [34].

See section 5.2 of [16] for a background on how to assess the quality of a playlist: user studies, log analysis, objective measures, comparison with handcrafted playlists

Part II

Methodology

Chapter 4

Requirements and approach

4.1 Catalogue of music

The catalogue of music provided features 584 songs, for a total length of 91 hours, 43 minutes and 35 seconds. The average length of each song is 9 minutes and 25 seconds circa. This catalogue has been provided with metadata indicating only artist, year of release and title of each song. Furthermore, all of these work can be labelled as belonging to the electro-acoustic genre, which usually indicates very abstract and arrhythmic, for which is difficult to provide semantic descriptors or tags. Given this latter feature of the music and the length of the entire catalogue, the possibility of manually annotating it with proper metadata has been soon disregarded. This collection of music has therefore represented a great chance for developing a system based on the latest findings on audio content analysis.

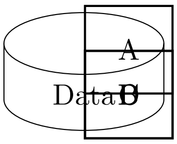
4.2 Requirements

Despite its intended use as part of the exhibition *"Phonos, 40 anys de música electrònica a Barcelona"*, the software developed should feature good flexibility to different catalogues of music, in order to be exploited as a part of the research for the GiantSteps project. This has represented a strong requirement during the development, and has induced the adoption of several descriptors that may not be particularly meaningful for the Phonos catalogue of songs, but that extend the range of possible music catalogues in which the system performance could be satisfactory. Furthermore, as a part of a research project, the system developed should be easily extendable in other research activities, hence a modular, coherent and well-document code is preferred.
The software is intended to be used at the exhibition through an interactive kiosk: it will

be available to users as a link inside a more general *webpage* containing several information regarding Phonos history and instrumentation. In addition, it must fully support touch devices, provided that this will be the only way users will be able to interact with the application.
All of these requirements have lead to the choice of developing a *web-application*.
Anyways, the interactive kiosk to be used at the exhibition was not available during the development; furthermore, its technical specification was unknown. For these reasons, it was therefore decided to develop a *two-layers system* made of the interactive kiosk machine connected (by an Ethernet cable) to a server machine. The latter one is in charge of providing and executing all the complex functions required during the functioning of the system.
Furthermore, the system must react to the real-time interaction of users with the user interface. Computation times must hence be as low as possible, in order to avoid a notable and inconvenient delay between the user interaction and the effective perception of changes in the flow of audio.
Finally, given the substantial average length of the songs, the system should segment the songs into very short excerpts (from 2 seconds to around 30; the choice of this length should be available to users in a real-time fashion), in order to allow users to listen to as many works as possible during the visit at the exhibition and to more easily find tracks that fulfill their taste. It must then be found a way to properly segment the audio pieces and computing descriptors for each slice obtained. In order to achieve a better sense of *"flow of music"*, the computation of similarities should therefore be carried out between these short excerpts, instead of exploiting descriptors for the entire songs.

4.3 Design of the system

The requirements cited above have lead to the following choices for the design of the system. First, the computation of audio descriptors can be performed offline, because the catalogue of music on which the system will run is not subject to changes. It is therefore safe to compute descriptors prior to the public use. The performance of the system will greatly benefit from this choice, given that the computation of audio descriptors for each excerpt of every song of the catalogue is the most computationally intensive step to be performed. The audio descriptors will be stored on the server machine.
Second, for the system is intended to have low response times to user inputs, the computation of music similarity is being carried out on the server (for the performance of the interactive kiosk machine are unknown, as already cited in 4.2), with proper music similarity algorithms. The flow of music is not supposed to require an human interaction,



to the meaning that it will automatically generate a flow of music based on the computation of audio similarity also without an interaction of the user. Actually, the system always concatenates segments in a way that only very similar segments are consecutive elements of the playlist. The interaction of the user will eventually give a direction to this flow, according to the user's will and taste.

Third, the application running on the server machine will be in charge of collecting the user interaction with the web-application running on the interactive kiosk machine, and that will come in the form of *HTTP POST requests*. At each user interaction, the application running on the server machine deletes the current and already computed playlist and performs an audio similarity computation between the currently playing excerpt and a set of excerpts that fulfill all the requirements about music that the user has imposed through the graphic user interface. One of the most similar excerpt is taken from the list and a new content-aware playlist starts being built above that.

4.4 Evaluation

Chapter 5

Off-line computation of audio features

In order to achieve good performance, two very computationally intensive tasks of the system are performed off-line, and their output is then going to be used by the real-time application. These tasks consist of the computation of the audio content descriptors and of the building of a *fast-map*, a high dimensionality space in which each point correspond to an excerpt. This space is built in a fashion that guarantees that nearby points of this space correspond to very similar excerpts.

5.1 Audio content features extraction

Solving this problem has involved two very important choices: what audio content descriptors to use and what library or tool to use for computing them.
Many factors have been taken into account for solving both of these problems.
Among the features of the tools, flexibility has constituted the strictest requirement: an easy way to compute descriptors for each excerpt of every track is required, while many tools provide only ways of computing descriptors for the entire file. In this latter case, the file should manually split into *subfiles* (one for each segment), therefore implying a huge waste of memory. This has soon lead to the exclusion of *jMir*, for it doesn't fulfill this requirement.
Second, the tool should easily be callable by source code or bash scripts, and results of the analysis must be stored in output files.
Third, the computation of descriptors should be as fast as possible, given that the excerpts to be analyzed are in the order of tens of thousands.
Last but not least, the tool must provide descriptors whose usefulness for this specific

case study has been empirically verified during the development of the system.
All of these requirements lead to the choice of performing the audio analysis with Essentia and Echo Nest: the first for its speed, flexibility and reliability. Echo Nest has been used for some of its descriptors are not present or not as accurate in Essentia, and have shown a great usefulness during the development.
Furthermore, both of the two libraries are offered in Python, allowing the entire analysis task to be written in a single programming language, therefore improving the code consistency and readability.
The schema for the extraction of the audio features is illustrated in figure 5.1.
At first, the user is required to give the path of the folder in which the audio files are stored. The collection is entirely stored as .mp3 files with a sample rate of 44100Hz and a bitrate of 192kbps. The application then collects the path to all the .mp3 files in this folder, and mark them as to be analyzed if no previous analysis has be performed. An analysis of these files with Echo Nest (through Pyechonest) is performed, and we specifically use the following fields of the output of this analysis: *bars*, *BPM*, *loudness*, *HPCP* and *acousticness*. *Bars* give the starting and ending point of each bar detected and, although not particularly meaningful for the arhythmic Phonos catalogue of music, have shown to perform well on the additional and more generic personal catalogue used during development testing; therefore, it was decided to use them in order to improve the flexibility of the system.

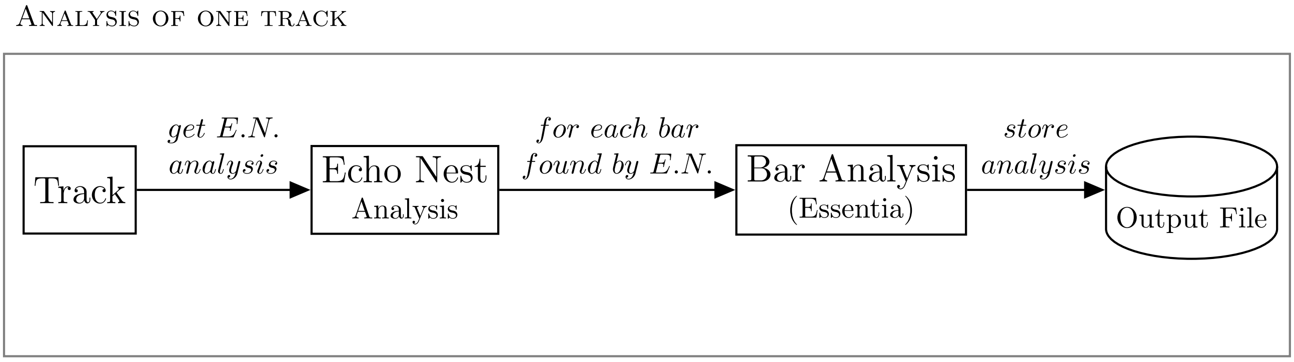


Figure 5.1: Schema for the extraction of audio features.

Segmentation of songs into excerpts is then performed, based on starting and ending point of each bar. Then, we compute more specific descriptors with Essentia for these excerpts, with the following strategy:

- each excerpt is divided into frames, with a size of 2048 samples and a hop of 1024 samples. For each of these frames:
    - we apply an Hann windowing function

  - we apply the FFT algorithm provided by Essentia in order to get a spectral representation of the signal
  - we look for peaks in the spectrum, collecting their frequencies and magnitudes, and then we use them to compute the dissonance in the frame, with Essentia's algorithm `Dissonance`
  - an HFC onset function is computed on the spectrum, that will be used afterwards to compute the onset times
  - the MFCC bands and coefficients are computed with Essentia's algorithm `Mfcc`[1]
  - the energy in 27 Bark bands of the spectrum is computed

- onset times in the excerpt are calculated, according to the onset function computed in each frame, and then onset rate is calculated with the formula:

$$OR_{excerpt} = \frac{Onsets_{excerpt}}{Length_{excerpt}} \tag{5.1}$$

- dissonance in the excerpt is computed as a mean of the dissonance in each of its frames

- a single Gaussian model for the collected MFCC values is computed. Specifically, we collect its mean, covariance and inverse covariance. Mean is a 13 size vector, while covariance and inverse covariance are 13x13 matrices. The inverse covariance is stored in order to prevent having to compute it in the real-time application or during the fast map computation, therefore increasing the performance of both these stages. If a problem of ill-conditioned covariance matrices is encountered (i.e., a not positive semi-definite covariance matrix has been computed), only values of the diagonal of these problematic covariance matrices are used. This has allowed to avoid the presence of outliers when computing similarity, while still taking into account excerpts for which a covariance matrix of the MFCC values could not be correctly computed.

- based on the HPCP values computed by Echo Nest, we use Essentia's `Key Detector` to associate a key to each first and fourth beat of the bar. The reason why we keep values for these two particular beats is that this allows us to perform a more precise tonal comparison when trying to merge two excerpts in the real-time application: if the key of the first beat of the inspected excerpt is very different from the key of the fourth beat of the excerpt for which we're looking for similar pieces, the candidate is discarded.

[1] Essentia uses the MFCC-FB40 implementation, which decomposes the signal into 40 bands from 0 to 11000Hz, takes the log value of the spectrum energy in each mel band and finally applies a Discrete Cosine Trasform of the 40 bands down to 13 mel coefficients.

This procedure is repeated for each excerpt, in order to get a deep description for all of them and perform more precise similarity computation in the real-time application. In addition, we store some additional level-song descriptors, specifically artist, title and year of release, and acousticness (computed with Echo Nest). Finally, for each song we create a corresponding JSON file in which we store all the descriptors computed.
The list of descriptors computed during this task is summarized in table 5.1.

Features	**Source**	**Level**	**Motivation**
Title, Artist, Year	Provided	Song-Level	Display more information about the current playing track in the GUI
Acousticness	Echo Nest	Song-Level	Give the user the chance to filter music in regards to its nature (acoustic or electronic music)
MFCC	Essentia	Bar-Level	Timbre similarity computation
BPM	Echo Nest	Bar-Level	Avoid consecutive excerpts with very different BPM
Onset Rate	Essentia	Bar-Level	Give the user the chance to filter music in regards to the presence of percussive elements
Dissonance	Essentia	Bar-Level	Give the user the chance to filter music in regards to the dissonance[2] of excerpts
Loudness	Echo Nest	Bar-Level	Give the user the chance to filter music in regards to its loudness
Bark Bands	Essentia	Bar-Level	Give the user the chance to filter music in regards to its “sparseness”, i.e. the amount of mel bands with significant energy level
HPCP	Echo Nest	Beat-Level	Use them to compute key
Key	Essentia	Beat-Level	Use them to discard the possibility of having two consecutive dissonant excerpts in the playlist

Table 5.1: Descriptors computed by the offline application.

[2] During development, it has been empirically noticed that dissonance has a significant correlation to the perception of noise: the more dissonant an excerpt is, the more it is perceived as noisy.

5.2 FastMap computation

The procedure just described for computing descriptors give us a 410 size vector for each excerpt, and a total number of 159239 excerpts.

In order to achieve good real-time performance when comparing these excerpt, a dimensionality reduction of these vectors is required. Furthermore, as seen in 2.3 the computation of Kullback-Leibler divergence, although showing very good results in capturing the timbre similarity, is a very intensive computational operation and therefore a simpler distance measure with comparable results is preferred.

These requirements were faced also in [51], in which Schnitzer et al. present a filter-and-refine method to speed up nearest neighbor searches with the Kullback-Leibler divergence for multivariate Gaussians, yielding high recall values of 95-99% compared to a standard linear search. The original FastMap was proposed in 1995 by Faloutsos and Lin [57] for indexing and data-mining multimedia datasets. It was used for the first time for computationally heavy, non-metric measures and nearest neighbor retrieval in [55], for speeding up classification of handwritten digits. FastMap was used for the first time in MIR by Cano et al. in [56] in the attempt of reduing high dimensional music timbre similarity space into a 2-dimensional space. This was done not for speeding up classification, but rather for visualization purposes.

The idea behind the use of a FastMap for classification or computing similarities is to compute with the original distance measure $D()$ (computationally intensive) just a subset of all the distances, specifically the distances between each point and a subset of $2k$ points (the *"pivots"*); then, on the basis of these distances computed, each feature vector is mapped with a non-linear trasformation into a point of a k-dimension space, where a simpler distance measure can be applied, with a little loss of accuracy.

For choosing the $2k$ pivot elements, the original FastMap [57] follows this strategy:

- k element $x_1^1, x_2^1, ..., x_k^1$ are randomly selected from the collection of feature vectors
- for each x_i^1, its corresponding most distant object x_i^2 according the original distance measure $D()$ is picked

Each vector of features x is then mapped into the point $(F_1(x), ..., F_k(x))$ of the new k-dimensional space, where $F_j(x)$ is computed with the formula:

$$F_j(x) = \frac{D(x, x_j^1)^2 + D(x_j^1, x_j^2)^2 - D(x, x_j^2)^2}{2D(x_j^1, x_j^2)} \tag{5.2}$$

In other words, the coordinate in the $j - th$ dimension of each point is determined by the pair (x_j^1, x_j^2), specifically by the original distance (computed with $D()$) of the point from both these pivots and the distance between the pivots themselves.

For our work, we have decided to use the Kullback-Leibler as the original distance function, computed with the closed formula:

$$KL(x_1, x_2) = \frac{1}{2}\left(tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1) - k + ln\left(\frac{det\Sigma_2}{det\Sigma_1}\right)\right) \tag{5.3}$$

where tr is the $trace$ operation of a matrix (i.e., the sum of the elements in its main diagonal), μ_i the mean vector of the MFCC values of point i, Σ_i is its the covariance matrix, Σ_i^{-1} its inverse covariance matrix, and k the size of μ_i.

This formula has been used in the state-of-the-art similarity computation system and its accuracy in measuring the timbre similarity between MFCC values is no longer questionable. Anyways, we must take into account several aspects.

As already seen in 2.3, the Kullback-Leibler cannot be intended as a pure distance measure, for it fails to be symmetric and to fulfill the triangle inequality. It can simply be made symmetric by considering the distance SKL (symmetric Kullback-Leibler) defined as:

$$SKL(x_1, x_2) = \frac{1}{2}KL(x_1, x_2) + \frac{1}{2}KL(x_2, x_1) \tag{5.4}$$

Regarding the triangle inequality, a proper solution is not that trivial. However, in [51] Schnitzer et al. have shown that rescaling the symmetric Kullback-Leibler divergence with the square root leads the new distance function to fulfill the triangle inequality in more than 99% of the cases. Therefore our original distance function $D()$ that we use in equation 5.2 is:

$$D(x_1, x_2) = \sqrt{SKL(x_1, x_2)} = \sqrt{\frac{1}{2}KL(x_1, x_2) + \frac{1}{2}KL(x_2, x_1)} \tag{5.5}$$

This procedure can be further improved by a little modification in the strategy for choosing pivots: once the pivot x_i^1 is randomly picked, we choose to pick as x_i^2 the object lying at the distance media, i.e. the object at the index j=$\lfloor\frac{N}{2}\rfloor$ once all the distances from point x_i^1 are sorted. We've decided to use $k = 20$ (therefore having 20 pairs of pivots and a final 20-dimensional space) as this has allowed us to find a good balance between computational times and quality of the output the similarity computation.

The accuracy and performance of this procedure are well-documented in [51]. This technique constitutes the basis on which our system will perform the real-time similarity computation allowing excellent performance, with some additional tweak that will see in the Chapter 6.

The computed data is stored on a JSON file: for each point (corresponding to an excerpt), we store its coordinates in the new 20-dimensional space plus some additional descriptors that allow us to do a faster filtering in the real-time application, as we won't

need to lookup to the original JSON descriptor file for each song just for retrieving the values of these descriptors. The list of features stored in the map for each point is shown in table 5.2, while we propose also an illustrative figure in Figure **??**. During this stage, we additionaly save lists that associate each segment to the decade the song it has been extracted from has been produced. This will allow very fast filtering techniques on the real-time application when the user interacts with the sliders for selecting music according to the year of release.
The computational times of this stage are shown in table 5.4 and the configuration of the computer used in table 5.3.

Features	**Motivation**
Year, Artist, Title	Speed up access to information
Starting and ending point inside the track	Allows fast extraction of the excerpt from the entire audio signal
BPM, Key	Be faster when filtering out music with very different BPM or key
Acousticness, Loudness, Dissonance, Bark Bands, OR	Perform a fast filtering of database of excerpt when the user interacts with the GUI for controlling the musical output

Table 5.2: Features stored in the map for each point.

Laptop Model	Packard Bell EasyNote TS-11HR
CPU	Intel®Core$^{\text{TM}}$i5-2410M @ 2.50GHz
RAM	4GB DDR3 @ 1066MHz
Hard Disk Drive	5400rpm
OS	Linux Mint 17.1 “Rebecca” (64 bit)

Table 5.3: Hardware configuration of computer used during off-line descriptors computation.

Stage	**Time required**	**Stats**
Descriptors computation	04h 32m 25s	Minimum time for track: 00m 15s
		Maximum time for track: 00m 52s
		Average time for track: 00m 28s
FastMap computation	00h 47m 12s	Choosing pivots: 16m 43s
		Computing points coords: 30m 29s
Total	05h 19m 37s	

Table 5.4: Computational times for descriptors computation of a collection of 584 tracks (the time for uploading these tracks to Echo Nest is not considered in these results).

Chapter 6

The real-time application

The real-time application is based on a two-tier architecture, organized as follows:

- the server machine runs a Python Flask application, and it's responsible for generating playlist and audio
- the client displays an HTML web-page that collects user interactions and sends them to the server machine for realtime editing of playlist. Additionally, it receives audio streaming from the server.

Therefore, the realtime computation of music similarity happens on the server machine.

6.1 The server application

As already stated above, the server application is in charge of offering several features: it generates the playlist, sending audio and additional information to the client (such as *artist* and *title* of current playing piece, so that the client can display them for the user on the GUI). Additionally it has to generate audio, that will be streamed to the client in order for the user to listen to it through its own device. For generating the playlist, a realtime music similarity algorithm with very good performance must run on the server.
Many Python web frameworks are available; the most used ones are Django[1], Flask[2] and Pyramid[3]. This realtime server application has been based upon Flask framework, that

[1] https://www.djangoproject.com/

[2] http://flask.pocoo.org/

[3] http://www.pylonsproject.org/

Figure 6.1: Flask logo.

is a lightweight web application framework written in Python and based on the WSGI toolkit[4] and Jinja2 template engine[5]. It's provided with a BSD license and, contrarily to Django and Pyramid, is aimed at small applications with simple requirements. Its first version was released in 2010 and it comes with a great usability, where a simple *"Hello World"* web-app can be written with only 7 lines of source code[6]. Web application framework are usually thought to be separated into several conceptual units called "apps", each one providing different functionalities to the system. Flask is intended to make really simple the development of a single app; many others may be added, but in this latter case Django and Pyramid may provide a better ease of use.

All of these factors have lead to the choice of this framework for our system: the web platform to develop is actually intended to be quite simple, displaying just the main GUI and a few more details and options for the user. Given that the application is meant to be offered just to one client at time, we decided to use the builtin server of Flask also on production; indeed, we considered a full deployment option (such as Apache or CGI) to be a waste of resources for this simple use case. The server application executes two parallel tasks: the generation of the playlist, based on realtime computation of music similarity, and the generation and streaming to the client of the audio of this playlist. It furthermore provides several methods that are handled by Flask routing techniques and invoked at specific interaction of the user with the client application; these methods have deep impact on the generation of the playlist and allow the user real-time control over this process.

[4] A specification for universal communication between web servers and web applications or frameworks for Python programming language. Published on December 2003 by its author Phillip J. Eby, it has become a standard for Python web application development.

[5] `http://jinja.pocoo.org/docs/dev/`

[6] `http://flask.pocoo.org/docs/0.10/quickstart/#a-minimal-application`

6.1.1 Realtime computation of music similarity and playlist generation

As we mentioned, this computation is performed on the server machine, for the hardware configuration of the interactive kiosk has been unknown until the beginning of the exhibition, and may have been not able to achieve good performance with the software developed. The hardware configuration of the server machine is shown in table 6.1.

CPU	Intel®Core^{TM}2 Quad Processor Q6600 @ 2.40GHz
RAM	2GB DDR2 @ 800MHz
Hard Disk Drive	5400rpm
OS	Linux Mint 17.1 “Rebecca” (32bit)

Table 6.1: Hardware configuration of the server machine.

The task for generating the playlist follows a well-defined schema.
At first, the FastMap computed as described in section 5.2 is loaded into memory; this process usually takes just few seconds. A random point of this map is pick, and will be used as the first excerpt of the playlist. This excerpt in then put inside the playlist, a Python dictionary whose keys are the position of the elements inside the playlist and the corresponding values are tuples containing several important aspects for the playback; the details of these tuples are shown in table 6.2.

URI of file	Song title	Song artist	Song Year	Starting time	Ending Time

Table 6.2: Information stored for each element of the playlist.

Once the first segment is picked, the application enters in a loop in which each iteration ends in adding a new excerpt to the playlist. The comparison of music similarity is always performed between all the candidate elements of the FastMap and the last element of the playlist. The procedure invoked in this loop can be summarized as follows:

1. If any user interaction with sliders or knobs has happened since the last iteration, delete the content of playlist. This allows the user to immediately hear musical differences in the playlist as soon as he interacts with the client application.

2. Delete already played elements from the playlist in order to avoid memory leaks

3. If we already have enough elements in the playlist, let the task *"sleep"* for one second and then go back to step one. This prevents the cpu from always working at full load, a behaviour that could cause serious overheating problems in a server machine running this application for several consecutive hours at the museum.

4. At this point, we get into the procedure for actually choosing the next excerpt to be inserted into the playlist. At first, a weighted queue according to the sliders for filtering by decades is created.

5. The entire map of excerpts is now filtered according to the current positions of sliders in the client application. If there is no excerpt fulfilling all the constrainst imposed by the sliders, we only take the segments whose descriptors values fulfill less strict thresholds based on actual sliders values. If instead the amount of excerpts available after this filtering is over 500, a Monte Carlo sampling of them is performed, to bring the total number of candidates to 500. We experienced unsatisfying performance of the application during successive steps of the procedure (also due to a not particularly powerful configuration hardware of the server machine) with less aggressive sampling, and we noticed that with 500 candidate excerpts good results were still achieved. This value may be increased in more powerful devices.

6. Additional filtering is performed, based on the values of BPM and loudness of the candidates. Candidates who greatly differ on these values from the last element of the playlist are discarded. For judging similarity in terms of BPM, the formula **??** (with $\alpha = 1$) has been used, with a maximum distance allowed of 3. The maximum discrepancy allowed in loudness is of 5dB. If no candidate excerpt fulfill this stage, the list of candidates before this filtering is restored.

7. At this point we finally choose the number of candidates in which we'll perform deeper analysis. This number, that we call $N_{Neighbors}$, is computed according to the following formula:

$$N_{Neighbors} = filter\_size * |FastMap| \tag{6.1}$$

   where $|FastMap|$ is the number of excerpts in the FastMap (i.e., the total number of excerpts in the catalogue), and $filter\_size$ is a value in $[0, 1]$. We empirically noted that a value of 0.1 for $filter\_size$ already gives good results, while allowing to achieve highly satisfactory computational times. We then select the $N_{Neighbors}$ nearest neighbor to the current element through an Euclidean distance on the 20-dimensional space.

8. We now compute the symmetric Kullback-Leibler distance between the last element of the playlist and all its neighbors. We do this specifically only if:

   - We have a margin of at least 5 seconds of playback in the current playlist after the current playing excerpt
   - The user has not interacted with the controllers of the client-application since the last iteration of the loop

   If any of this two condition is not met, we don't compute the symmetric Kullback-Leibler distances but we rather choose the next element of the playlist on the basis of the euclidean distance on the 20-dimensional space. We do this because this stage could require several seconds (around 4 to 9 seconds on the server machine[7]) and the conditions for performing such a slow computation could not be met, resulting in a perception of a high-latency system. The second condition is used because, even if the playlist is emptied as soon as the user interacts with the controllers (but there still may be more than 5 seconds to play, if the current excerpt is very long), it doesn't make sense for us to perform computational intensive task for computing similarity when the user's will is actually to change the flow of the music by interacting with the controllers.
   Once all the distances are computed, we keep only the segments whose SKL distance from last element in the playlist is less than 20, a threshold that we empirically noticed to be quite selective in the quality of the output despite not being extremely selective in the amount of results. An excerpt from this list is finally randomly picked and put in the playlist. If the list is empty (or the computation of symmetric Kullback-Leibler couldn't be performed), the next excerpt of the playlist is randomly picked among the 10 nearest neighbors by the mean of the Euclidean distance.

The procedure described allows to choose the next element of the playlist with satisfying performance (see Chapter 7), although this may greatly vary with the condition; specifically, computational times become much longer when all the symmetric Kullback-Leibler distances are computed, but this generally leads to better musical results.
It may be useful to mention two further features of the application:

[7] This considerable amount of time is due not only to the complexity of the formula for computing the symmetric Kullback-Leibler distance, but also to the necessary access to JSON files, where the needed MFCC values are kept.

- When the user interacts with the slider for changing the length of the excerpt to be played, the procedure for computing similarity doesn't change. Longer segments are obtained by playing consecutive excerpts of the same song, and the procedure for computing similarity will look for similar excerpts to the last one in this queue of consecutive excerpts of the same song.
- The software provides options for managing the playlist generation in regards to repetition of songs or excerpts: specifically, the user can force the application of never picking two excerpts belonging to the same song unless a specific amount of different excerpt in the playlist has already put between them. We noticed that disabling this feature may greatly improve the quality of the musical flow (some loops between excerpts of the same song may generate, creating a strong cohesion of the musical output; this behaviour is the same one proposed by *The Infinite Jukebox*) but may annoy some users if they want to explore the collection of music and would possibly like to avoid repetitions.

6.1.2 Audio generation and streaming

Everything we have seen so far allows to dinamically generate a content-aware playlist of excerpts. To allow the user to actually listen to this playlist we need to read the corrisponding slices of the audio files and implement a streaming over the network of this audio content.

Feature	**Motivation**
Seek by millisecond	Perform very accurate extraction of excerpts from audio tracks, in order to perform beat synchronized track mixing
Audio Crossfade	Improve the audio "flow", making the transition between consecutive excerpts less abrupt
Programmable	Facilitate communication with the code for computing music similarity. Python preferred.
Streaming	Streaming over the network is required for the user to listen to the playlist.

Table 6.3: Requirements of the audio player.

This is not a trivial task, for not many audio players on Linux provide the needed flexibility by the application. Specifically, it has been found no audio player on this platform that simultaneously provides all the needs reported on Table 6.3. Therefore, we decided to build our custom audio player, exploiting the very popular multimedia framework *GStreamer*.

GStreamer

GStreamer[8] is a free and open-source multimedia framework written in the C programming language, subject to the GNU Lesser General Public License (LGPL). It allows developer to modularly build multimedia applications with the use of *pipelines*, where lower-level units are connected; each unit has a specific purpose. It fully supports Linux, Android, iOS, Mac OS X and Windows, and offers bindings in several programming languages, Python included. The list of popular applications built upon this framework includes *Amarok*[9], *Banshee*[10], *Flumotion*[11], *Pitivi*[12], *QuodLibet*[13] and *RhythmBox*[14].

The main advantage in the use of this framework lies in its modularity: it offers many units (also called *plugins*) with media-handling features, including audio and video playback, recording, streaming and editing. The pipeline design serves as a base to create many different types of multimedia applications, for instance media players, video editors, and streaming media broadcasters.

It fulfills all the requirements of Table 6.3 and therefore we decided to use it for developing our custom audio player.

FIGURE 6.2: GStreamer logo.

[8] http://gstreamer.freedesktop.org/

[9] https://amarok.kde.org/

[10] http://banshee.fm/

[11] http://www.fluendo.com/

[12] http://www.pitivi.org/

[13] https://code.google.com/p/quodlibet/

[14] https://wiki.gnome.org/Apps/Rhythmbox

Audio player developed

For we want to smooth the transition between two consecutive excerpts, the use of a crossfade is preferred. This implies that two different audio players should be playing simultaneously when the crossfade is being performed. We solved this by creating a simple audio player (the custom bin shown in Figure 6.3) for each track that is then connected in a global pipeline (Figure 6.4) responsible for the audio synchronization of different custom bins and of the streaming over the network of the audio content.
The units used in the custom bin are explained in Table 6.4, while the ones used in the global pipeline are explained in Table 6.5.

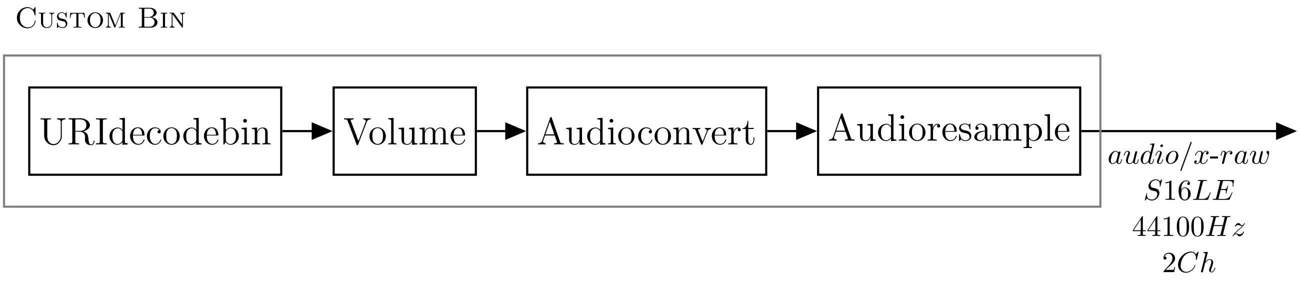


Figure 6.3: Custom audio bin, that corresponds to an audio player only responsible for the playback of a single excerpt.

Unit	**Input**[15]	**Output**[15]	**Motivation**
URIdecodebin	*mp*3 file	*audio/x-raw* $S32LE$, $2Ch$ $44100Hz$	Loads the raw audio content of a file by its location (URI)
Volume	*audio/x-raw* $S32LE$, $2Ch$ $44100Hz$	*audio/x-raw* $S32LE$, $2Ch$ $44100Hz$	Used in crossfades, allows fade in and fade out on single audio tracks
Audioconvert	*audio/x-raw* $S32LE$, $2Ch$ $44100Hz$	*audio/x-raw* $S16LE$, $2Ch$ $44100Hz$	Negotiates a raw audio format according to formats supported by its end and the format of the input
Audioresample	*audio/x-raw* $S16LE$, $2Ch$ $44100Hz$	*audio/x-raw* $S16LE$, $2Ch$ $44100Hz$	Needed by the adder to ensure that its input files will always be of the same type

Table 6.4: Elements of the custom bin.

[15] Values shown here are related to particular files of the Phonos catalogue of music used by the system, and they have been inserted just as examples. Their values may vary with different types of files.

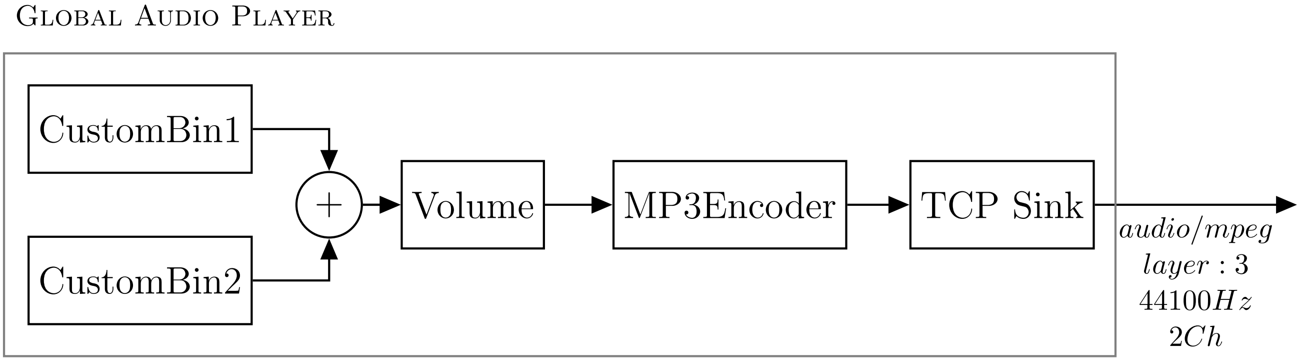


Figure 6.4: Schema for the audio player implemented.

Unit	**Input**[15]	**Output**[15]	**Motivation**
Adder	*audio/x-raw* $S16LE$, $2Ch$ $44100Hz$	*audio/x-raw* $S16LE$, $2Ch$ $44100Hz$	Mixes together samples coming from multiple audio streams, producing a single audio stream
Volume	*audio/x-raw* $S16LE$, $2Ch$ $44100Hz$	*audio/x-raw* $S16LE$, $2Ch$ $44100Hz$	Gives control over the global volume. This will be settable by the user on the client application GUI
MP3Encoder	*audio/x-raw* $S32LE$, $2Ch$ $44100Hz$	*audio/mpeg* $layer3$, $2Ch$ $44100Hz$	Converts the raw audio stream into an mpeg layer 3 stream
TCPSink	*audio/x-raw* $layer3$, $2Ch$ $44100Hz$		Provides streaming over the network of the mpeg audio content

Table 6.5: Elements of the pipeline.

The class responsible for handling the global audio player has access to the playlist generated by the algorithm explained in Section 6.1.1. It extracts the first element on this queue, creates a custom bin for it, performs the seeking[16] and plays it with an initial fade in, whose length is `CROSSFADE`[17]. `CROSSFADE` seconds before the end of the current excerpt, the algorithm extracts the next element on the playlist. If this is empty, we

[16] Seeking is actually performed on the URIdecodebin element.

[17] The default value is 0.8s, enough for creating a sense of music "flow". The user can edit this value through the client graphical user interface.

keep playing the current track until a new excerpt is inserted into the playlist. The algorithm then creates a new custom bin for this new excerpt, adds it to global pipeline, performs the seeking and starts the playback of this custom bin with a fade in. The seek sets the inpoint of the playback to the point `(start_point`[18] `- CROSSFADE)`, so to have a beat-level synchronization of music (see Figure 6.5): when the old excerpt reaches the end of its length (i.e. at the end of the crossfade, that also corresponds to the first beat of the next bar), the new one reaches the first beat of its corresponding bar[19]. These two beats are then played together. This aspect greatly improved the musicality of the output with the music collection used during development, while not being particularly relevant for the arhythmic Phonos collection of music.
In order to prevent memory leaks, the old excerpt and its corresponding custom bin are both removed respectively from the playlist and from the global pipeline.

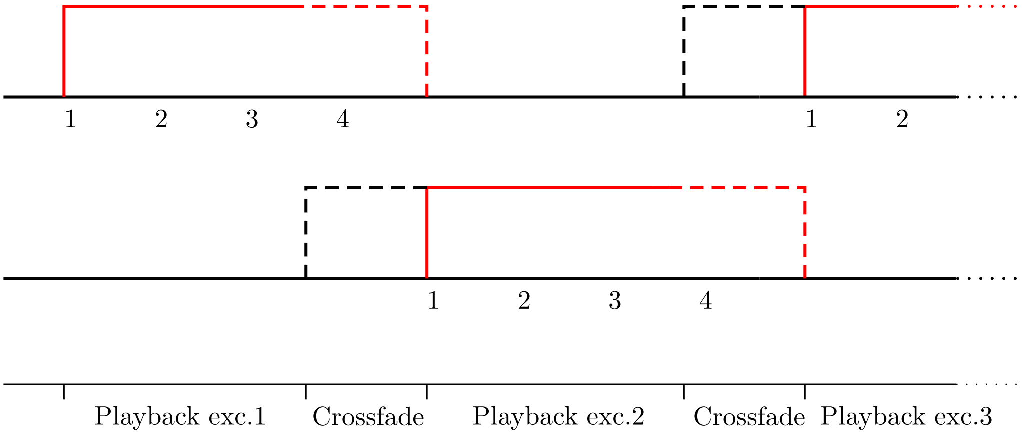


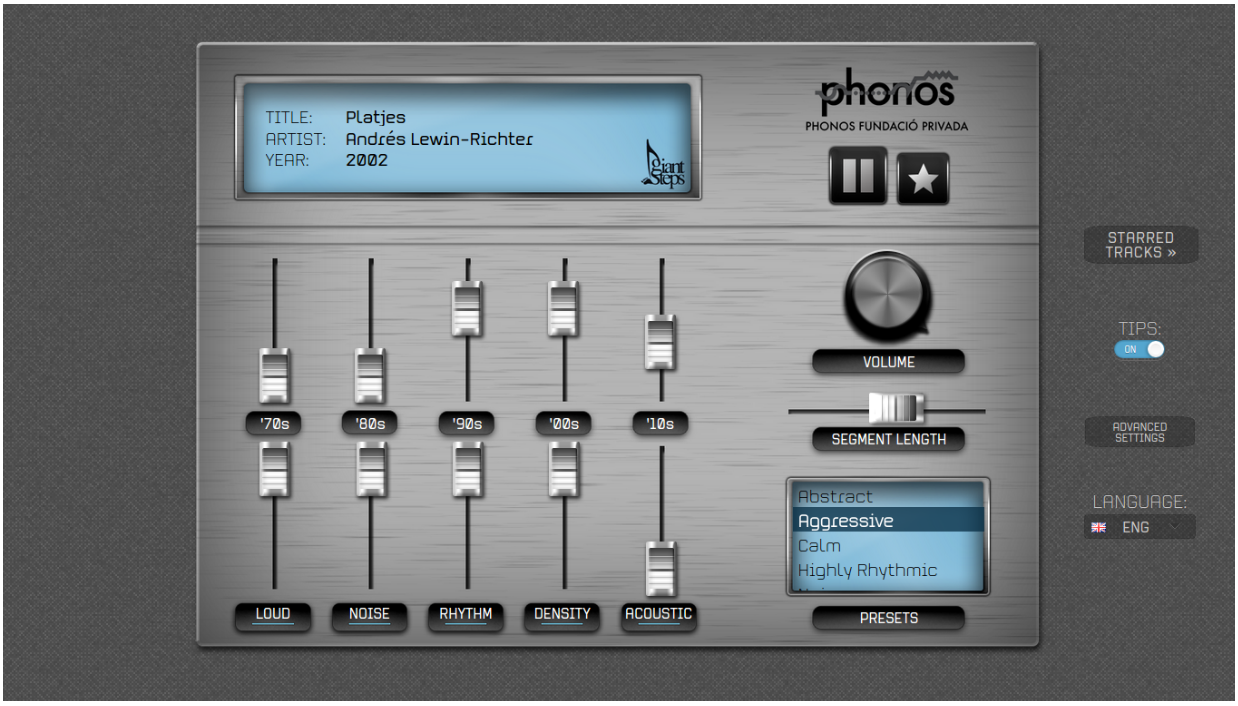
FIGURE 6.5: Handling of crossfades. The red rectangles indicated the content of the excerpt, and dashed lines indicate crossfades. Note that the playback involves more than just the excerpts' content: we use the portion of audio before it during the fade-in to achieve a beat-level synchronization. The indices indicate the number of the beats inside the excerpts.

The audio of the global pipeline is collected by the TCP sink, that is in charge of streaming this content over the TCP port 8070. This stream will be collected by the client application.

[18] By `start_point` we mean the starting point of the excerpt inside the track it belongs to.
[19] We recall that each excerpt corresponds to a bar.

6.2 The client application

The client application consists of a web-application hosted by the Flask application running on the server. To access it, the client needs to connect to the address `http://server_address:5000` on a browser. We entirely designed the graphical user interface of this application; this is shown in Figure 6.6.



TITLE: Platjes
ARTIST: Andrés Lewin-Richter
YEAR: 2002

Figure 6.6: Client application GUI.

This interface provides several ways for the user to control the music flow. Each time the user interacts with one of them, an HTTP post request is done from the client machine to the server, resulting in a change of the candidates for the playlist.

There are ten sliders: five of them are related to the year of release of the musical pieces, the other five are instead related to intrinsic characteristics of the music. In this way, the user has control both over the decade, and both over the type of music he wants to listen to. The motivation of this design choice is that we want to make the process of discovering music interactive while preserving ease of use. Furthermore, the suddivision of music into decades may be particularly useful in the use at the exhibition, since visitors could be particularly interested in hearing the differences between the works belonging to just a particular era over the entire 40 years life of Phonos.

The five sliders for music features are:

- Loudness

- Noisiness: related to the dissonance of the signal
- Rhythm: higher values of the slider lead to excerpts with high Onset Rates on high frequency content
- Density: higher values of the slider lead to excerpts where many Barkbands have a considerable amount of energy
- Acousticness: sets the ratio Acoustic/Electronic. Lower values of the slider mostly lead to electronic music.

The ranges of the sliders are dinamically generated during the computation of the fastmap: once the corresponding values for all the excerpts have been collected, these are sorted and we then pick the minimum, the maximum, and the first, second and third quartile for the values related to each descriptors. Therefore keeping the slider of the loudness at maximum will for instance lead to all the excerpts whose loudness value is between the third quartile and the maximum value of loudness of all excerpts. Step values for these five descriptors are calculated after the computation of the fastmap and kept in a separate JSON file.

The GUI additionally provides a set of presets for the values of these five sliders, a monitor for displaying information about the currently playing track, a slider for selecting the length of the audible segments (from 1 to 5 bars), and a knob for the volume (which controls the volume element of the global pipeline explained in Table 6.5).

By clicking on the button with a star on it, the user has the possibility of marking a track as favorite. The list of *"starred tracks"* is accessible on the second page of the GUI (shown in Figure 6.7), together with the list of the five last played track. The motivation behind this choice is to give the user the chance to keep track of the songs he has been finding interesting. At the exhibition, visitors may be particularly interesting in looking for more information about a track they like.

Furthermore, this interface is offered in three different languages: English, Spanish and Catalan. This has been done to increase the usability of the software at the exhibition, taking into account possible cultural differences.

The interface fully supports touch screen environments and is based on HTML5, CSS3 and Javascript. Many features of the jQuery library for Javascript are also used. The range sliders are based on *noUiSlider*[20], while the volume knob is based on *jQuery Knob*[21]. The design of the graphical user interface has been directed toward the ease of use, for the visitors of the Museum may not be particularly confortable with the use of software or of tools related to music manipulation or playback.

Corcerning the reception of the audio streaming, many efforts have been done in order to

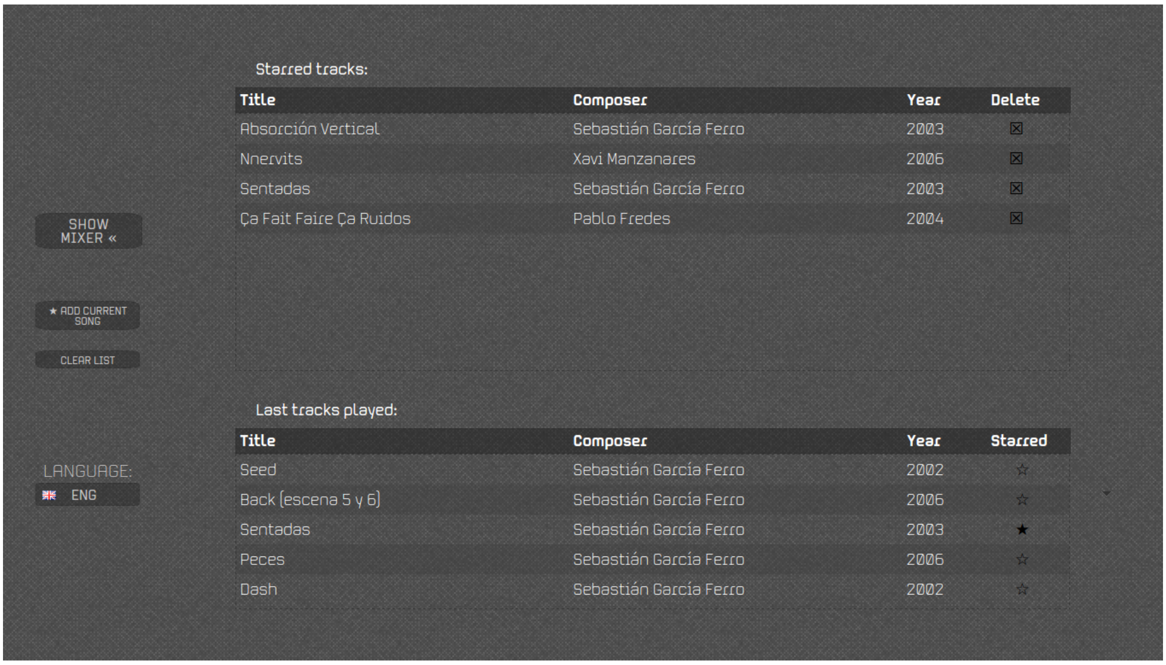
[20] `http://refreshless.com/nouislider/`

[21] `http://anthonyterrien.com/knob/`

achieve low-latency in the transmission of the multimedia content. Specifically, tries have involved the use of an icecast[22] server or specific GStreamer units to try to implement low-latency audio streaming directly accessible from the html5 page. None of these tries have fully worked: latency was always registered around 5 seconds, probably due to browser's buffering techiques. This performance was clearly unacceptable. Thus we decided to exploit the functionalities provided by VideoLAN VLC[23]: specifically, we wrote a daemon for the interactive kiosk that launches an hidden istance of VLC as soon as it detects a stream on the TCP port 8070 (generated by the TCPSink of GStreamer). This istance of VLC is then in charge of capturing and playing this stream of multimedia content. The main advantages of this choice are:

- Good latency (around 500ms)
- The user is completely unaware of this, for it is possible to start VLC in a daemon mode, thus without any sort of windows popping up.

The RTP/RTSP have not been taken into account, for the use of Flash in webpages is generally discouraged. Furthermore, we had no experience with it, and it could have not been feasible to implement it within the inauguration of the exhibition.



Starred tracks:

Title	Composer	Year	Delete
Absorción Vertical	Sebastián García Ferro	2003	☒
Nnervits	Xavi Manzanares	2006	☒
Sentadas	Sebastián García Ferro	2003	☒
Ça Fait Faire Ça Ruidos	Pablo Fredes	2004	☒

Last tracks played:

Title	Composer	Year	Starred
Seed	Sebastián García Ferro	2002	☆
Back (escena 5 y 6)	Sebastián García Ferro	2006	☆
Sentadas	Sebastián García Ferro	2003	★
Peces	Sebastián García Ferro	2006	☆
Dash	Sebastián García Ferro	2002	☆

FIGURE 6.7: The second page of the client application GUI, providing information about favorite and last played tracks.

[22] http://icecast.org/

[23] http://www.videolan.org/vlc/index.html

Part III

Results and Discussion

Chapter 7

Results

7.1 Performance

Performance has been the main concern in the development of the system. As already seen in previous chapters, many efforts have been made in order to achieve a good responsiveness to user input in the real time application. We made the clear choice of preferring low times in the offline computation of descriptors (reported in Table 5.4) for this has helped us in achieving good response times in the real time application. The latter ones, in general, greatly vary with the use of the application. For instance, the user interaction with sliders has the effect of emptying the playlist queue (which will result in temporary shorter computational times, due to the use of the least precise but fastest music similarity computation algorithm in order to get some new element into the playlist as soon as possible), while choosing to use longer segments or not interacting with the sliders may increase the computational time (for the system realizes that it has more time available for computing music similarity and then uses the most accurate algorithm[1]).

For us, this instability of performance is not intended as a flaw: it could rather be seen as good flexibility of the system to many different computational conditions.

We decided to collect data about computational times of the real time application for the choice of 1000 consecutive excerpts, with occasional interaction of the user. This is a reasonable analysis case, for it may be very similar to the real use of the system and also provides a good perspective on the computational times while using the most demanding algorithm of the system for computing music similarity. The results are shown for each main point of the procedure explained in Section 6.1.1.

[1] We recall that the only difference between the two algorithms lies in the choice of the similarity function, as shown at the point 8 of Section 6.1.1

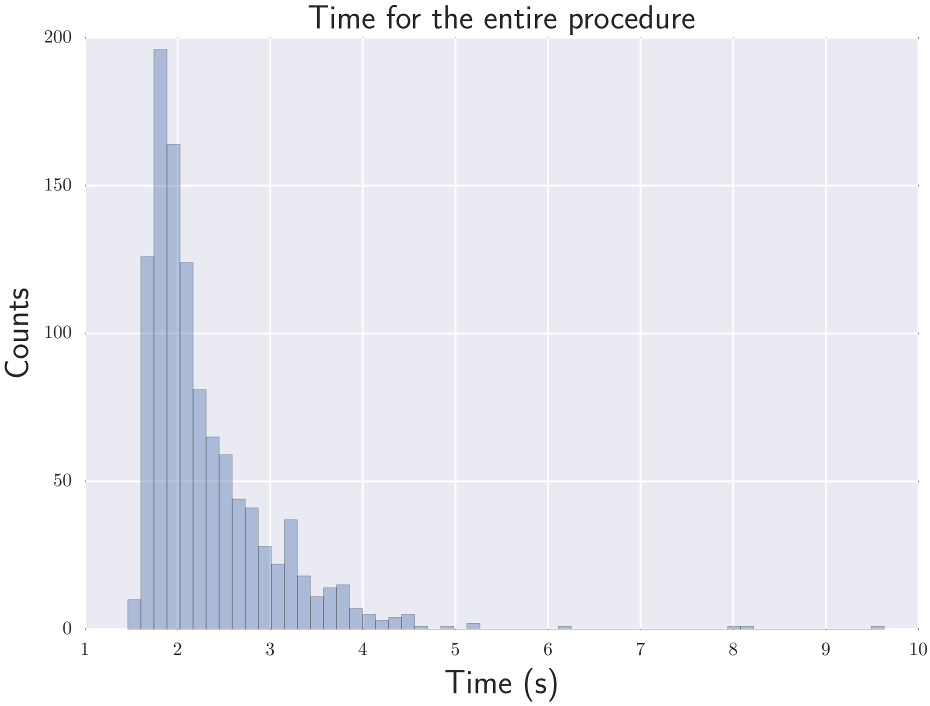


Figure 7.1: Global time for selecting next segment.

It can be seen that most of the times, the algorithm for choosing the next excerpt requires between 1.5 seconds and 3 seconds. The presence of some outliers above 5 seconds is due to particular conditions of the environment or of the operative system (such as some other process starting running in the background) and should not be considered meaningful for judging the performance of the algorithm itself.
We consider particurly appreciable that the system is capable of adapting its responsiveness to the environment, while still getting good response times also with most intensive computations. As already stated, during this experiment user interaction were occasional, leading the system to use the most intensive variant of the algorithm 732 out of 1000 times.

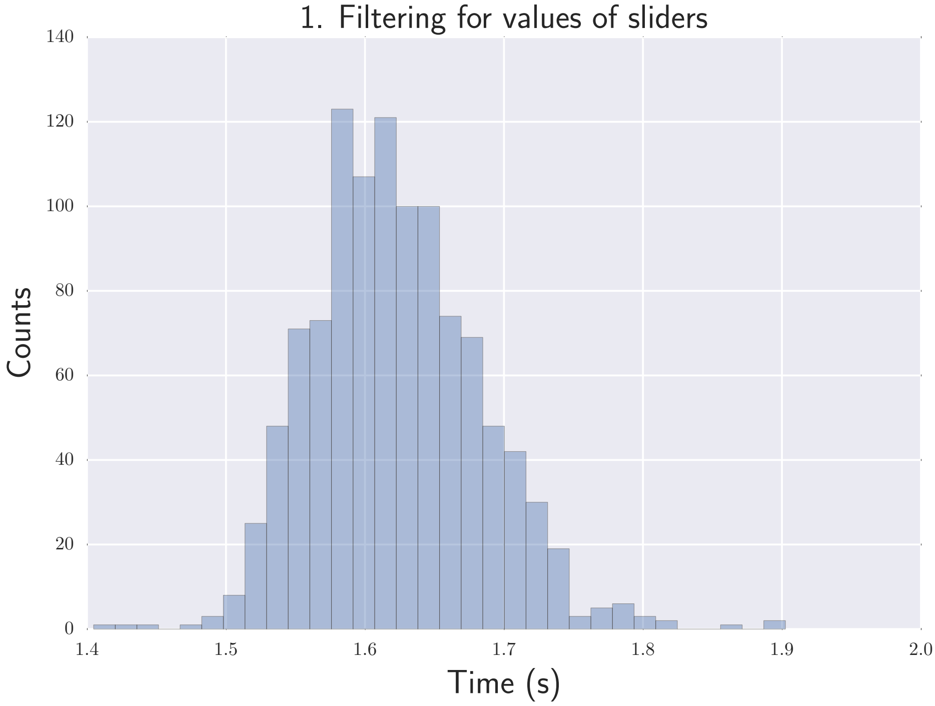


Figure 7.2: Time for performing the first step of the procedure: filtering of excerpts based on the current positions of sliders.

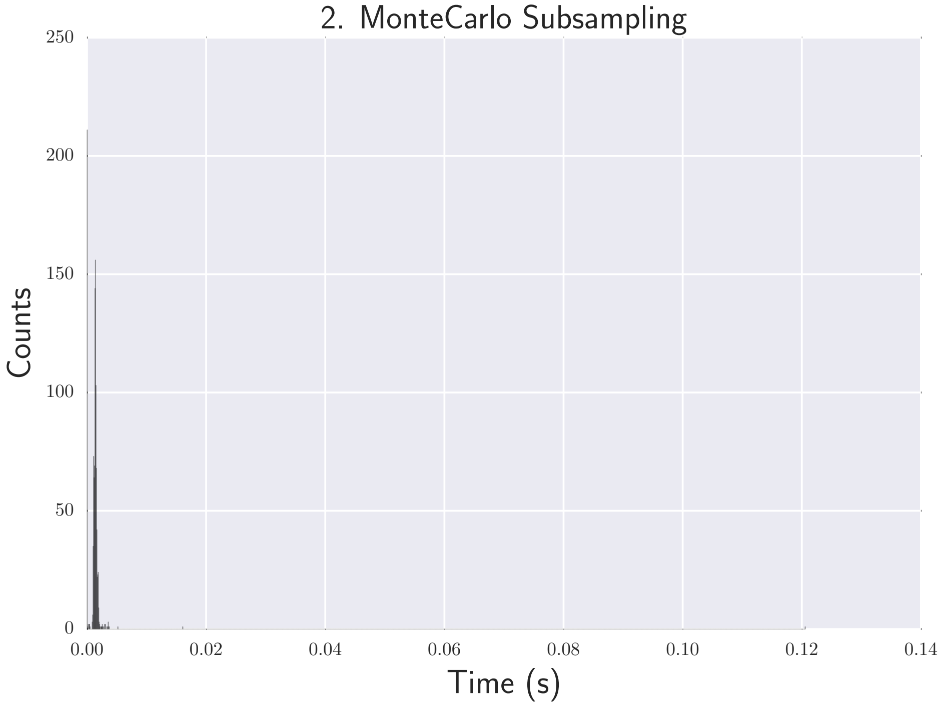


Figure 7.3: Time for performing Monte Carlo subsampling.

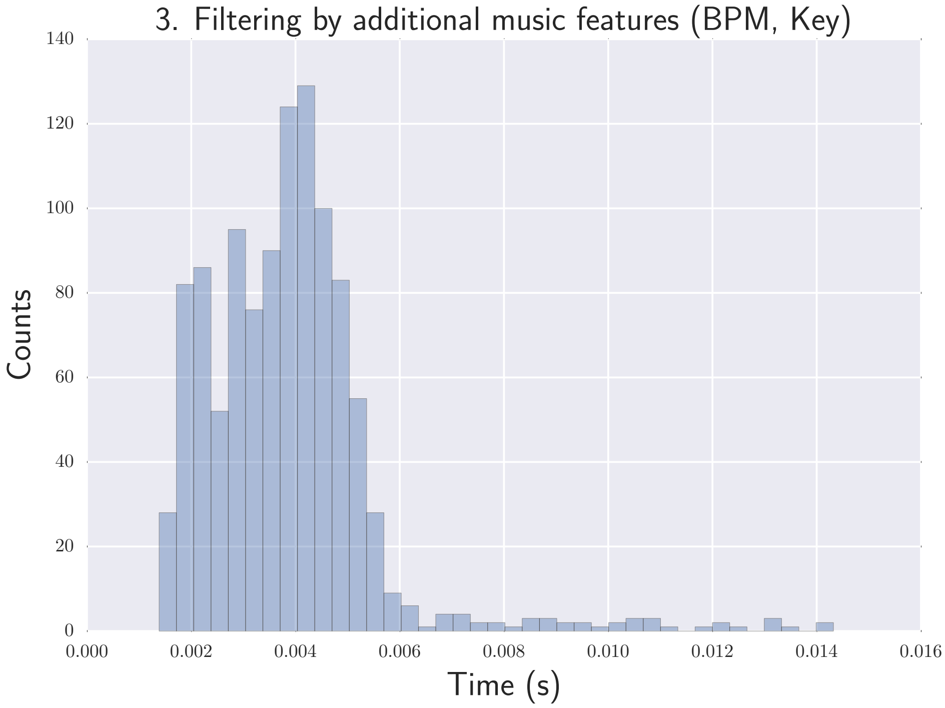


Figure 7.4: Time for filtering music according to musicality with current excerpt (in regards of BPM and key).

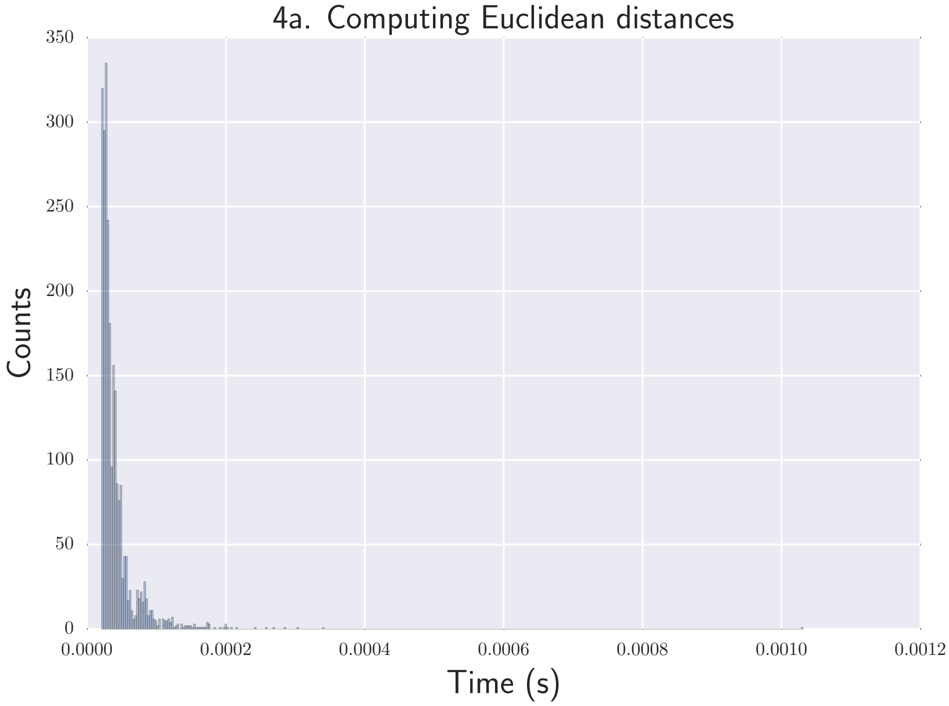


Figure 7.5: Time for computing euclidean distance between two 20D points.

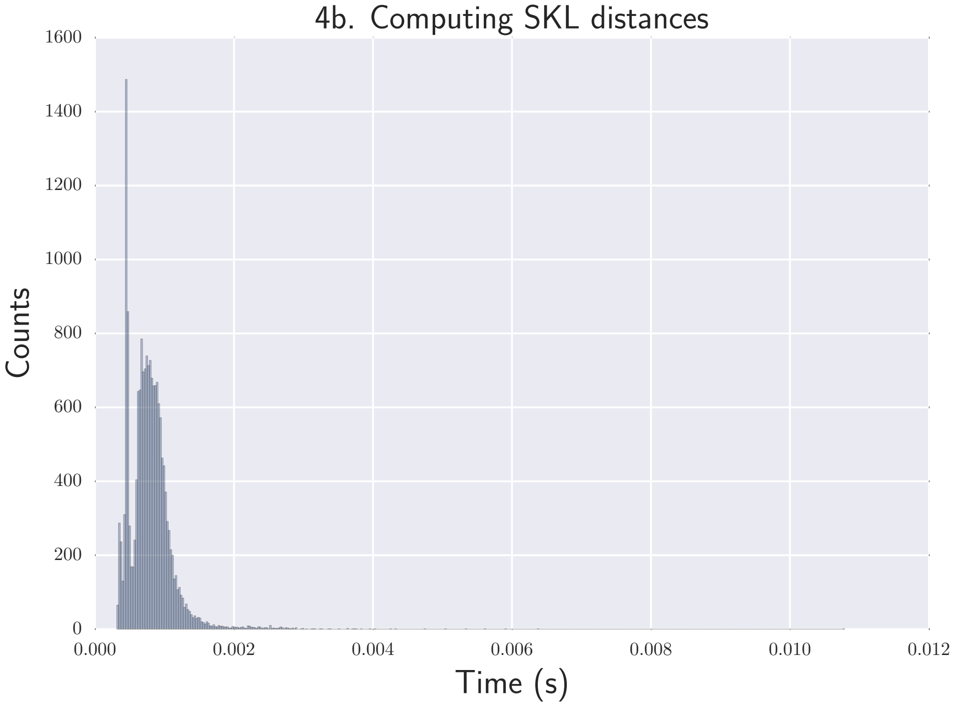


FIGURE 7.6: Time for computing symmetric Kullback-Leibler distance between two excerpts.

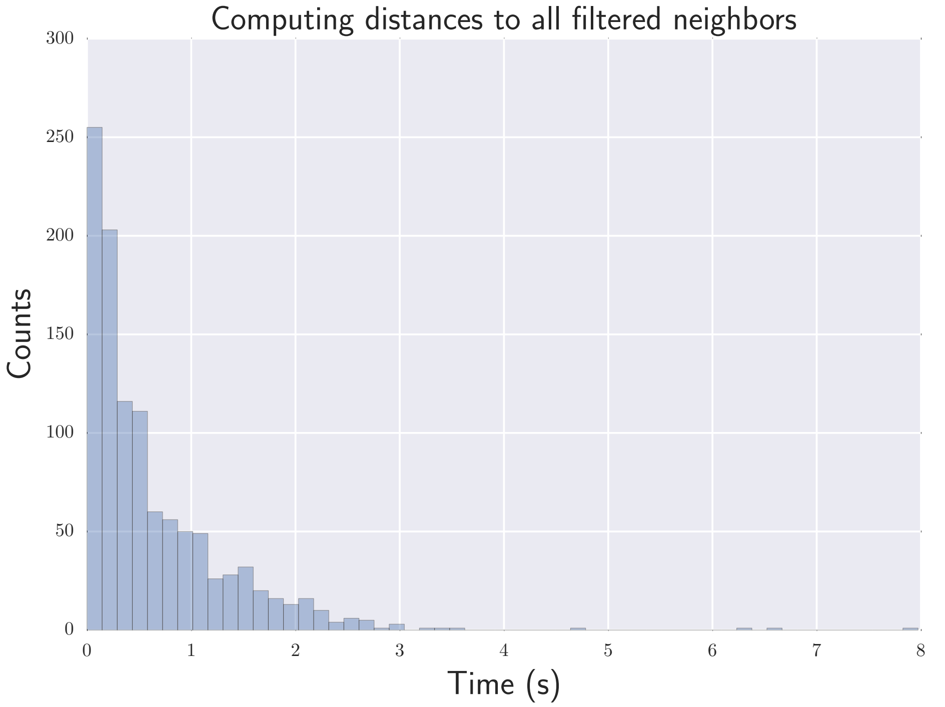


FIGURE 7.7: Time for computing distances from all filtered segments.

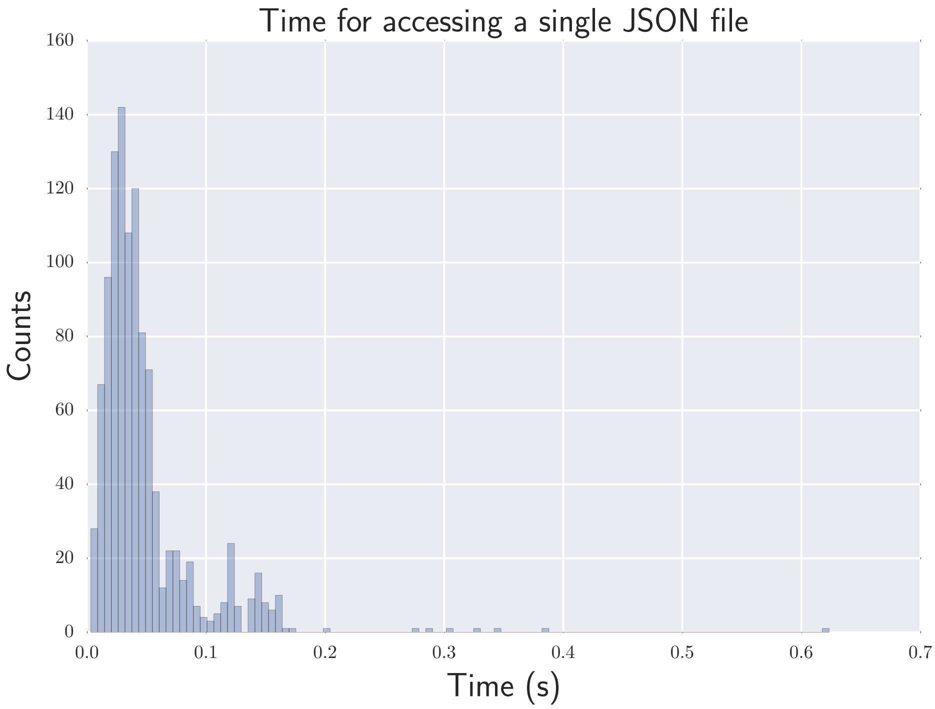


Figure 7.8: Time for accessing and parsing a JSON file.

Looking at these graphs, several details emerge:

- Without any doubt, the most demanding task is the filtering of candidates on the base of sliders value. This is due to the fact that this is step acting on the highest number of excerpts. This filtering is based on values that are stored on RAM and therefore is not sensibly slowed down by the time for accessing these values.
- Random subsampling, having possibly to act on a very large collection of excerpt, is one of the longest tasks.
- Once all the filtering steps are done, computing all the similarity distances generally requires around one second.
- Computing symmetric Kullback-Leibler distance is around 10 times slower then computing Euclidean distance.
- Time for accessing and parsing JSON file is not negligible and is actually 100 times longer than computing the symmetric Kullback-Leibler distance.

7.2 Evaluation

As explained in Section 4.4, we have decided to perform the evaluation of the system with specific experiments, followed by the compilation of a survey. Specifically, the experiments are organized as follows:

- The subject of the experiment is introduced to the purpose of the application, without explaining any details about the interaction or the functioning;
- The subject is given 5 minutes to freely interact with the application (playing with the Phonos collection of music), asking for clarification about the use if necessary;
- The subject is finally given the chance to ask about more the functioning of the system;
- The subject compiles a survey with specific questions about ease of useness, enjoyment of musical output, familiarity with the music and with this kind of software, and any problems.

NUM subjects took part to this evaluation, with the following global results:

7.3 Use at exhibition

7.4 Results obtained by the study

Chapter 8

Future Work

Appendix A

List of Essentia descriptors

As of November 2014, the features provided by Essentia 2.0.1 are:

Category	**Subcategory**	**Name**
Low-level	Barkbands	Values
		Kurtosis
		Skewness
		Spread
	Pitch	Value
		Instantaneous confidence
		Salience
	Spectral	Centroid
		Complexity
		Crest
		Decrease
		Energy
		Energyband high
		Energyband low
		Energyband middle high
		Energyband middle low
		Flatness db
		Flux

		Kurtosis
		Rms
		Rolloff
		Skewness
		Spread
		Strongpeak
	Other	Average loudness
		Dissonance
		Hfc
		Mfcc
		Sccoeffs
		Scvalleys
		Silence rate 30dB
		Silence rate 30dB
		Silence rate 60dB
		Zerocrossingrate
Rhythm	Beats	Position
		Loudness
		Loudness band ratio
	BPM	Value
		Estimates
		Intervals
	First peak	BPM
		Spread
		Weight
	Onset	Onset Rate
		Onset Times
	Second peak	BPM
		Spread

		Weight
Sfx	Pitch	After max to before max energy ratio
		Centroid
		Max to total
		Min to total
	Other	Inharmonicity
		Oddtoeven harmonic energy ratio
		Tristimulus
Tonal	Chords	Changes rate
		Histogram
		Key
		Number rate
		Progression
		Scale
		Strength
		HPCP
	Key	Value
		Scale
		Strength
		Thpcp
	Tuning	Diatonic strength
		Equal tempered deviation
		Frequency
		Nontempered energy ratio

Table A.1: List of features computable with Essentia.

Appendix B

List of Echo Nest Features

Category	Subcategory	Name
Meta		Timestamp
		Duration seconds
		Audio MD5
		Analysis time
		Num samples
		Album
		Decoder version
		Sample rate
		Title
		Duration
		Sample md5
		Decoder
		Artist
		Id
		Window seconds
		Genre
		Analysis sample rate
		Analyzer version
		Bitrate

		Md5
		Analysis channels
Structure	Segments	Start
		Duration
		Confidence
		Loudness start
		Loudness max
		Loudness max time
		Pitch 01
		Pitch 02
		Pitch 03
		Pitch 04
		Pitch 05
		Pitch 06
		Pitch 07
		Pitch 08
		Pitch 09
		Pitch 10
		Pitch 11
		Pitch 12
		Timbre 01
		Timbre 02
		Timbre 03
		Timbre 04
		Timbre 05
		Timbre 06
		Timbre 07
		Timbre 08
		Timbre 09
		Timbre 10

		Timbre 11
		Timbre 12
	Sections	Start
		Duration
		Confidence
		Mode
		Mode confidence
		Key
		Key confidence
		Tempo
		Tempo confidence
		Time signature
		Time signature confidence
		Loudness
Rhythm	Bars	Start
		Confidence
		Duration
	Beats	Start
		Confidence
		Duration
	Tatums	Start
		Confidence
		Duration
Desc		Danceability
		Speechiness
		End of fade in
		Start of fade out
		Liveness
		Acousticness
		Valence

	Energy
	Loudness
	Tempo
	Tempo confidence
	Time signature
	Time signature confidence
	Mode
	Mode confidence
	Key
	Key confidence

Table B.1: List of audio features provided by Echo Nest.

Appendix C

Phonos: list of songs

The musical pieces to be used during the *"Phonos, 40 anys de música electrònica a Barcelona"* exhibition at Museu de la Musica (L'Auditori, Carrer de Lepant, 150, 08013 Barcelona) are:

Artist	**Title**	**Year**
Alain Perón	De Dos Para Uno	1996
	Los Edictos	1998
Albert Llanas	Nexus	1999
	Formants	2004
Alejandro Martínez	Monoleg	N.A.
	Helesponto	1982
	Tazir	1984
	Crisálida	1987
	Machina animata	1987
	Canción de Otoño	1989
	Homenaje L.Nono	1990
	Música Palimpesto	1991
	Vaciando el hueco	1996
Alex Arteaga	Témenos	2006
Alex Geell	Panales	2010
Alex Sanjurjo	Fluir	2003

Alexandra Gardner	Ayehli	2002
	Snapdragon	2002
	New Skin	2003
	Onice	2003
	Luminoso	2004
	Tourmaline	2004
Alexandre Marino	Apparatus, Experimentalis	2008
	Apparatus, Musical	2008
Andrés Lewin-Richter	Joc - Eventos	1976
	Joc - Fondo	1976
	Acción 2 - 1	1978
	Acción 2 - 2	1978
	Acción 2 - 3	1978
	Acción 2 - 4	1978
	Giravolt	1978
	El Paraiso	1979
	El Viento I - 1	1979
	El Viento I - 2	1979
	El Viento I - 3	1979
	El Viento II	1979
	Reacciones I II	1979
	Secuencia IV	1979
	Baschetiada	1980
	El Viento III	1980
	El Viento IV	1980
	Reacciones III	1980
	Wagler Walricci	1981
	Actualidad discográfica	1982
	Sones	1982

6 Songs	1983
Quorum	1983
Secuencia V	1983
Secuencia VI	1983
Tinell	1983
Cogida	1984
In memoriam Manuel Valls	1984
Isaac el Cec	1984
Juegos	1985
Musica electroacústica	1985
Solars Vortices	1985
Desfigurat	1986
Diálogos	1987
Secuencia VII	1987
Homenaje a Zinovieff	1988
Secuencia VIII	1988
Verra la Morte	1988
Verra la Morte 1	1988
Verra la Morte 2	1988
Verra la Morte 3	1988
Verra la Morte 4	1988
Verra la Morte 5	1988
Verra la Morte 6	1988
Verra la Morte 7	1988
Verra la Morte 8	1988
99 Golpes	1989
Ben avra questa donna cor di ghiacio	1989
Secuencia IX	1989
Strings	1989
Brossiana	1990

Donne Fiori	1990
Fragmento (a Nono)	1990
Frullato	1990
Ludus Basiliensis	1991
Reacciones IV	1991
Secuencia X	1991
Radio 2	1996
Sarangi	1999
Configuraciones	2000
Constelaciones	2000
Figuras	2000
Resonancias	2000
Secuencia XI	2001
Secuencia XII	2001
Dreams	2002
Ludus Allavarium	2002
Platjes	2002
Secuencia XIII	2002
Signals	2002
Viso di Primavera	2002
Fantasia	2003
Juego de Acordeón	2003
Meisoh No Ne	2003
Melodias	2003
Metálica	2003
Omaggio a Berio: sequenza per tuba	2003
Secuencia XIV	2003
Essay on Trombone	2004
Fragments	2004

	Secuencia XV	2004
	Arssonxx.rne	2005
	Fluxus es zen?	2005
	Interacciones	2006
	On ”Freesound” Water	2006
	Secuencia XVI	2006
	For Harry	2007
	Retales	2007
	Sombras	2007
	Soplos	2007
	Sospiri	2007
	Friendship Quartet	2008
	Homenaje a Pierre Schaeffer	2008
	Makeup	2008
	Schaeffer granulado	2008
	Aire	2009
	Génesis	2009
	Homenaje a Varese	2009
	Memento	2009
	Paseo BCN	2009
	Sancta Maria	2009
	Slapring	2009
	Spring	2009
	Imagenes	2010
	Secuencia XVIII Fagot	2010
	Multifonia	2011
	Campanas para una celebracion	2012
	Multifonia III	2012
	Secuencia XIX	2014
Espai Sonor	Anna Bofill	

Trio para Violin y Cinta	Anna Bofill	
Sinapsis	Ariadna Alsina	2006
Reconstrucció	Ariadna Alsina	2011
Vels Vitris	Ariadna Alsina	2012
Contramarea	Ariadna Alsina & David Salleras	2009
La Música Que Había en Mis Objetos	Arturo Moya	1996
Estampas de Caza 1	Arturo Moya	2000
Estampas de Caza 2	Arturo Moya	2000
Estampas de Caza 4	Arturo Moya	2000
Estampas de Caza 5	Arturo Moya	2000
Estate quieto Voltaire	Arturo Palaudarias	
Adolescencia y Estrella	Arturo Palaudarias	1980
Escudellers	Arturo Palaudarias	1981
Piamo	Arturo Palaudarias	1984
Toda la Memoria de un Hombre	Arturo Palaudarias	1987
El Destino de las Cosas	Arturo Palaudarias	1988
La Luz de los Sueños	Arturo Palaudarias	1989
Boule de Feu	Arturo Palaudarias	1990
Paréntesis militar	Arturo Palaudarias	1990
El Juicio Estético Universal	Arturo Palaudarias	1991
Moverse en el Tiempo	Arturo Palaudarias	1997
Women in Process	Aurelio Edler-Copes	2013
Exquisits	Cadavers	2003
Latido	Carlos Lupprián	1995
Agugagá	Carlos Lupprián	1996
Naturaleza Muerta	Carlos Lupprián	1997
Improvisación con Oratio Trio	Claudio Nervi	2010

Valent La Notte	Claudio Zulian	
El Libro de los Excesos	Claudio Zulian	1983
San Claudio Vive Solo	Claudio Zulian	1985
Sexo y Politica	Claudio Zulian	1987
I Quattro Continenti	Claudio Zulian	1989
Por de Ser Set	Claudio Zulian	1989
Sueños Ecléctricos	Claudio Zulian	1989
Variazione Angelica	Claudio Zulian	1990
2 Escenas de Macbeth - 1	Claudio Zulian	1991
2 Escenas de Macbeth - 2 Ruidos	Claudio Zulian	1991
Armonias 1	Concha Trallero	1980
Armonias 2	Concha Trallero	1980
Armonías Sonoras 1	Concha Trallero	1980
Armonías Sonoras 2	Concha Trallero	1980
Leftraru, Viajero Ensoñado - El Río de la Vida	Cristián López	2005
Leftraru, Viajero Ensoñado - Espírutu Azul	Cristián López	2005
Leftraru, Viajero Ensoñado - Interludio	Cristián López	2005
Leftraru, Viajero Ensoñado - Piedra Solitaria	Cristián López	2005
Leftraru, Viajero Ensoñado - Relámpago Azul	Cristián López	2005
Relief II	Cristián Morales-Ossio	2001
TRTPS	Daniel Domínguez Teruel	2010
SKTHN	Daniel Domínguez Teruel	2012
Study I	Daniel Domínguez Teruel	2013
Study II	Daniel Domínguez Teruel	2013
Study V	Daniel Domínguez Teruel	2013

Say It	Daniel Rios Aranda	1987
Erial	Daniel Rios Aranda	1990
Sueños	Danilo Vidotti	2008
Psicofonias Urbanas 1	Danio Catanuto	2010
Psicofonias Urbanas 2	Danio Catanuto	2010
Formantes	Darío Cortés	1998
Pulsajes	David Dalmazzo	2010
Confluencies	David Padros	1985
Caosmofonia	Diego Dall'Osto	1998
Kinoko Tabí	Doénado, el Ur	1988
Pedicoj en la Arena del Pamir	Doénado, el Ur	1989
Zalody	Doénado, el Ur	1990
Yñé do zalod	Doénado, el Ur	1991
A Sensu Contrario	Doénado, el Ur	1992
Blordt Prelar	Doénado, el Ur	1992
Kzadzak	Doénado, el Ur	1994
Tu Mateix	Edgar Barroso	2004
Dux	Edgar Barroso	2005
Tau	Edgar Barroso	2005
Tu Soplo Que Transporta	Edgar Barroso	2005
IOD	Edgar Barroso	2006
CYT	Edgar Barroso	2007
Mármore	Edson Zampronha	2001
Mármore 1	Edson Zampronha	2001
Mármore 2	Edson Zampronha	2001
Mármore 3	Edson Zampronha	2001
Read my LISP	Eduard Resina	1991
L'Esquizofrènia Dels Sons	Eduard Resina	1993
Aca Amaron	Eduard Resina	2001
L'Anna-crusa	Eduard Resina	2002

Espai Sonor	Eduardo Polonio	1976
Requiem per una Veu Perduda	Eduardo Reck Miranda	1997
Midi de Sable	Elsa Justel	2000
Elementos Constantes, Hechos Variables	Enrique Marín	2002
Transiciones de Fase	Enrique Marín	2007
Untitled 1	Ensamble Crumble y ReacTable	2006
Untitled 2	Ensamble Crumble y ReacTable	2006
Untitled 3	Ensamble Crumble y ReacTable	2006
Untitled 4	Ensamble Crumble y ReacTable	2006
Untitled 5	Ensamble Crumble y ReacTable	2006
Untitled 6	Ensamble Crumble y ReacTable	2006
Untitled 7	Ensamble Crumble y ReacTable	2006
Untitled 1	FMOL Trio	2001
Untitled 2	FMOL Trio	2001
Untitled 3	FMOL Trio	2001
CampoSanto	Felipe Pérez Santiago	2004
Encandilado	Felipe Pérez Santiago	2007
Hunger FM	Felipe Pérez Santiago	2009
Hurt	Felipe Pérez Santiago	2009
Ishmael	Felipe Pérez Santiago	2009
Miuk	Felipe Pérez Santiago	2009
Post War	Felipe Pérez Santiago	2009
Tacto	Felipe Pérez Santiago	2009
War-Post War	Felipe Pérez Santiago	2009
Pronto Desapareceremos	Felipe Pérez Santiago	2012
Ecos 1	Fernando Jobke	2008
Ecos 2	Fernando Jobke	2008
Ecos 3	Fernando Jobke	2008

Ecos 4	Fernando Jobke	2008
Ecos 5	Fernando Jobke	2008
Ecos 6	Fernando Jobke	2008
Cuerpos Sensibles	Félix Luque & Ricardo Gadea	2005
The Machine Manifesto	Félix Luque & Thomas Charveriat	2004
Batucada Amenazante	Gabriel Brncic	1970
El Túnel (a Ernesto Sabato)	Gabriel Brncic	1970
Agua 1	Gabriel Brncic	1971
Agua 2	Gabriel Brncic	1971
Agua 3	Gabriel Brncic	1971
Cielo	Gabriel Brncic	1980
Destierro	Gabriel Brncic	1980
Chile Fértil Provincia	Gabriel Brncic	1983
Concert Gothique	Gabriel Brncic	1985
Operas Rotas	Gabriel Brncic	1985
Clarinen Tres	Gabriel Brncic	1986
Clarinen Tres	Gabriel Brncic	1986
Desêtre a Oscar Masotta	Gabriel Brncic	1986
Triunfo Para las Madres	Gabriel Brncic	1986
Aria y Pasacalle	Gabriel Brncic	1987
Ese Mar	Gabriel Brncic	1987
Música de cámara	Gabriel Brncic	1987
Historia de Dos Ciudades	Gabriel Brncic	1988
Alegrias	Gabriel Brncic	1989
Composición de 1989 a Eduardo Polonio	Gabriel Brncic	1989
Dulcian Concert	Gabriel Brncic	1989
ariaciones sobre Sonatas e Interludios	Gabriel Brncic	1989
Adagio-Scherzo	Gabriel Brncic	1990

Vade Retro a Luigi Nono	Gabriel Brncic	1990
Dos Esbozos Para Antiguos Instrumentos Electrónicos	Gabriel Brncic	1994
...Que No Desorganitza Cap Murmuri	Gabriel Brncic	1995
Constanza	Gabriel Brncic	1996
Claro-Oscuro	Gabriel Brncic	1998
Meng	Gabriel Brncic	1998
Clarinet Concert	Gabriel Brncic	1999
Coreutica	Gabriel Brncic	1999
Ergon-Rondeau	Gabriel Brncic	2000
A Joan Miró	Gabriel Brncic	2001
Alto-Concert II	Gabriel Brncic	2001
Bass clarinet-Concert for Harry Sparnaay	Gabriel Brncic	2003
Son(ru)idos I	Gabriel Brncic	2003
Son(ru)idos II	Gabriel Brncic	2003
La Casa del Viento 1	Gabriel Brncic	2006
La Casa del Viento 2	Gabriel Brncic	2006
Pregoneros de Barcelona	Gaspar Lukacs Esguep	2002
Sin título	Germán Brull Moreno	2004
Sin título	Germán Brull Moreno	2004
Arboleda	Graciela Muñoz Farida	2011
Fragmentos de un Arbol	Graciela Muñoz Farida	2011
Lo Que No Das Te Lo Quitas	Graciela Muñoz Farida	2011
Viento Sur	Graciela Muñoz Farida	2011
Piece for Guitar and Tape	Graeme Truslove	2001
Improvisation	Graham Coleman	2007
Improvisation	Graham Coleman	2007
Guitarrísticamente	Guillermo Eisner	2007

Duo Para Siete	Igor Bimsbergen	1996
Luis y Marylin	Igor Bimsbergen	1998
Free What	Ismael Sanoja & Kai Kraatz	2006
Free What 1	Ismael Sanoja & Kai Kraatz	2006
Free What 2	Ismael Sanoja & Kai Kraatz	2006
Free What 3	Ismael Sanoja & Kai Kraatz	2006
Free What 4	Ismael Sanoja & Kai Kraatz	2006
Free What 5	Ismael Sanoja & Kai Kraatz	2006
Free What 6	Ismael Sanoja & Kai Kraatz	2006
Free What 7	Ismael Sanoja & Kai Kraatz	2006
Free What 8	Ismael Sanoja & Kai Kraatz	2006
Traumtäntze	Jan Schacher	2000
Preludios	Javier Navarrete	1976
Almogavers	Jelena Vico	2008
Brithm	Jelena Vico	2008
Mrzbw	Jelena Vico	2008
Pangea	Jelena Vico	2008
Zeno	Jelena Vico	2008
Zitar	Jelena Vico	2008
Gallinària	Jep Nuix	1980
Doble Peça de Lletres i Sons	Jep Nuix	1981
Tres Canons de Noces	Jep Nuix	1981
Ad Valorem	Jep Nuix	1984
Halterofilia 1	Jep Nuix	1984
Serenata Nocturna	Jep Nuix	1985
L'Inizio	Jep Nuix	1986
Dit a Dit, Pas a Pas	Jep Nuix	1988
Asirara	Jep Nuix	1989
Monoleg	Jep Nuix	1989
Trialeg	Jep Nuix	1989

His Master's Voice	Jep Nuix	1990
Improvisació per a tubs	Jep Nuix	1990
Pensant en Nono	Jep Nuix	1990
Percuflu	Jep Nuix	1990
Atentament	Jep Nuix	1992
Stack	Jep Nuix	1995
Intersections-BouleWav 2.0	Joan Bagés i Rubí	2006
Al Tranquilodromo	Joan Josep Ordinas & Claudio Zulian	1981
Passadis	Joan Sanmarti	1200
Reflexos Improvisacxiones Asistidas por Ordenador	Joan Sanmarti	1997
Xtrapolució 4	Joan Sanmartí	1998
Ricercare a 5	Jordi Rossinyol	1986
Objectes Trobats a la Platja	Jordi Rossinyol	1987
Ocellots	Jordi Rossinyol	1988
Mòbils Inquiets i Altres Equivocs	Jordi Rossinyol	1989
Prosper Laberint Intermitent	Jordi Rossinyol	1990
Variaciones guit	Jordi Rossinyol	1990
Concert Mestis	Jordi Rossinyol	1997
Ecliptic	Jordi Rossinyol	2004
El Doble Bandoneón	Jorge Sad	1998
La Ida Hacia Abajo de la Tierra de la Tarde	Jorge Sad	1999
Landscape	Josep Maria Guix	2010
Landscape	Josep Maria Guix	2010
Landscape	Josep Maria Guix	2010
Oxo	Josep Maria Mestres Quadreny	1963
Peça per a Serra Mecanica	Josep Maria Mestres Quadreny	1963
Homenaje a Galileo	Josep Maria Mestres Quadreny	1965

Trois Cánones en Hommage à Galilea	Josep Maria Mestres Quadreny	1968
Aronada	Josep Maria Mestres Quadreny	1972
El Teler de Teresa Codina	Josep Maria Mestres Quadreny	1973
Song for Jane Manning	Josep Maria Mestres Quadreny	1973
Espai Sonor	Josep Maria Mestres Quadreny	1976
Espai Sonor	Josep Maria Mestres Quadreny	1976
Quina	Josep Maria Mestres Quadreny	1979
Cánones a Galileo	Josep Maria Mestres Quadreny	1989
El Pensamiento Que Se Trabaja Hacia la Luz	José Manuel Berenguer	
Spira	José Manuel Berenguer	
Montardo	José Manuel Berenguer	1983
A Florats	José Manuel Berenguer	1984
La Logica de la Sorpresa	José Manuel Berenguer	1984
El Ponent Excesiu	José Manuel Berenguer	1985
La Perla Estranya	José Manuel Berenguer	1985
La Relojeria del Tío Paco	José Manuel Berenguer	1985
Música en la Noche	José Manuel Berenguer	1985
Quartet Ambar	José Manuel Berenguer	1986
Color	José Manuel Berenguer	1987
Polifonía de Colores	Juan Antonio Moreno	1984
Preludio III a Lluis Callejo	Juan Antonio Moreno	1988
Nono Está Aqui	Juan Antonio Moreno	1990
Buenhache	Juan Antonio Moreno	1991
G-Gems	Lina Bautista	
Bombyx Mori	Lina Bautista	2010
Encélado	Lina Bautista	2011
A River From the Walls	Linda Antas	1999

Sueño sin palabras	Linda Antas	2001
Untitled	Lisos-Estriados	2001
Carota i Caramel	Llorenç Balsach	1976
Espais residuals (Espai I)	Llorenç Balsach	1976
L'assassi Bagliatti	Llorenç Balsach	1977
El Cant de les Arteries	Llorenç Balsach	1979
Caleidoscopi	Lluis Callejo	
Dibuixos	Lluis Callejo	1981
Estructures 6502	Lluis Callejo	1982
Paisatges	Lluis Callejo	1983
Tèxtils	Lluis Callejo	1984
A Pitàgores en do	Lluis Callejo	1985
A Pitàgores en re	Lluis Callejo	1985
Espai Sonor	Lluis Callejo	2003
Stokos IV	Lluis Callejo	2003
La Triste Herida de Margot	Luis Caruana	2001
Por Tus Pliegues Transita la Pena	Luis Caruana	2001
Animales Divinos	Marcelo DeMatei & Carlos Smith	2003
Petit Estudi	Mario Peña y Lillo	
Beso	Mario Peña y Lillo	2013
El Contorno de sus Ojos	Mario Peña y Lillo	2013
Esencia	Mario Peña y Lillo	2013
He Perdido la Apuesta	Mario Peña y Lillo	2013
Youkali	Mario Peña y Lillo	2013
Figuras Negras	Mario Verandi	1992
Flamencas	Mario Verandi	1995
Faces and Intensities	Mario Verandi	1996
Frèquences de Barcelone	Mario Verandi	1997
Mu	Mario Verandi	1997

Mists	Matthew Burtner	1996
Fern	Matthew Burtner	1997
Incantation S4	Matthew Burtner	1997
Split Voices	Matthew Burtner	1997
Glass Phase	Matthew Burtner	1998
Portals of Distortion	Matthew Burtner	1998
Delta 1	Matthew Burtner	2000
Duo	Mauricio Valdés	2002
Popan II	Mauricio Valdés	2008
Gramatges	Mercè Capdevila	1983
Baobab	Mercè Capdevila	1985
Nu	Mercè Capdevila	1990
Alegries de Comèdia	Mercè Capdevila	1991
Mercuri	Mercè Capdevila	1991
Fons de Mar	Mercè Capdevila	2000
Pols	Mercè Capdevila	2000
Puente	Mercè Capdevila	2000
A Chillida	Mercè Capdevila	2009
Time Machine	Miquel Jordà	2000
La Máquina, el Humano y el Olivo	Nadine Kroher	2013
Mixed Signals	Nadine Kroher	2014
Concierto Sonocromático	Neil Harbisson	2011
Catarsis III	Oliver Rappoport	2009
Laberint Mutant II	Oriol Graus	1987
Miradaclosa IV	Oriol Graus	1987
I despres...	Oriol Graus	1990
La Solitud de l'Origen	Oriol Graus	1990
La conseqüència	Oriol Graus	1990

La intuïció	Oriol Graus	1990
Oketus	Oriol Graus	1990
Diferents Formes de Dir - T'Ho	Oriol Graus	1991
La Tolerancia	Oriol Graus	1993
El Laberint de l'Esperança	Oriol Graus	2000
Paisatge Interior	Oriol Graus	2010
Black Nature	Oscar Martin	2012
Black Nature	Oscar Martin	2012
Fer et Defer	Pablo Fredes	
Historia del Vinilo	Pablo Fredes	
Trama	Pablo Fredes	
Las Nenias del Sonido	Pablo Fredes	2002
Ça Fait Faire Ça Ruidos	Pablo Fredes	2004
El Círculo de Cero	Pablo Fredes	2009
sX-off-on	Pablo Fredes	2009
Azu Gemma Torralbo	Pablo Fredes	2011
Son-ethos (Sueños en el Sueño)	Pablo Fredes	2011
Son-file	Pablo Fredes	2011
iO	Pablo Fredes	2011
on_off Gemma Torralbo	Pablo Fredes	2011
Cero Roce Sostenuto	Pablo Fredes	2012
Estratos	Pedro Barboza	2001
Estratos	Pedro Barboza	2001
La fila de Ocata	Pedro Barboza	2001
inTENSIONtres	Pedro Barboza	2004
Mantra I	Ramon Humet	2005
1	Rebecka Biro	2005
2	Rebecka Biro	2005
Daffodil for Peter Billings	Ricardo Arias	
Improvisación	Ricardo Arias & Carlos Gómez	2009

Sol Sonoro 1	Ricardo Arias & Roberto García	2008
Sol Sonoro 2	Ricardo Arias & Roberto García	2008
Je Suis l'Autre	Roger Costa	2012
off ICMC2005	Ross Bencina	2005
off ICMC2005	Ross Bencina	2005
Simple Math	Sanjay Fernandes	2010
Ella Era Todo - Escribir Sobre Piel	Sebastián García Ferro	
Ella Era Todo - Yang	Sebastián García Ferro	
Europa 1 - Piano	Sebastián García Ferro	
Europa 2 - Crescendo	Sebastián García Ferro	
Europa 3 - Bosque	Sebastián García Ferro	
Europa 4 - Vibracion	Sebastián García Ferro	
Europa 5 - Noise Delay Long	Sebastián García Ferro	
Europa 6 - Piano	Sebastián García Ferro	
Equs	Sebastián García Ferro	2001
Noise	Sebastián García Ferro	2001
Pulso	Sebastián García Ferro	2001
Afro Dero	Sebastián García Ferro	2002
Ceratti	Sebastián García Ferro	2002
Dash	Sebastián García Ferro	2002
Seed	Sebastián García Ferro	2002
Shadow	Sebastián García Ferro	2002
Silla	Sebastián García Ferro	2002
Absorción Vertical	Sebastián García Ferro	2003
Bosa	Sebastián García Ferro	2003
Drugs	Sebastián García Ferro	2003
Etheric	Sebastián García Ferro	2003
Fiesta	Sebastián García Ferro	2003

Final	Sebastián García Ferro	2003
Huellas	Sebastián García Ferro	2003
Huellas Intro	Sebastián García Ferro	2003
Mistrius	Sebastián García Ferro	2003
Nervio	Sebastián García Ferro	2003
Rebotes	Sebastián García Ferro	2003
Rhesus	Sebastián García Ferro	2003
Sentadas	Sebastián García Ferro	2003
Solo Caro	Sebastián García Ferro	2003
Trio	Sebastián García Ferro	2003
Viaje Transparente	Sebastián García Ferro	2003
Vacio y Multitud 1	Sebastián García Ferro	2004
Vacio y Multitud 2	Sebastián García Ferro	2004
Bajo el Agua	Sebastián García Ferro	2005
Caidas	Sebastián García Ferro	2005
Come Home	Sebastián García Ferro	2005
Flotar	Sebastián García Ferro	2005
Sumergir	Sebastián García Ferro	2005
Back (escena 1)	Sebastián García Ferro	2006
Back (escena 3)	Sebastián García Ferro	2006
Back (escena 5 y 6)	Sebastián García Ferro	2006
Gatos	Sebastián García Ferro	2006
Mandrös	Sebastián García Ferro	2006
Modified - Intro	Sebastián García Ferro	2006
Peces	Sebastián García Ferro	2006
Caras Jazzie End	Sebastián García Ferro	2007
Clock	Sebastián García Ferro	2007
Corn	Sebastián García Ferro	2007
Despertar	Sebastián García Ferro	2007
Fork	Sebastián García Ferro	2007

Mañana	Sebastián García Ferro	2007
Mediodia	Sebastián García Ferro	2007
Metting	Sebastián García Ferro	2007
Noche	Sebastián García Ferro	2007
Pointing	Sebastián García Ferro	2007
Sueños	Sebastián García Ferro	2007
Tarde	Sebastián García Ferro	2007
Vaiven Parte 1	Sebastián García Ferro	2007
Vaiven Parte 2	Sebastián García Ferro	2007
Travellers 1	Sebastián García Ferro	2008
Travellers 2	Sebastián García Ferro	2008
Travellers 3	Sebastián García Ferro	2008
La Lámpara	Sebastián Jara Bunster	2010
For Eric	Sergi Jordá	2001
Big Bang	Sergio Naddei	2011
Rock Memories	Sergio Naddei	2011
The Fly	Sergio Naddei	2011
Windows	Sergio Naddei	2012
Almost New Places	Sergio Naddei	2013
Almost New Spaces	Sergio Naddei	2013
Through Memories 1	Sergio Naddei	2013
Through Memories 2	Sergio Naddei	2013
Through Memories 3	Sergio Naddei	2013
Through Memories 4	Sergio Naddei	2013
Through Memories 5	Sergio Naddei	2013
Reactable	Sergio Naddei	2014
Actions	Sergio Poblete	1998
Místicos I Phonos Fund.Miro	Sáez,Ignacio	1987
El Riu Fosc	Sáez,Ignacio	1988

Horizonte Encadenado	Sáez,Ignacio	1990
For Fernando Riera	Teruyoshi Kamiya	1996
Dance of Stone	Teruyoshi Kamiya	1998
The Machine Manifesto	Thomas Charveriat & Félix Luque	2004
Hemispherical Glitch Study	Tim Schmele	2013
Neurospaces	Tim Schmele	2013
Waiting	Tim Schmele	2013
Seguiriyas	Trino Zurita & Teresa Carrasco	2013
Doll_sa_caustika	Xavi Manzanares	2006
Errortunnel	Xavi Manzanares	2006
H2O	Xavi Manzanares	2006
Massiva	Xavi Manzanares	2006
Nnervits	Xavi Manzanares	2006
Nuvols	Xavi Manzanares	2006
Openspaceinvaders	Xavi Manzanares	2006
Plastiknazzxs	Xavi Manzanares	2006
R4gg4gg4r	Xavi Manzanares	2006
Rezzaka	Xavi Manzanares	2006
Segmentationfault0100	Xavi Manzanares	2006
Segmentationfault1001a	Xavi Manzanares	2006
Segmentationfault1001b	Xavi Manzanares	2006
Standbykut	Xavi Manzanares	2006
Stirofoammentre	Xavi Manzanares	2006
Tripikx	Xavi Manzanares	2006
East Cocker	Xavier Maristany	1984
Remember Me	Xavier Maristany	1999

Table C.1: Phonos catalogue to be used during the exhibition *"Phonos, 40 anys de música electrònica a Barcelona"*.

Bibliography

[1] A. Li-chun Wang, and Th Floor Block F. *An industrial-strength audio search algorithm.* Proceedings of the 4 th International Conference on Music Information Retrieval, 2003.

[2] J.S. Downie. *The Scientific Evaluation of Music Information Retrieval Systems: Foundations and Future.* Computer Music Journal, 28:12-23, 2004.

[3] N. Orio. *Music Retrieval: A Tutorial and Review.* Foundations and Trends®in Information Retrieval, 1(1):1-90, 2006.

[4] M. Schedl, E. Gómez, and J. Urbano. *Music Information Retrieval: Recent Developments and Applications.* Foundations and Trends®in Information Retrieval, 8(2-3):127-261, 2014.

[5] J.J. Aucouturier, and F. Pachet. *Improving Timbre Similarity: How High is the Sky?.* Journal of Negative Results in Speech and Audio Sciences, 1(1):1-13, 2004.

[6] X. Serra, M. Magas, E. Benetos, M. Chudy, S. Dixon, A. Flexer, E. Gómez, F. Gouyon, P. Herrera, S. Jorda, O. Paytuvi, G. Peeters, J. Schlüter, H. Vinet, and G. Widmer. *Roadmap for Music Information ReSearch.* Geoffroy Peeters (editor), Creative Commons BY NC ND 3.0 license, 2013.

[7] M.E.P. Davies, P. Hamel, K. Yoshii and M. Goto. *AutoMashUpper: Automatic Creation of Multi-Song Music Mashups.* IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22(12):1726-1737, 2014.

[8] G. Szymanski. *Pandora, or, a never-ending box of musical delights.* Music Reference Services Quarterly, 12(1):21-22, 2009.

[9] M. Slaney. *Web-scale multimedia analysis: Does content matter?.* IEEE Multimedia, 18(2):12-15, 2011.

[10] J.J. Aucouturier. *Sounds like teen spirit: Computational insights into the grounding of everyday musical terms.* Language, Evolution and the Brain. Frontiers in Linguistics, 35-64, 2009.

[11] S.J. Green, P. Lamere, J. Alexander, F. Maillet, S. Kirk, J. Holt, J. Bourque, and X.W. Mak. *Generating transparent, steerable recommendations from textual descriptions of items.* ACM Conference on Recommender Systems (RecSys'09), 281-284, 2009.

[12] D. Bogdanov, and X. Serra. *From music similarity to music recommendation: Computational approaches based on audio features and metadata.* PhD Thesis, Universitat Pompeu Fabra, 2013.

[13] D. Schnitzer. *Mirage – High-Performance Music Similarity Computation and Automatic Playlist Generation.* Master's Thesis, Vienna University of Technology, 2007.

[14] L. Barrington, D. Turnbull, D. Torres, and G. Lanckriet. *Semantic similarity for music retrieval.* Music Information Retrieval Evaluation Exchange (MIREX), 2007.

[15] K. West, and P. Lamere. *A model-based approach to constructing music similarity functions.* EURASIP Journal on Advances in Signal Processing, 149-149, 2007.

[16] G. Bonnin, and D. Jannach. *Automated Generation of Music Playlists: Survey and Experiments.* ACM Computing Surveys, 47(2), Article 26, 2014.

[17] R. Dias, and M. J. Fonseca. *MuVis: An application for interactive exploration of large music collections.* Proceedings of MM, 1043-1046, 2010.

[18] J.S. Downie. *Music information retrieval.* Annual Review of Information Science and Technology, 37(1):295-340, 2003.

[19] M. Grachten, M. Schedl, T. Pohle, and G. Widmer. *The ISMIR cloud: A decade of ISMIR conferences at your fingertips.* Proceedings of ISMIR, 63-68, 2009.

[20] M. Dopler, M. Schedl, T. Pohle, and P. Knees. *Accessing music collections via representative cluster prototypes in a hierarchical organization scheme.* Proceedings of ISMIR, 179-184, 2009.

[21] I. Schmädecke, and H. Blume. *High performance hardware architectures for automated music classification.* Algorithms from and for Nature and Life, 539-547, 2013.

[22] O. Celma, and X. Serra. *FOAFing the music: Bridging the semantic gap in music recommendation.* Web Semantics: Science, Services and Agents on the World Wide Web, 6(4):250-256, 2008.

[23] F. Pachet, P. Roy, and D. Cazaly. *A combinatorial approach to content-based music selection.* Multimedia, 7(1):44-51, 2000.

[24] D. Bogdanov, and P. Herrera. *How much metadata do we need in music recommendation? A subjective evaluation using preference sets.* WProceedings of ISMIR, 97-102, 2011.

[25] N. Aizenberg, Y. Koren, and O. Somekh. *Build your own music recommender by modeling internet radio streams.* Proceedings of WWW, 1-10, 2012.

[26] R. Van Gulik, and F. Vignoli. *Visual playlist generation on the artist map.* Proceedings of ISMIR, 520-523, 2005.

[27] F. Coelho, J. Devezas, and C. Ribeiro. *Large-scale crossmedia retrieval for playlist generation and song discovery.* Proceedings of OAIR, 61-64, 2013.

[28] O. Celma. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space.* Springer, 2010.

[29] J. McDermott. *Auditory preferences and aesthetics: Music, voices, and everyday sounds.* Neuroscience of Performance and Choice. Elsevier, 227-256, 2012.

[30] A. Germain, and J. Chakareski. *Spotify me: Facebook-assisted automatic playlist generation.* Proceedings of MMSP, 25-28, 2013.

[31] D. Hauger, J. Kepler, M. Schedl, A. Kosir, and M. Tkalcic. *The million musical tweets dataset: What can we learn from microblogs.* Proceedings of ISMIR, 189-194, 2013.

[32] B. Logan. *Music recommendation from song sets.* Proceedings of ISMIR, 425–428, 2004.

[33] F. Vignoli, and S. Pauws. *A music retrieval system based on user driven similarity and its evaluation.* Proceedings of ISMIR, 272-279, 2005.

[34] P. Knees, T. Pohle, M. Schedl, and G. Widmer. *Combining audio-based similarity with Web-based data to accelerate automatic music playlist generation.* Proceedings of MIR, 147-154, 2006.

[35] D. Jannach, M. Zanker, M. Ge, and M. Gröning. *Recommender systems in computer science and information systems — A landscape of research.* Proceedings of EC-Web, 76-87, 2012.

[36] N.H. Liu, S.W. Lai, C.Y. Chen, and S.J. Hsieh. *Adaptive music recommendation based on user behavior in time slot.* Computer Science and Network Security, 9(2):219-227, 2009.

[37] S. McNee, J. Riedl, and J. Konstan. *Being accurate is not enough: how accuracy metrics have hurt recommender systems.* CHI'06 extended abstracts on Human Factors in Computing Systems, p.1001, 2006.

[38] G. Jawaheer, M. Szomszor, and P. Kostkova. *Comparison of implicit and explicit feedback from an online music recommendation system.* Int. Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec'10), p.47-51, 2010.

[39] B. Whitman, and S. Lawrence. *Inferring Descriptions and Similarity for Music from Community Metadata.* Proceedings of the 2002 International Computer Music Conference (ICMC 2002), p.591-598, 2012.

[40] S. Dixon *Onset Detection Revised.* Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx'06), p.133-137, 2006.

[41] J.P. Bello, C. Duxbury, M. Davies, and M. Sandler. *On the use of phase and energy for musical onset detection in the complex domain.* Signal Processing Letters, IEEE, 11(6):553-556, 2004.

[42] P. Masri. *Computer Modeling of Sound for Transformation and Synthesis of Musical Signal.* Ph.D. dissertation, Univ. of Bristol, 1996

[43] J.P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. *A tutorial on onset detection in music signals.* IEEE Transactions on Speech and Audio Processing, 13(5):1035-1047, 2005.

[44] S. Böck, and G. Widmer. *Maximum filter vibrato suppression for onset detection.* Proceedings of the 16th International Conference on Digital Audio Effects (DAFx-13), Maynooth, Ireland, 2013.

[45] N. Degara, E.A. Rua, A. Pena, S. Torres-Guijarro, M.E. Davies, and M.D. Plumbley. *Reliability-informed beat tracking of musical signals.* IEEE Transactions on Audio, Speech, and Language Processing, 20(1):290-301, 2012.

[46] T. Fujishima. *Realtime chord recognition of musical sound: A system using Common Lisp Music.* Proceedings of the International Computer Music Conference, Beijing. 1999.

[47] E. Gómez. *Tonal Description of Polyphonic Audio for Music Content Processing.* INFORMS Journal on Computing, 18(3):294-304, 2006.

[48] J. Bonada. *Automatic technique in frequency domain for near-lossless time-scale modification of audio.* Proceedings of International Computer Music Conference, 396-399, 2000.

[49] G. Cabral, J.P. Briot, and F. Pachet. *Impact of distance in pitch class profile computation.* Proceedings of the Brazilian Symposium on Computer Music, 319-324, 2005.

[50] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra. *ESSENTIA: an open-source library for sound and music analysis.* Proceedings of the 21st ACM international conference on Multimedia, 855-858, 2013.

[51] D. Schnitzer, A. Flexer, and G. Widmer *A fast audio similarity retrieval method for millions of music tracks.* Multimedia Tools and Applications, 58(1):23-40, 2012.

[52] P. Cano, M. Kaltenbrunner, and N. Wack. *An industrial-strength content-based music recommendation system.* Proceedings of the 28th annual internation ACM SIGIR conference on research and development in information retrieval, 673-673, 2005.

[53] P. Roy, J.J. Aucouturier, F. Pachet, and A. Beurivé. *Exploiting the Tradeoff Between Precision and Cpu-Time to Speed Up Nearest Neighbor Search.* ISMIR, 230-237, 2005.

[54] M. Levy, and M. Sandler. *Lightweight measures for timbral similarity of musical audio.* Proceedings of the 1st ACM workshop on Audio and music computing multimedia, 27-36, 2006.

[55] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. *Boostmap: A method for efficient approximate similarity rankings.* Proceedings of the 2004 IEEE Computer Society Conference, 2:II-268, 2004.

[56] P. Cano, M. Kaltenbrunner, F. Gouyon, and E. Batlle. *On the use of FastMap for Audio Retrieval and Browsing.* ISMIR, 2002

[57] C. Faloutsos, and K.I. Lin. *FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets.* Proceedings of the 1995 ACM SIGMOD international conference on management of data, 24(2):163-174, 1995.