IDENTIFICACIÓN AUTOMÁTICA DE SEÑALES DE AUDIO

TESIS

Que para obtener el grado de

DOCTOR EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

José Antonio Camarena Ibarrola

Dr. Edgar Leonel Chávez Gonźalez

Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Enero 2008

To my wife Susy and my kids Alejandra and Jorge

Acknowledgments

Thanks to Dr. Edgar Chávez, this work would not have been possible without his dedication.

Thanks to Dr. Juan Manuel Garcia, Dr. Juan Flores, Dr. Juan Nolazco, and Dr. Leonardo Romero for reviewing this work so patiently and for the valuable recommendations that definitively increased the quality of this thesis report.

Thanks to the Universidad Michoacana de San Nicolas de Hidalgo (UMSNH) for all the facilities given to me for studying and obtaining the degree.

Thanks to the Consejo Nacional de Ciencia y Tecnología (CONACYT) for the support given for the realization of this work.

Thanks to Dr. Felix Calderon for his advises and friendship.

List of Publications

"A Robust Entropy-Based Audio-Fingerprint"
Antonio C. Ibarrola, Edgar Chavez
International Conference on Multimedia and Expo (ICME 2006)
July, 2006 Toronto, Canada

"Using a New Discretization of the Fourier Transform to Discriminate Voiced from Unvoiced Speech"
Antonio Camarena-Ibarrola, Edgar Chavez
7o En- cuentro Internacional de Ciencias de la Computacion (ENC'2006)
September, 2006. San Luis Potosí, México

"On Musical Performances Identification, Entropy and String Matching"
Antonio Camarena-Ibarrola, Edgar Chavez
Mexican International Conference on Artificial Intelligence MICAI 2006
November, 2006 Tlaxcala, Mexico

"Robust Audio-Fingerprinting with Spectral Entropy Signatures"
Antonio Camarena-Ibarrola, Edgar Chavez
Submitted for revision

"Matching Musical Performances Using String Processing Techniques With Stable Features"
Antonio Camarena-Ibarrola, Edgar Chavez
Submitted for revision

Resumen

Dada una colección de señales de audio, descubrir si una nueva señal de audio se identifica con alguno de los miembros de la colección, este problema se vuelve interesante cuando las señales a reconocer han sufrido deformaciones como ecualización, contaminación por ruido, compresión con pérdidas, regrabación, etc. El problema se complica cuando se deben identificar piezas musicales que han sido ejecutadas por diferentes artistas. Este problema se conoce como *Identificación automática de señales de audio*.

Las *Huellas de Audio* también conocidas como *Firmas de Audio* reemplazan a las señales de audio con fines de identificación. La huella mas robusta encontrada en la literatura se basa en una medida de lo plano que resulta ser el espectro de la señal, esta huella forma parte del estándar de MPEG-7 [1]. La huella mas citada en la literatura fué desarrollada por Haitsma y Kalker en los laboratorios de Philips Research, esta huella se ha convertido en una referencia obligada.

En esta tesis presentamos dos huellas de audio basadas en la entropía de Shannon. La primera de ellas a la que denominamos TES (*Time domain Entropy Signature*) es mas robusta que la huella de Haitsma y Kalker para canciones completas bajo todas las deformaciones contempladas y se calcula tres veces mas rápido. Por otro lado MPEG-7 es mas robusta que TES bajo ecualización, siendo ambas igualmente efectivas para las demás deformaciones. A la segunda huella la denominamos MBSES (*Multi Band Spectral Entropy Signature*). Con MBSES logramos el 100 % de reconocimiento incluso haciendo mas severas las deformaciones (ninguna de las huellas anteriores logra el 100 % de reconocimiento en estas condiciones). En una colección de 4000 canciones de géneros diversos, al utilizar MPEG-7 solo se logró identificar al 63 % de las canciones contaminadas por ruido, al 79 % de las re-grabadas y al 82 % de las filtradas. Con Haitsma-Kalker solo se pudo identificar al 40 % de las ecualizadas, al 20 % de las contaminadas por ruido, al 10 % de las re-grabadas, al 70 % de las filtradas y al 80 % de las comprimidas. Finalmente, en la identificación de diferentes interpretaciones de una misma pieza musical, usando MBSES se obtuvo una precisión de 99.2 %, mientras que las firmas de audio de Haitsma-Kalker, MPEG-7 y TES lograron una precisión de 97.6 %, 93.75 % y 88.3 % respectivamente.

[1] MPEG-7 es un estandar desarrollado por MPEG (Moving Pictures Expert Group). El nombre formal de MPEG-7 es "Interfaz para la descripción de contenidos multimedia". Este estándar no solo describe el contenido multimedia sino un cierto grado de interpretación o significado del mismo y en que forma esa descripción puede ser transmitida o accedida por un dispositivo o computadora.

Abstract

Given a set of audio signals, find out if a new audio signal may be identified with one of the members of the set, this problem becomes interesting when the signal to identify has been deformed by equalization, noise contamination, lossy compression, re-recording, etc.,. The problem is even more complex if the audio-signals to be identified are actually musical pieces performed by different artists. This problem is known as *Automatic Audio-signal Identification*.

Audio-Fingerprints also known as *Audio-Signatures* replace the audio signals for identification purposes. The most robust Audio-fingerprint (AFP) found in the literature is based in a measure of the flatness of the spectrum of the signal, this AFP is now part of the standard of MPEG-7 [2]. The most cited AFP was developed by Haitsma and Kalker at the Philips Research Labs, this AFP is a classic reference in the literature.

In this thesis report, two AFPs based on Shannon's entropy are introduced. The first one which we named TES (Time-domain Entropy Signature) is more robust than Haitsma-Kalker's AFP, it is very small and fast to compute since it is extracted directly in time domain. TES is more robust than Haitsma-Kalker's AFP for whole songs under every deformation considered in this work, still TES is computed three times faster. The AFP of MPEG-7 is as effective as TES for every deformation considered in this work except for equalization where MPEG-7 was found to be more robust. For identifying audio-signals using 5 seconds excerpts we introduce the *Multi-Band Spectral Entropy Signature (MBSES)*. With MBSES we achieved 100 % of precision rate for severe deformations (For such levels of deformations none of the other AFPs gets 100 % of precision rate). In a collection with 4000 songs of various gneres MPEG-7 could only identify 63 % of the songs contaminated with noise, 79 % of the re-recorded ones, and 82 % of the filtered ones. With Haitsma-Kalker's AFP only 40 % of the equalized songs could be identified, 20 % of the songs contaminated by noise, 10 % of the re-recorded ones, 70 % of the filtered ones, and 80 % of the lossy compressed ones. Finally, at identifying music performed by different artists, MBSES obtained a precision rate of 99.2 %, while TES, Haitsma-Kalker's AFP, and the AFP of MPEG-7 obtained a precision rate of 88.3 %, 97.6 %, 93.75 % respectively.

[2]MPEG-7 is an standard developed by the Moving Pictures Expert Group (MPEG), formally named "Multimedia contents description interface", this is a standard for describing the multimedia content data that supports some degree of interpretation of the information meaning, which can be passed onto, or accessed by, a device or a computer.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

H, H_S	Shannon's Entropy
H_{R_α}	Renyi's Entropy of order α
μ_x	Mean of variable x
σ_x	Variance of variable x
I	Information
p_i	Probability of value v_i to appear in the signal
$p(x)$	Probability distribution function
n_b	Number of values from the spectrum that correspond to band b
c	Module spectrum
SFM_b	Spectral Flatness Measure for band b
$D_M(x, y)$	Mahalanobis distance between x and y
$\delta(k)$	Unitary impulse located at k
$\varphi(x, y)$	Function for comparing x and y reports 1 if they are equal and zero otherwise
f_i	frequency of value i
D_E	Episode Distance
L	Levenshtein's Distance
C	Longest Common Subsequence distance
D	Dynamic time warping distance
f	Time Warped LCS similarity measure
F	Time Warped LCS distance
$\mathcal{N}(\mathbf{0}, \mathbf{R})$	Normal probability distribution with zero mean and covariance matrix $\mathbf{R}$
$SE_k(t)$	Spectral Entropy Value for band k and frame t
SE_k	Spectral Entropy Curve for band k

Chapter 1

Introduction

We are interested in the automatic identification of audio-signals such as songs and commercial spots. We want to be able to identify songs even if they have been severely degraded by equalization, noise contamination, lossy compression, re-recording, low-pass filtering and scaling (i.e. volume variation). We want to be able to recognize a song using a small excerpt of only a few seconds of it. In the literature, a excerpt of an audio-signal is sometimes considered as an audio-signal that has suffered a deformation called *cropping*. We are also interested in the ability to identify a song independently of the performing artists, perhaps even using a different set of instruments, such a generalization of the audio identification problem known as *Polyphonic Audio Matching* requires the use of aligning techniques to deal with rhythm variations.

Just as a human being is born with fingerprints that are not supposed to have significative changes throughout his/her life term. An *Audio-Fingerprint* (AFP) is a small representation of an audio-signal that is not supposed to suffer major changes when the audio-signal is slightly degraded or deformed. To understand why audio signals are degraded in practically every real world application, consider the following examples:

- In Querying by melody [Shalev-Shwartz02] a small excerpt of only a few seconds of a song is used to identify it, the exact beginning of the excerpt is random for all practical purposes and the probability of any frame to begin at exactly the same sample with a frame of the whole song is approximately zero. This particular deformation in known

as time-shift and is considered unavoidable.

- If a song is to be identified using a piece of audio captured with a microphone [Haitsma02], the noise in the environment will be recorded along with the music.

- Songs stored in a lossy compressed format (i.e. MP3) are degraded songs, the lower the rate bit, the more deformed they are.

An AFP should be robust to signal degradations, it should also be compact for all storage or transmission purposes. In addition, an AFP should be determined with as little computational effort as possible. Finally, an audio-fingerprinting system should be scalable, meaning that it should be able to work with hundreds or thousands of songs with a good performance both in time and precision.

1.1. Objective

It is the objective of this work to design a more robust audio signature than the state-of-the-art audio signatures with respect to equalization, noise contamination, low- pass filtering, lossy compression, time shifting, scaling, loudspeakers to microphone transmission and cropping. The desired audio signature should allow for identification of audio-signals using small excerpts of a few seconds. The audio signature should be adequate for matching musical performances.

1.2. Thesis Statement

By tracking how the information content of an audio-signal evolves both in time and frequency, a very robust audio-fingerprint can be extracted. Such an audio-fingerprint would allow the identification of the corresponding audio-signal even if it has been severely degraded.

1.3. Motivation

We were motivated by the fact that the applications of a robust Audio-Fingerprint cover a wide spectrum, some audio-fingerprinting applications are listed below:

1. *Radio Broadcasts Monitoring*. Sometimes to assess sponsorship effectiveness, others for auditory purposes, people are actually being employed just to monitor a radio station and keep track of the commercial or political spots that are being transmitted by the radio station in turn. Many people are needed to monitor 24 hours a day all the local radio and tv stations, such a job may be performed by computers equipped with multi channel FM/TV cards [Shin02].

2. *Duplicate detection*. To discover if two audio files store the same song independently of the bit rate and compression format is very useful for maintaining the integrity of a multimedia database since it allow us to avoid duplicates.

3. *Automatic labeling*. Modern MP3 [1] players provide the user with tools for organizing songs, these tools rely in the contents of meta-data labels (e.g. Album's title), when these labels are empty they can be automatically filled up using audio-fingerprinting techniques. [Mus02].

4. *Querying by example*. A song may be identified using a small excerpt of audio captured for example by a cell phone as explained in [Haitsma02].

5. *Filtering in p2p networks*. When music is transmitted in a peer to peer network, the audio-fingerprint is determined from the packets and searched for in a list of copyrighted songs to prevent illegal copies [Shrestha04].

The most robust Audio-fingerprint (AFP) found in the literature is based in a measure of the flatness of the spectrum of the signal, this AFP is now part of the standard of MPEG-7 [2]. The most cited AFP was developed by Haitsma and Kalker at the Philips

[1] MP3 is shorthand for the standard MPEG-1 layer 3 developed by the Moving Pictures Expert Group (MPEG) where the codec for audio compression is defined.

[2] MPEG-7 is an standard developed by the Moving Pictures Expert Group (MPEG), formally named "Multimedia contents description interface", this is a standard for describing the multimedia content data that supports some degree of interpretation of the information meaning, which can be passed onto, or accessed by, a device or a computer.

Research Labs, this AFP is a classic reference in the literature. We believe Haitsma-Kalker's AFP has become an obligated reference.

In this work, two audio-fingerprints based on Shannon's entropy are introduced. The first one, the *Time-domain Entropy Signature* (TES) is very small and fast to compute since it is extracted directly in time domain. TES turned out to be more robust than Haitsma-Kalker's AFP. TES however, did not work well for the specific deformation of equalization, this fact motivated us to design the *Multi-Band Spectral Entropy Signature* (MBSES). MBSES turned out to be more robust than Haitsma-Kalker's AFP and more robust than the state-of-the-art AFP of MPEG-7 as well. The superior robustness of our MBSES is more evident under severe degradations such as noise contamination with a Signal to Noise Ratio (SNR) of approximately 3.5 dB.

The experiments were carried out using a collection of 4000 songs of almost every genre. Using MBSES we managed to identify 100 % of the songs independently of the kind of degradation. Using TES we identified 100 % of the songs deformed by low-pass filtering, lossy compression and scaling, unfortunately we could only identify 53.7 % of the equalized songs, 63 % of the noisy songs, and 91 % of the re-recorded ones. With the AFP of MPEG-7 100 % of the equalized, lossy compressed or scaled songs were identified, but only 55.3 % of the noisy songs, 80 % of the re-recorded ones and 72.1 % of the filtered ones.

In the experiments where 5 seconds excerpts were used as queries, the robustness of MBSES was confirmed since 100 % of the songs were identified independently on the kind of degradation. With the audio-fingerprint of MPEG-7 100 % of the lossy-compressed, scaled, time-shifted, and equalized songs were identified. However only 63 % of the noisy songs, 79 % of the re-recorded and 82 % of the filtered ones could be identified. Using Haitsma-Kalker's AFP, 100 % of the time-shifted songs were identified but only 40 % of the equalized, 20 % of the noisy songs, 10 % of the re-recorded, 70 % of the filtered, 90 % of the Scaled, and 80 % of the compressed were identified.

Finally, the experiments on matching musical performances reported a precision rate of 99.2 % when MBSES was used in combination with the algorithm of the Longest Common Subsequence (LCS) as the alignment technique. With Haitsma-Kalker's audio-fingerprint a precision rate of 97.6 % was reported. with the audio-fingerprint of MPEG-7

a precision rate of 93.75 % was obtained. TES achieved a precision rate of 88.3 %.

The rest of this work is organized as follows: In the next chapter, a review of techniques for feature extraction and audio modeling is given. Extraction of the AFP of MPEG-7 and Haitsma-Kalker's AFP is given in detail. In the same chapter (Chapter 2) there is a survey on the problem of the matching of musical performances and some preliminaries on aligning techniques are discussed. In chapter 3 our new entropy based signatures are defined. In chapter 4 the experiments are detailed and the results are reported. Finally in chapter 5 the results are discussed and future work is proposed.

Chapter 2

State of the Art

Audio-Fingerprints are used to asses the perceptual similarity between audio-signals. Ideally, an AFP should be an *invariant* of the audio signal, an intrinsic characteristic found in it even if it has suffered severe degradations as long as it is still recognizable.

2.1. Characteristics of an AFP

For an AFP to be successful, there are several issues to consider, the most important of them are listed below:

1. *Robustness*. Audio signals may be subject to a variety of signal degradations such as noise contamination, lossy compression, loudspeaker to microphone transmission (LsMic), low-pass filtering simulating narrow band telephone line transmission, equalization, cropping, time shifting and loudness variation. The AFP of a song should not be too different from the AFP of a degraded version of the same song.

2. *Compactness*. Some applications need to store the AFP of every song from a possible large collection; other applications need to transmit the AFP over the internet, these facts make compactness a very desirable characteristic of an AFP.

3. *Granularity*. Some *Music Information Retrieval* applications require the ability to identify a song with a small excerpt of it. For example, in *querying by humming* we do

not want the user having to hum the entire song he/she is searching for. Granularity is also known as *robustness to cropping*.

4. *Time complexity*. The AFP should be determined with as little computational effort as possible. The AFP of every song from the collection has to be determined in a reasonable time. Real time systems have to extract the AFP of a song on line, furthermore, in *Radio Broadcasts Monitoring* it is desirable to be able to compute the AFP of several audio channels simultaneously.

5. *Scalability*. It is defined as the ability of an audio fingerprinting system to operate with a large collection of songs, this feature is conditioned by a low time complexity, compact AFP as well as a good indexing technique.

2.2. Audio-Fingerprint Survey

The most important aspect of an audio-fingerprint is the selection of the perceptual feature it is based on. The perceptual features considered to be the more relevant for audio-fingerprinting purposes in the literature are:

- *Loudness*. Loudness is a Psycho-Acoustic measure of how loud or soft a sound is really heard. Loudness can be measured by judging a ratio of two sensations produced by two given stimuli. For loudness evaluations, the simplest ratio is doubled and halved, the subject searches for the level increment that leads to a sensation that is twice as loud as that of the starting level. The average indicates that the level of a 1-KHz tone has to be increase by 10 dB in order to enlarge the sensation of loudness by a factor of two [Zwicker90].

- *Joint Acoustic and Modulation Frequency* (JAMF). Sukitatton [Sukittanon02] claimed that incorporating modulation spectral features into signal classification provided some improvement over systems that used only short-term features. The *modulation spectrum* of an audio signal $P(\eta, \omega)$ not only contains short-term information about the signals, but also provides long-term information representing patterns of time variation. η represents the Modulation frequency and ω represents the Acoustic

or Fourier frequency. To determine $P(\eta, \omega)$, first a joint time-frequency representation of the signal (i.e. a spectrogram) $P(t, \omega)$ is estimated (t represents time), then, a Fourier transform is applied along the time dimension of the spectrogram, yielding an estimate of the modulation spectrum also called Join Acoustic and Modulation Frequency representation of the signal [Sukittanon05].

- *Mel-Frequency Spectral Coefficients (MFCC).* These coefficients are among the most preferred features for building audio-fingerprints. To determine the Mel-Frequency Spectral Coefficients of a windowed short segment of an audio signal, first the Discrete Fourier Transfrom (DFT) is computed. From the spectrum, the so-called Mel filter bank is applied, this filter bank consists of M (frequently 30) overlapped triangular filters with central frequencies positioned logarithmically (on the Mel scale). From the resulting list of M log-amplitudes the Discrete Cosine Transform is determined, as if this M values made a short signal. The Mel-Frequency Spectral Coefficients are the amplitudes of the resulting spectrum [Sigurdsson06], [Logan00].

- *Spectral Crest Factor* (SCF). The SCF is the ratio of the largest magnitude and the mean magnitude from the spectral coefficients of a band. This value is always greater or equal than 1. If it is near 1, then the signal is surely noisy. If the SCF is far from 1 then the signal has at least an important spectral component (an Harmonic), suggesting the signal is more tone-like. This feature belong to a family of flatness oriented spectral features [Herre01].

- *Spectral Flatness Measure* (SFM). This state-of-the-art feature will be widely discussed in Subsection 2.4.1.

- *Spectral Sub-band Centroids* (SSC). Also called *Spectral Sub-band moments*. The frequency centroid has been found to be related to the human sensation of the *brightness of a sound* [Seo05]. The subband moment of order v at the i-th subband of an audio spectrum $P[k, m]$ is defined as in Equation 2.1.

$$M_i^v[m] = \sum_{k=CB_i+1}^{CB_{i+1}} k^v P[k, m] \tag{2.1}$$

where k, m and CB_i denote the frequency bin, the frame index, and the frequency boundary of the i-th critical band respectively.

The SSC for band i and frame m ($C_i[m]$) is the first-order normalized moment given as in Equation (2.2) [Sungwoong07].

$$C_i[m] = \frac{M_i^1[m]}{M_i^0[m]} \tag{2.2}$$

- The sign of Energy's second derivative. Also called "String Hash" by Haitsma and Kalker [Haitsma02], [Sinitsyn06]. This feature will be widely discussed in Subsection 2.4.2.

- *Chroma Values.*

  Two tones separated by an integral number of octaves share the same value of chroma. This is an intuitive concept for musicians, since chroma is closely related to the musical-theoretic concept of pitch class. Even thought Chroma values have been used for audio-fingerprinting [Bartsch05], this feature is mostly used for the estimation of musical keys from the audio-signal [Pauws04], [VandePar06]. A chromagram represents the likelihood of the chroma occurrences in the audio-signal. To determine a chromagram, the tonal (sinusoidal) components need to be extracted. The initial tonal component selection is based on a peak-picking method Fourier domain that selects local maxima. The chromagram of an audio-signal is defined as a restructuring of the spectrum of the audio-signal in which the frequencies are mapped onto a limited set of 12 chroma values in a many-to-one fashion. This is done by assigning frequencies to the "bin" that represents the ideal chroma value. The "bins" correspond to the twelve chromas in an octave.

Seo [Seo05] shows how the *Normalized SSC* are more robust than the *Mel-frequency Cepstral Coefficients (MFCC)* and *Tonality* [Hellman72] for lossy compression and equalization. Sukittanon [Sukittanon02] reported that the Normalized JAMF has superior robustness than a "spectral estimate" for compression and equalization. Herre [Herre01] reported that SFM has superior robustness than *Loudness* and SCF as well. SFM was then adopted

by MPEG-7 for audio fingerprinting purposes [Group01]. Sert [Sert06] reported that SFM was more robust than MFCC. Kurth [Kurth03] found that the sign of the time derivative of the signal was robust to lossy compression and low-pass filtering.

2.3. Audio-Fingerprint Extraction

The extraction of the audio-fingerprint of an audio-signal is performed by *Front-End* and the *audio modeling* modules as depicted in Figure 2.1. In the *Front-end* module, the perceptual features of the audio-signal are extracted. Since an audio-fingerprint is intended to represent the audio-signal, the features delivered by the *front-end* are taken by the *audio-modeling* module with the purpose of building an AFP that is adequate for *Information Retrieval* purposes.

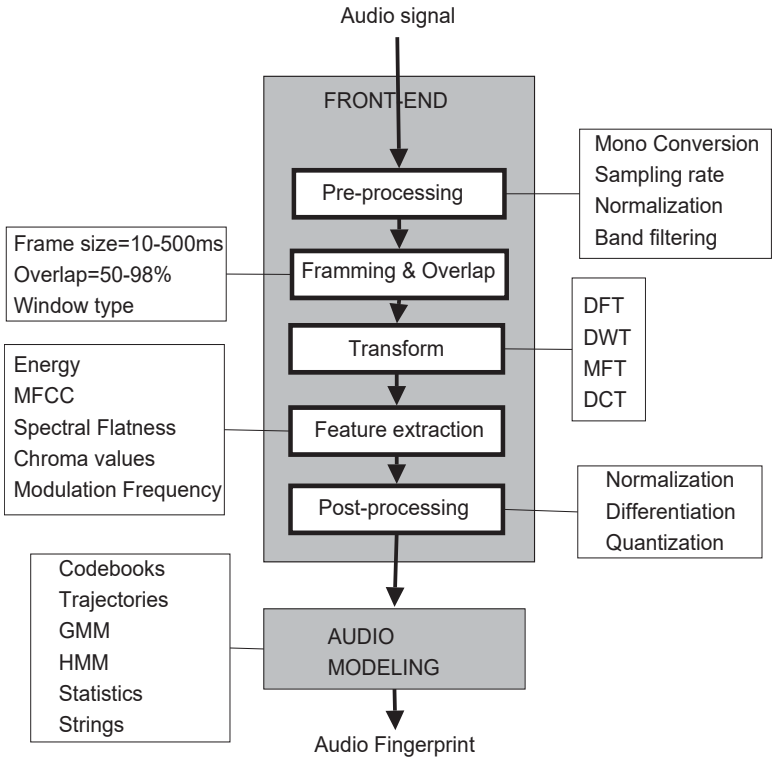


Figure 2.1: Audio-Fingerprint Extraction Framework [Cano02]

2.3.1. The Front-End Module

The audio signal is normally processed on a *frame by frame* basis. A *frame* of signal is a short segment of audio to which a "window" is applied. The Parzen window the Hann window and the Hamming window are good examples [Rabiner78]. Windows emphasize the signal near the middle of the frame and fade the signal near the end points of the frame. When no window is applied it is said that the *rectangular window* is being used.

Stereo audio signals are normally converted to mono-aural and an amplitude normalization is frequently used to make the AFP robust to changes in volume. The frame should contain at least two periods of the lowest frequency component of the signal that it is being considered, therefore the frame is never shorter than 10ms. On the other hand, the frame should not be longer than 500 ms in order to consider the signal as stationary.

A large overlap is normally used to ensure a slow variation of the extracted features, the exception of the case is the audio-fingerprint of MPEG-7 which is determined without overlapping. The reason for using an overlap of zero is that the AFP of MPEG-7 is determined using the average of the features every 32 frames, so there is no need for a slow variation of the feature vectors. The smaller the overlap the fewer the number of computations needed to determine the AFP.

There actually are systems that extract signal features directly in time domain as in [Kurth03]. However, most systems extract signal features in the frequency domain using a variety of linear transforms such as the Discrete Cosine Transform, the Discrete Fourier Transform, the Modulation Frequency Transform [Sukittanon02] and some Discrete Wavelet Transforms like Haar's and Walsh-Hadamard's [Subramanya99]. Once in the frequency domain, there is a variety of features that have been used as the relevant characteristic of audio signal as described in the previous paragraph (Energy, MFCC, SFM, etc.,). Some AFP systems perform some post-processing of the features, for example by time-deriving the features, the information of how fast such features change over time would be available.

2.3.2. The Audio Modeling Module

The Audio signal is modeled in a way that best serves the purpose of the application for which the audio-fingerprint has been designed. Some existing audio models are:

- *Sequences of Feature Vectors.* This kind of AFPs are also known as *trajectories* or *traces*. The features extracted at equally spaced periods of time are simply stored in a list of vectors or in a table, one row per frame. An example of this kind of AFP is the binary vector sequence described in [Haitsma02].

- *Statistics.* Instead of storing every feature vector, only statistical data over the set of feature vectors are stored. The audio-fingerprint designed for MPEG-7 computes the means, variances, minima and maxima every 32 frames. The minima and maxima are used for delimiting the search and the means and variances are used for the actual search using some distance measure such as the Mahalanobis' distance [Hellmuth01].

- *Codebooks.* The sequence of feature vectors extracted from a song is replaced by a small number of representative code vectors stored in a *codebook*, which from then on represents the song. This model disregards the temporal evolution of the audio signal [Allamanche02].

- *Strings.* Trajectories can be converted into long strings of integers using vector quantization. This model allows the treatment of songs as texts that may be compared using flexible string matching techniques [Guo04].

- *Single vectors.* These are perhaps the smallest AFPs. They are usually built with average features extracted from the whole song, for example, an AFP may be a vector containing the beats per minute, the average zero crossing rate and the average spectrum [Mus02].

- *Hidden Markov Models (HMM).* These finite state machines model non stationary stochastic processes (e.g. songs). For each song of the collection a HMM is built. The features extracted from the test song are considered to be a sequence of acoustic events and then used as the input for the candidate's HMM. The candidate's HMM in turn reports the probability that the test song matches the candidate song, this probability is used as a proximity measure for choosing the right song [Batlle04a],[Batlle04b].

- *Gaussian Mixture Models (GMM).* An audio-signal is modeled by a probability density function (PDF). If we assume that such PDF is the result of a combination of gaussian

components or *mixtures*, then the parameters (i.e. mean and variance) of every component have to be estimated. The estimation of the parameters is made by maximizing the probability of the audio frames actually present in the audio signal. For this maximization the Baum-Welch or Expectation-Maximization (EM) Algorithm [Bilmes98] is normally used. In [Ramalingam05] an audio clip modeled by a GMM made out of 16 mixtures is searched in a database of GMMs, the GMM that gives the highest likelihood is assumed as a match. In [Lin06] it was assumed that all the audio-segments shared a common set of gaussian components and only the weights of those common components were estimated, they called it *Common Components GMM* or *CCGMM*. The Kullback-Leibler distance (also known as joint entropy) was used in [Lin06] to measure the dissimilarity between two probabilistic models (i.e. Audio-fingerprints).

2.4. Two Successful Audio-Fingerprints

Several years were needed for the definition of the standard that will be used by MPEG-7 for audio-fingerprinting purposes. It is also true that the audio-fingerprint developed in the Philip's Research Labs has been the classical reference for robust audio-fingerprinting for years, it is known as Haitsma-Kalker's Hash String. In this section both AFPs will be detailed.

2.4.1. The Standard for Audio-Fingerprinting in MPEG-7

In Figure 2.2 a block diagram of the process for the determination of MPEG-7's AFP of an audio signal is depicted. The AFP of MPEG-7 is a configurable audio signature with variable resolution, where the maximum number of bands allowed is 24. The maximum resolution allowed by the MPEG-7 standard is 1/4 of an octave. The range of frequencies to take into consideration is also configurable, the widest range allowed covers the frequencies between 250 Hz and 16 KHz. There is always an overlap of 5 percent between bands. The frame size is fixed at 30 milliseconds with no overlapping between frames. The Hann window [Stanley84] is applied to every frame. Every 32 frames the minimum, maximum, mean and variance of the SFM values are determined for each band. Therefore, a vector is added to

the AFP every 0.96 seconds.

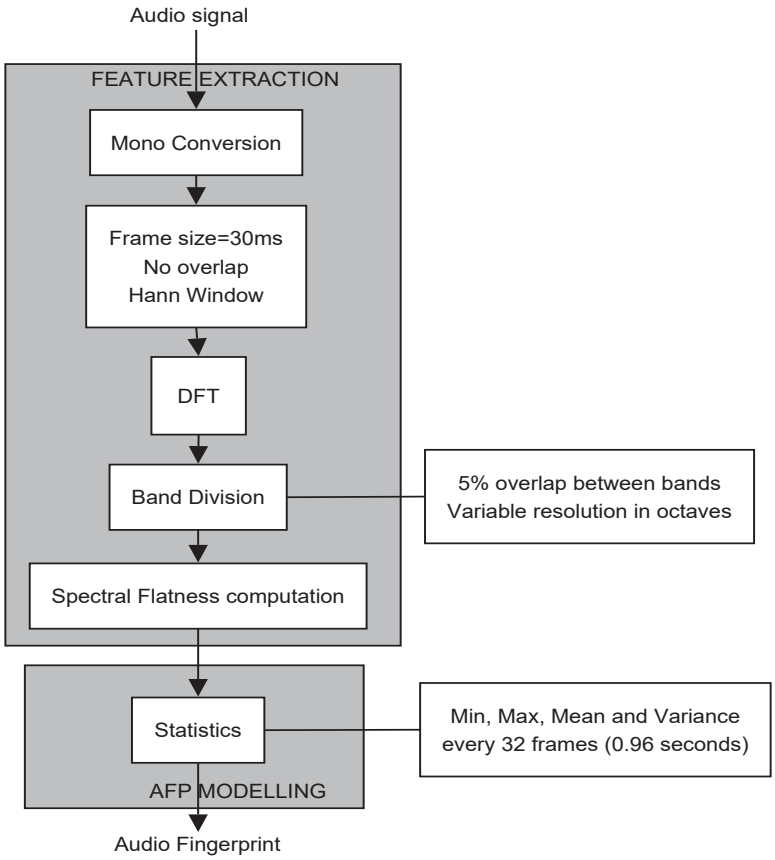


Figure 2.2: Block diagram for the determination of the AFP of MPEG-7 from an audio signal

The SFM is a feature related to the tonality aspect of the audio signal. The SFM is defined as the ratio of the geometric mean and the arithmetic mean of the power spectrum coefficients. The SFM for band b with bandwidth n_b is computed with Formula (2.3).

$$SFM_b = \frac{\left[\prod_{i=1}^{n_b} c(i)\right]^{\frac{1}{n_b}}}{\frac{1}{n_b}\sum_{i=1}^{n_b} c(i)} \tag{2.3}$$

where c is the vector where the power spectrum coefficients are stored.

The SFM reports values between zero and one, values near one indicate that the spectrum is flat and the audio is noisy, while values near zero show that the audio signal is more tone-like.

The values determined by the last block depicted in Figure 2.2 are stored in XML files. Suppose a small AFP of an audio signal of only two seconds was determined, this particular AFP was determined using only 4 bands (i.e. vector size) in the range of 500Hz to 1000Hz, there are eight values for each row corresponding to 2 vectors (i.e. elementNum). One row stores the min values, another row stores the max values, another row is for the means and another for the variances. The XML file would look as follows:

```
<AudioDescriptor xsi:type="AudioSpectrumFlatnessType" loEdge="500"
hiEdge="1000">
   <SeriesOfVector vectorSize="4" totalSampleNum="128">
      <Scaling ratio="64" elementNum="2">
      <Min dim="2 4"> 0.3 0.1 0.4 0.3 0.2 0.3 0.3 0.2 </Min>
      <Max dim="2 4"> 0.8 0.5 0.7 0.6 0.5 0.6 0.6 0.8 </Max>
      <Mean dim="2 4"> 0.6 0.4 0.5 0.4 0.4 0.5 0.4 0.4 </Mean>
      <Variance dim="2 4"> 0.1 0.11 0.06 0.08 0.07 0.1 0.09 0.07 </Variance>
   </SeriesOfVector>
</AudioDescriptor>
```

The minima and maxima are useful for delimiting the search in the collection and the means and variances may be used to estimate how close AFP x and AFP y are, using the Mahalanobis distance for that matter as in Equation (2.4).

$$D_M(x,y) = \sqrt{\sum_{i=1}^{NumBands} \frac{(\mu_{x_i} - \mu_{y_i})^2}{\sigma_{x_i} + \sigma_{y_i}}} \tag{2.4}$$

To increase a frequency in one octave is equivalent to multiply it by two. Increasing a frequency in three octaves is equivalent to multiplying it three times by two (i.e. 2^3).

Increasing the frequency by $\frac{1}{4}$ of an octave is done by multiplying it by $2^{0.25}$. Finally, the frequency that is located m fourths of an octave above f is $2^{0.25m}f$. The smallest frequency allowed by the standard of MPEG-7 to build AFPs is 250Hz, so the 24 bands used in the highest resolution allowed (i.e. a forth of a octave) are listed on Table 2.1 with and without the 5 percent overlapping between bands.

Table 2.1: Bands for the maximum resolution of MPEG-7's AFP

Band	Without overlapping	Overlapped
1	250.00 - 297.30 Hz	237.50 - 312.17 Hz
2	297.30 - 353.55 Hz	282.44 - 371.23 Hz
3	353.55 - 420.45 Hz	335.88 - 441.47 Hz
4	420.45 - 500.00 Hz	399.43 - 525.00 Hz
5	500.00 - 594.60 Hz	475.00 - 624.33 Hz
6	594.60 - 707.11 Hz	564.87 - 742.46 Hz
7	707.11 - 840.90 Hz	671.75 - 882.94 Hz
8	840.90 - 1000.00 Hz	798.85 - 1050.00 Hz
9	1000.00 - 1189.21 Hz	950.00 - 1248.67 Hz
10	1189.21 - 1414.21 Hz	1129.75 - 1484.92 Hz
11	1414.21 - 1681.79 Hz	1343.50 - 1765.88 Hz
12	1681.79 - 2000.00 Hz	1597.70 - 2100.00 Hz
13	2000.00 - 2378.41 Hz	1900.00 - 2497.33 Hz
14	2378.41 - 2828.43 Hz	2259.49 - 2969.85 Hz
15	2828.43 - 3363.59 Hz	2687.01 - 3531.76 Hz
16	3363.59 - 4000.00 Hz	3195.41 - 4200.00 Hz
17	4000.00 - 4756.83 Hz	3800.00 - 4994.67 Hz
18	4756.83 - 5656.85 Hz	4518.99 - 5939.70 Hz
19	5656.85 - 6727.17 Hz	5374.01 - 7063.53 Hz
20	6727.17 - 8000.00 Hz	6390.81 - 8400.00 Hz
21	8000.00 - 9513.66 Hz	7600.00 - 9989.34 Hz
22	9513.66 - 11313.71 Hz	9037.97 - 11879.39 Hz
23	11313.71 - 13454.34 Hz	10748.02 - 14127.06 Hz
24	13454.34 - 16000.00 Hz	12781.63 - 16800.00 Hz

2.4.2. Haitsma-Kalker's AFP

Haitsma-Kalker's AFP was developed in the Philips research Labs, their goal was to design an audio-fingerprint for identifying songs as captured and transmitted by a cell phone using only three seconds of audio signal for that purpose (i.e. Granularity=3 sec).

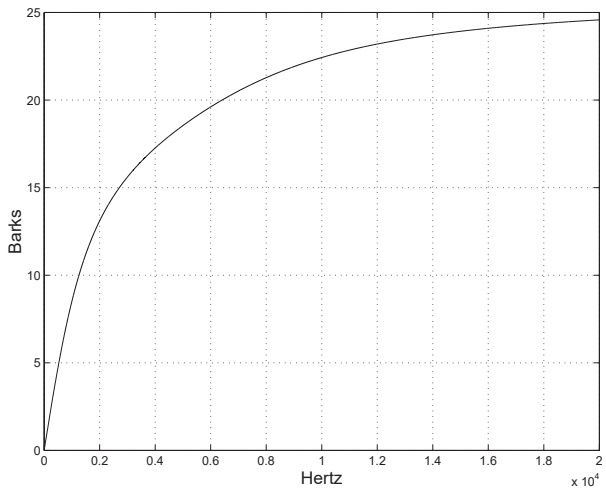


Figure 2.3: Bark scale

To extract Haitsma-Kalker's AFP the audio-signal is processed in frames of 0.37 seconds with an overlap of 31/32, therefore a frame is processed every 11.6 milliseconds (i.e. The Frame Rate is 86.2 frames per second). To each frame the Hann window is applied and then the Discrete Fourier Transform is determined. The spectral coefficients are grouped in 33 bands ranging from 300 Hz to 2000 Hz uniformly distributed in the Bark scale.

To convert any frequency in Hertz to Barks units, Equation (2.5) may be used. In Figure 2.3 the correspondence between Barks and Hertz is shown. Some of the 33 bands used by Haitsma and Kalker are shown on Table 2.2.

$$z = 13tan^{-1}\left(\frac{0.76f}{1000}\right) + 3.5tan^{-1}\left(\frac{f}{7500}\right)^2 \tag{2.5}$$

where f is the frequency in Hertz and z is the frequency in Barks

For each band the Energy is computed and from these 33 energy values, the differences between each consecutive band are determined resulting in a vector with 32 values for each frame. Each of the 32 values corresponding to a frame are compared with the corresponding value of the next frame, a single bit indicates which one is greater. This indicator bit is added to the "Hash string". The Hash string is a binary matrix conformed by 256

Table 2.2: Bands used for Haitsma-Kalker's AFP

Band	Hertz	Barks
1	200 - 243	1.9635 - 2.3768
2	243 - 330	2.3768 - 3.2034
3	330 - 420	3.2034 - 4.0300
4	420 - 514	4.0300 - 4.8566
:	:	:
32	2612 - 2798	14.7758 - 15.1891
33	2798 - 3000	15.1891 - 15.6024

binary vectors of 32 bits each. The Hash String represents 3 seconds of audio [Haitsma02]. Figure 2.4 shows the extraction framework of Haitsma-Kalker's AFP.

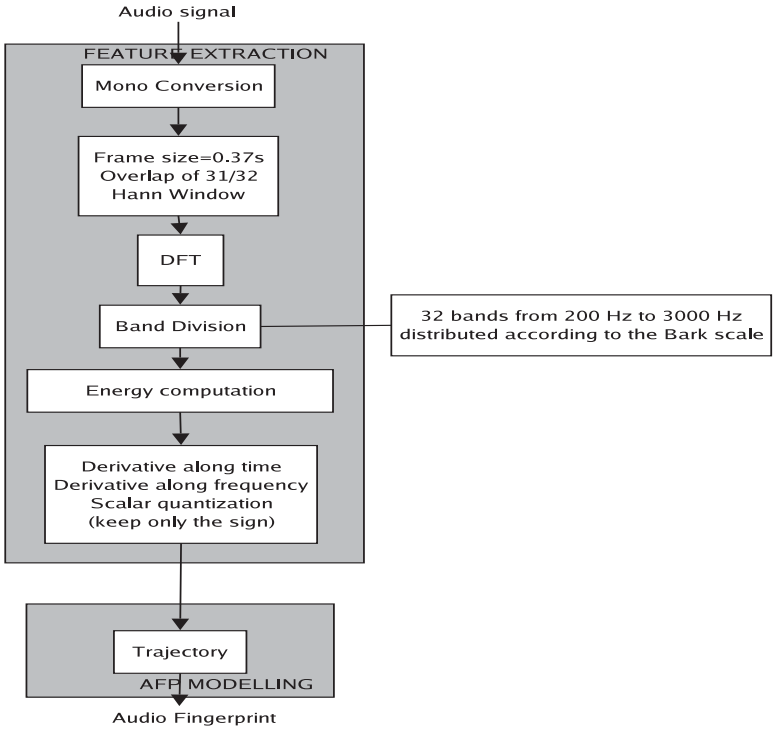


Figure 2.4: Extraction framework of Haitsma-Kalker's AFP

2.5. Music Information Retrieval

Once a fingerprint has been determined from an audio signal, it may be searched in the database in order to retrieve the meta-data associated with it. Figure 2.5 shows a diagram to illustrate the generic process of audio identification. A metric distance, an indexing technique and some criteria for the search are embedded in the searching process.

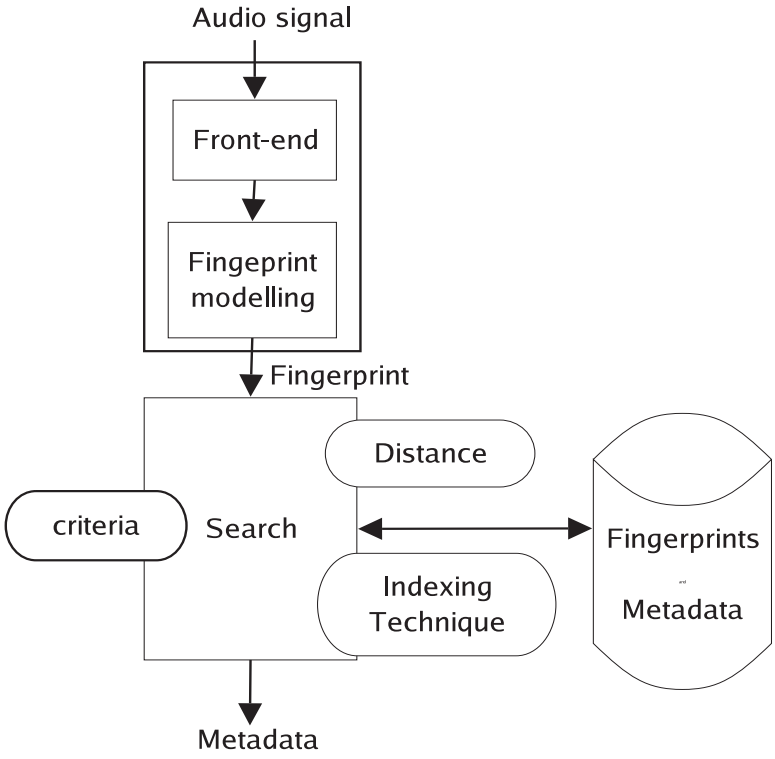


Figure 2.5: Audio Identification framework [Cano02]

2.6. Matching Musical Performances

Aligning performances of classical musical masterpieces is a problem that is commonly addressed as *Polyphonic Audio Matching*. Several applications related to *Music Information Retrieval* rely in robust and efficient solutions to this problem. *Polyphonic Score retrieval* [Pickens02] is one of such applications, the query might be specified by choosing an audio file containing a performance of a musical masterpiece whose score is required, the score is probably linked to another performance of the same masterpiece on the server's database. For example, suppose you need the score of Mozart's *Symphony 41* and you make a query using a performance of this masterpiece by the *London Philharmonic Orchestra*, now suppose this particular score is linked in the server's database to another performance of this masterpiece by the *Berliner Philharmonische Orchester*, then the query server would eventually have to align these two musical performances in order to answer the request. The kind of AFPs that stores how the relevant features of the audio signal evolve in time are what would actually be aligned, more precisely, the AFP of the query's musical performance and those AFPs from the performances included in the database would have to be aligned.

Another application to the Matching performances problem arise in *radio broadcast monitoring* where a performance of some song could be used to monitor occurrences of other renditions of the same song. To enable such application, we will make use of flexible string matching techniques for aligning musical performances, such techniques would allow for multiple songs to be monitored simultaneously. Efficient methods such as Kimmo-Navarro's algorithm based on filtering and hierarchical verification [Fredriksson04] might be used.

Music is considered to be *monophonic* when it is produced by a single instrument and has at most one note being played at any given time, this restriction enables accurate pitch class estimation techniques. Pairs of pitch and duration values were used in [Kosugi00] for the *Querying by Humming* problem.

Aligning performances of polyphonic music was addressed in [Cano99] for the problem known as *Score-Performance following*, aligning a performance with a score is used for automatic accompaniment and automatic adding of special effects using meta-data included in the score. Energy, Delta Energy, Zero crossings, Fundamental Energy, and Delta Fundamental Energy were chosen as the relevant features of polyphonic audio while Hidden

Markov Models (HMM) were preferred to align a performance and a score in [Cano99].

Polyphonic Audio Matching was also addressed in [Hu03] where *Chromagrams* were extracted from the audio signal and Dynamic Time Warping was used to align musical performances with scores which were also converted to chromagrams. *Chromagrams* are sequences of chroma vectors, where each chroma vector contains 12 elements corresponding to the same number of chromas in an octave (i.e. C, C#, D, D#, etc.). Under this formulation, two tones separated by an integral number of octaves share the same value of chroma. To compute chroma values, the frequencies are assigned to bins that represent pitch classes, then the chroma value is computed simply by taking the arithmetic mean of the magnitudes of all the similarly-classified bins [Bartsch05]. Hu *et al* compared the use of the chromas with the use of pitch histograms, Mel Frequencies Cepstral Coefficients (MFCC) and Normalized MFCC get better results with chromas [Hu03].

Dixon and Widmer claimed pitch recognition to be notoriously unreliable for polyphonic music and so preferred to use a low level spectral representation of the audio data. For the alignment, Dixon and Widmer implemented a linear time Dynamic Time Warping (DTW) by estimating the forward path to restrict the search of the optimal warping path [Dixon05b]. The forward path estimation is based on the on-line time warped algorithm developed by Dixon for live tracking performances applications [Dixon05a].

In the *Querying by melody* problem, only a piece of the performance (i.e. a melody) is available. In [Shalev-Shwartz02] the melody is taken as a chorus and searched for.

2.7. Aligning Techniques

To compare musical performances an aligning technique has to be used. The classical approach for aligning time series is the Dynamic Time Warping algorithm. As opposed to the Hidden Markov Model (HMM) approach, DTW requires no training nor choosing a topology (i.e. number of states and how they should be connected), this is specially advantageous since the dictionary (i.e. collection of songs) can be quite dynamic. Flexible String Matching techniques were developed for matching DNA sequences [Gusfield97], finding strings occurrences in texts allowing errors [Navarro02] or finding computer viruses.

As DTW, flexible string matching requires no training, but unlike DTW it easily allows for monitoring occurrences of musical performances in an audio stream (for radio broadcast monitoring) as will be explained below. Methods for the efficient implementation of flexible string matching methods have already been designed either using a finite automata or by using bit parallelism [Navarro02]. Algorithms for Fast Multiple Approximate String Matching have also been developed [Fredriksson04]. Let us first give a short introduction to the classical DTW algorithm.

2.7.1. Dynamic Time Warping (DTW)

The task of aligning a couple of musical performances $R(n)$ and $T(m)$ of lengths N and M respectively is equivalent to the common task with time series of comparing one sequence with another. Figure 2.6 shows two time series that are being compared. In order to establish a similarity measure between them, the time axis of a sequence (or both) needs to be "warped". Dynamic time warping (DTW), is the classic technique for accomplishing this.

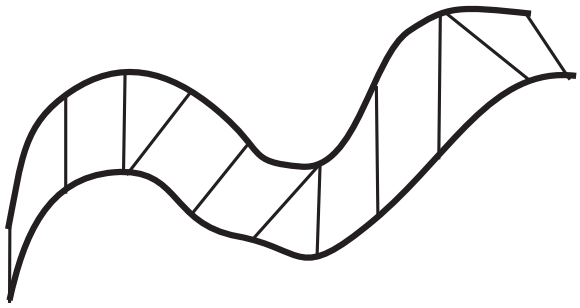


Figure 2.6: Aligning Time Series

At aligning musical performances $R(n)$ and $T(m)$, DTW solves the problem of finding a warping function $m = w(n)$ that maps indices n and m so that a time registration between the time series is obtained. Function w is subject to the boundary conditions $w(0) = 0$ and $w(N) = M$ and it is also subject to local restrictions. An example of such local restriction says that if the optimal warping function goes through point (n, m) it must go through either $(n-1, m-1)$, $(n, m-1)$ or $(n-1, m)$, this particular restriction is depicted in Figure 2.7. A penalization of 2 is charged when choosing $(n-1, m-1)$, a penalization

of 1 if either $(n, m-1)$ or $(n-1, m)$ is chosen. This way, the three possible paths from $(n-1, m-1)$ to (n, m) (i.e. first to $(n, m-1)$ and then (n, m)) will all have the same cost of 2. Other local restrictions defined by Sakoe and Chiba [Sakoe78] can be used.

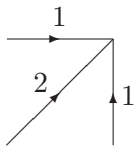


Figure 2.7: Symmetric local restriction of first order

Let $d_{R(n),T(m)}$ be the distance between frame n of performance R and frame m of performance T, then the optimal warping function between R and T is defined by the minimum accumulated distance $D_{n,m}$ as in Equation (2.6).

$$D_{n,m} = \sum_{p=1}^{n} d_{R(p),T(w(p))} \tag{2.6}$$

The DTW distance between performance R of size N and performance T of size M is $D_{N,M}$ and it may be computed using the recurrence defined in Equations (2.7), (2.8), (2.9) and (2.10). This recurrence use the local restriction shown in Figure 2.7. Based on this recurrence $D_{N,M}$ may be obtained using dynamic programming for that purpose.

$$D_{0,0} = 0 \tag{2.7}$$

$$D_{i,0} = \sum_{k=0}^{i} d_{R(k),T(0)} \tag{2.8}$$

$$D_{0,j} = \sum_{k=0}^{j} d_{R(0),T(k)} \tag{2.9}$$

$$D_{i,j} = min \begin{cases} D_{i-1,j-1} + 2d_{R(i),T(j)} \\ D_{i-1,j} + d_{R(i),T(j)} \\ D_{i,j-1} + d_{R(i),T(j)} \end{cases} \tag{2.10}$$

Figure 2.8 shows the warping path as found by DTW corresponding to the alignment of the time series shown on Figure 2.6.

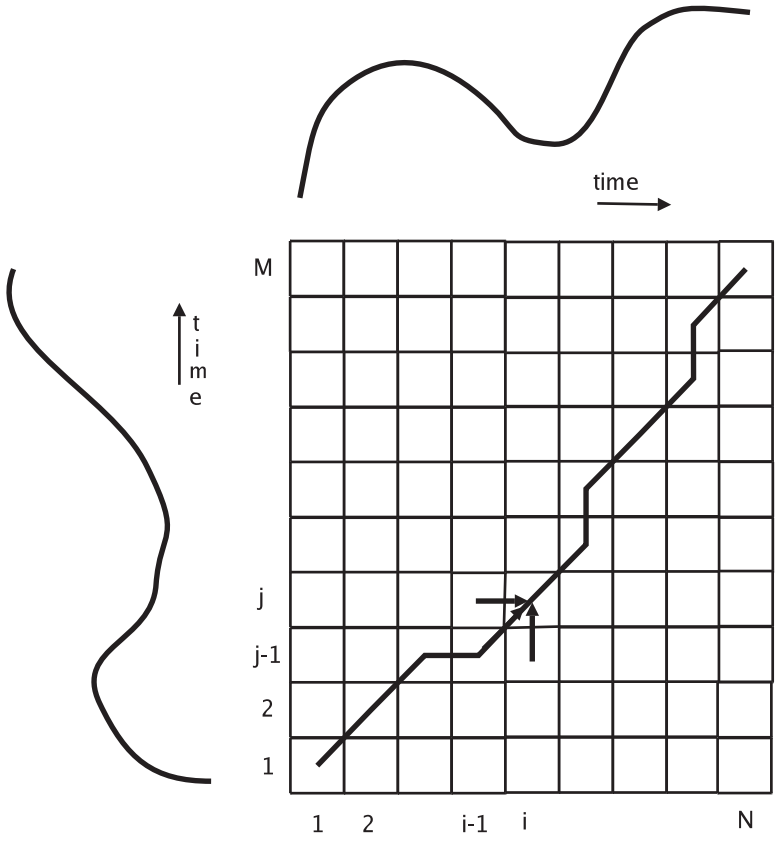


Figure 2.8: The optimal warping path

We will now give a short introduction to flexible string matching techniques, beginning with the most general string edit distance also called the *Levenshtein distance*.

2.7.2. Levenshtein Distance

The string edit distance between two strings is defined as the number of operations needed to convert one of them into the other. The considered operations are *insertions*, *deletions* and *substitutions*. For some specific problems *transpositions* are also considered as valid operations. However, we do not believe they are useful in the problem of matching musical performances. A different cost may be assigned to each operation. If only substitutions are allowed and the cost is 1.0, the distance becomes the *Hamming distance*. If only insertions and deletions are allowed and the cost of both operations is 1.0, the *Longest common subsequence* (LCS) distance is obtained [Navarro97]. Finally if only insertions are allowed at the cost of 1.0, the asymmetric Episode distance is obtained [Navarro97]. The episode distance is used when all the symbols are expected to be found in both strings, in such case the episode distance $E(x, y)$ is simply $|y| - |x|$, otherwise it is defined to be ∞. We will not use this distance due to its lack of symmetry.

To compute the Levenshtein distance L between string t of length N and string p of length M, the recurrence defined by Equations (2.11),(2.12) and (2.13) is used. Note that the same cost of 1.0 has been assigned to all considered operations (insertion,deletions and substitutions).

$$L_{i,0} = i \qquad \forall \quad i = 0..N-1 \tag{2.11}$$

$$L_{0,j} = j \qquad \forall \quad j = 0..M-1 \tag{2.12}$$

$$L_{i,j} = \begin{cases} L_{i-1,j-1} & t_i = p_j \\ min[L_{i-1,j-1}, L_{i,j-1}, L_{i-1,j}] + 1 & t_i \neq p_j \end{cases} \quad j = 1..M-1 \; ; \; i = 1..N-1 \tag{2.13}$$

The classical approach for computing the Levenshtein distance relies on Dynamic Programming. For example, to compute the Levenshtein distance between String “hello”

and String "yellow" of lengths 5 and 6 respectively the dynamic programming matrix L of size 6×7 has to be filled up as follows:

Row zero and Column zero of Matrix L are first filled up with incremental sequences according to Equations (2.11) and (2.12). Proceeding to $L_{1,1}$, since the first character of String "hello" (i.e. character 'h') is different from the first character of String "yellow" (i.e. character 'y'), then $L_{1,1}$ will hold minimum among $[L_{0,0}, L_{1,0}, L_{0,1}]$ plus 1 (i.e. the cost of a edit operation) as in Equation (2.14).

$$L_{1,1} = min[L_{0,0}, L_{1,0}, L_{0,1}] + 1 = min[0, 1, 1] + 1 = 0 + 1 = 1 \tag{2.14}$$

The first character of String "hello" (i.e. character 'h') is different from the second character of String "yellow" (i.e. character 'e'). Therefore $L_{1,2}$ will hold minimum among $[L_{0,1}, L_{1,1}, L_{0,2}]$ plus 1 as in Equation (2.15).

$$L_{1,2} = min[L_{0,1}, L_{1,1}, L_{0,2}] + 1 = min[1, 1, 2] + 1 = 1 + 1 = 2 \tag{2.15}$$

The second character of String "hello" (i.e. character 'e') is different from the first character of String "yellow" (i.e. character 'y'). Therefore $L_{2,1} = min[L_{1,0}, L_{2,0}, L_{1,1}] + 1$ or $L_{2,1} = min[1, 2, 1] + 1 = 1 + 1 = 2$.

The second character of String "hello" (i.e. character 'e') is the same as the second character of String "yellow" (i.e. character 'e'). Therefore in Location (2,2) the accumulated edit distance stored in Location (1,1) will be copied $L_{2,2} = L_{1,1} = 1$. The rest of Matrix L has to be filled up, and when this process is finished, Location $(5, 6)$ of matrix L will hold the accumulated distance of 2 (See Matrix (2.16)). In conclusion, two edit operations are needed to transform String "yellow" into String "hello". For example, replace "h" in "hello" by a "y" and add a "w" at the end.

		y	e	l	l	o	w
	0	1	2	3	4	5	6
h	1	**1**	2	3	4	5	6
e	2	2	**1**	2	3	4	5
l	3	3	2	**1**	2	3	4
l	4	4	3	2	**1**	2	3
o	5	5	4	3	2	**1**	**2**

(2.16)

There is no need for keeping the whole dynamic programming matrix in memory. The Levenshtein distance may be computed using a single column for that matter. To compute the Levenshtein distance between string t of length N and string p of length M using column L, initialize it as $L_i^0 = i$ for $i = 0..N-1$ and then update it using Equation (2.17).

$$L_i^j = \begin{cases} L_{i-1}^{j-1} & \forall\, t_i = p_j \\ min[L_{i-1}^{j-1}, L_{i-1}^{j}, L_i^{j-1}] + 1 & \forall\, t_i \neq p_j \end{cases} \qquad j = 1..M-1\ ;\ i = 0..N-1 \tag{2.17}$$

where L^j stands for L once j characters of string p have been read. The Levenshtein distance between String t and String p will be in L_{N-1}^M.

For the purpose of finding occurrences of a string in a text we must allow a match to occur at any time. To achieve this, first set $L_0 = 0$ and then check the last element of L, if it is less than or equal to the maximum number of errors allowed then a match has occurred. For example, the DNA String ATT was found at positions 2, 3, and 5 with one error inside the DNA sequence $ATCATT$. See how L changes as characters of the DNA sequence $ATCATT$ are read in (2.18). In effect, sub-strings AT found at position 2, ATC found at position 3, and AT found at position 5 have all Levenhtein distances of 1 with String ATT. String ATT was also found at position 6 without errors since a Levenshtein distance of zero appeared in the last element of L after reading the sixth element of the DNA sequence $ATCATT$.

pos			1		2		3		4		5		6
			A		T		C		A		T		T
	0	→	0	→	0	→	0	→	0	→	0	→	0
A	1		0		1		1		0		1		1
T	2		1		0		2		1		0		1
T	3		2		**1**		**1**		2		**1**		**0**

(2.18)

2.7.3. The LCS Distance

The computation of the LCS distance is done just as the computation of the Levenshtein distance, except that only insertions and deletions are allowed. The LCS distance works better than the Levenshtein distance where the same sequence of symbols are present in both strings as is usually the case with speech signals or musical performances. In this context, a symbol represents an acoustic event. For example, the Levenshtein distance cannot tell that the strings *computer* and *curtain* are any more different than the strings *computer* and *cooommpuuuteeer*. However, the LCS distance does report that the distance between the first couple of strings is greater than the distance between the second couple of strings. See Table 2.3.

Table 2.3: Comparison between Levenshtein and LCS distance metrics. The first row are the distances between strings that contain the same sequence of symbols, the second row are distances between strings that do not

x	y	$Levenshtein(x, y)$	$LCS(x, y)$
computer	cooommpuuuteeer	7	7
computer	curtain	7	10

The recurrence defined in Equations (2.19),(2.20) and (2.21) is used with dynamic programming to compute the LCS distance. Of course, LCS can also be computed using a single vector.

$$C_{i,0} = i \qquad \forall \quad i = 0..N-1 \tag{2.19}$$

$$C_{0,j} = j \qquad \forall \quad j = 0..M-1 \tag{2.20}$$

$$C_{i,j} = \begin{cases} C_{i-1,j-1} & t_i = p_j \\ min[C_{i,j-1}, C_{i-1,j}] + 1 & t_i \neq p_j \end{cases} \qquad j = 1..M-1 \; ; \; i = 1..N-1 \tag{2.21}$$

2.7.4. Time Warped LCS

The Time Warped LCS algorithm (twLCS) was proposed by *AnYuan et al* [Guo04] for dealing with rhythm variations quite common in music. AnYuan defined his similarity measure as in Equation (2.22). A similarity measure reports high values for similar strings. We adapted twLCS as a distance measure, so smaller values correspond to similar strings as in Equations (2.23), (2.24) and (2.25). The twLCS distance may be computed in a single vector as well. Table 2.4 shows how string "456" and string "445566" have a zero twLCS distance as desired while the LCS distance between these strings is 3.

$$f_{i,j} = \begin{cases} 0 & if \; i = 0 \; or \; j = 0 \\ max[f_{i,j-1}, f_{i-1,j}, f_{i-1,j-1}] + 1 & if \; i,j > 0 \; and \; t_i = p_j \\ max[f_{i,j-1}, f_{i-1,j}] & if \; i,j > 0 \; and \; t_i \neq p_j \end{cases} \tag{2.22}$$

$$F_{i,0} = i \qquad \forall \quad i = 0..N-1 \tag{2.23}$$

$$F_{0,j} = j \qquad \forall \quad j = 0..M-1 \tag{2.24}$$

$$F_{i,j} = \begin{cases} min[F_{i-1,j-1}, F_{i,j-1}, F_{i-1,j}] & t_i = p_j \\ min[F_{i,j-1}, F_{i-1,j}] + 1 & t_i \neq p_j \end{cases} \qquad j = 1..M-1 \; ; \; i = 1..N-1 \tag{2.25}$$

Table 2.4: Example of how the twLCS distance deals with rhythm variations

x	y	$LCS(x,y)$	$twLCS(x,y)$
456	445566	3	0

2.8. Conclusions

- *State-of-the-art* works in audio-fingerprinting claim to have solved the problem of finding robust features from the audio-signals. These works however, acknowledge that they do not severely deform the audio signals used in their experiments. For example, in the experiments reported in [Herre01], the songs were contaminated with noise only with a "reasonable SNR of 20-25 dB simulating background noise". In [Haitsma02] the only equalization style used alternates attenuation and amplification of consecutive bands in just 3 dB. At such level of degradation the precision rate of those audio-fingerprint is in effect 100 %. We however use severe degradations such as noise mixing resulting in a Signal to Noise Ratio (SNR) of 3-4 dB, and equalization styles in the range of -20dB to +20dB. It is with severely deformed audio signals that our Multi-Band Spectral Entropy Signature outperforms the state-of-the-arts audio-fingerprints.

- There is not a shared database for benchmark purposes. Every author on audio-fingerprinting report to have used a different collection of songs, they do specify the genre of the music they used (i.e rock, country). Guo, in [Guo04] uses folk music claiming that this kind of music was "monophonic in nature", which was quite convenient for his experiments. However, a Database with monophonic music "may not be interesting in practice" as stated in [Yang02]. We therefore used our own collection of polyphonic music of almost all genres.

- The aligning techniques that are best known in the field of *Approximate String Matching* are more convenient for *Music Information Retrieval* purposes since a variety of indexes may be designed taking advantages of efficient algorithms for multiple approximate on-line search using LCS distance or Levenshtein distance. The classic approach of Dynamic Time Warping (DTW) is considered only as a reference for the

results, as opposite to the Hidden Markov Model (HMM) approach, DTW requires no training nor choosing a topology (i.e. number of states and how they should be connected), this is advantageous specially because the dictionary (i.e. collection of songs) can be quite dynamic

Chapter 3

Entropy Signatures

Perhaps someday there will be audio identification systems that work better than the human auditory system. Meanwhile, it would be considered as a great achievement if somehow machines with the ability of identifying sounds similar to that of human beings could be built. The question of How we identify sounds emerges naturally.

We know a human being improves the ability to recognize sounds with the years because the brain learns to expect certain sounds depending on the context. That is why in the presence of noise or when loosing hearing sensibility we still identify the words or at least confuse them with similar ones.

The following text might give us a clue about what the human brain perceives when subject to the problem of identifying sequences:

"Aoccdrnig to a rsecheearr at an Elingsh uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is that frist and lsat ltteer is at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit a porbelm".

The entropy (at least Shannon's entropy) of the scrambled words of the preceding text would be the same if they were not scrambled. Is the information content in a sequence what a brain perceives?. Such a possibility served as a motivation for using entropy as the perceptual feature for audio-identification purposes. The use of Shannon's entropy in computer vision was already aborded by Viola [Viola95] who used mutual information (i.e.

cross-entropy) for identifying images, he found that his method was robust with respect to variations of illumination.

The entropy of the spectrum of an audio signal has been used for the detection of the end-points (begin and end) of utterances in noisy environments [Shen98]. Entropy has also been used for choosing the desirable frame rate in the analysis of speech signals [You04]. In [Bank01] the entropy of an audio-signal was compared with the entropy of the spectrum of the same audio-signal. If the spectral entropy was smaller, then *spectrum coding* was chosen for compressing the audio-signal, otherwise *waveform coding*, a method that works in time domain was preferred.

Anyway, Entropy had never been used as the relevant perceptual feature for building audio-fingerprints. So we decided that the use of entropy for building AFPs would be the central idea of our research. Our first attempt for identifying songs using entropy consisted on transforming the audio-signal entropy into *the Entropy Signal* as we called it. For determining the *entropy signal* we used the first segment of the audio signal with a duration of two seconds (i.e. involving a number of samples that was twice the sampling rate) to build a histogram of the audio-samples. By normalizing the histogram it was converted into a probability density function and then the first entropy value was determined. From then on, for every incoming audio sample, the histogram was updated and a new entropy value was obtained. The *Entropy Signal* was almost the same size as the audio signal except it was one second shorter. We first used the exact position of the global maximum of the entropy signal as the shortest audio-fingerprint ever (i.e. a single number). This technique was proved to be robust to lossy compression when we used it with 50 songs, the position of the global maxima was indeed an invariant. Unfortunately, we soon discovered that it was not unique, at trying it with a hundred songs several collisions occurred. To avoid collisions we used not only the position of the maximum entropy but the position of the k higher peaks int the entropy signal. By increasing k collisions decreased, still collisions occurred.

Encouraged by the result the experiment described above, we decided that the aim of this research should be the use of entropy as the perceptual feature for building robust audio-fingerprints. We will now give a short introduction to the concept of entropy.

3.1. Entropy

The information content of a message is, in Claude Shannon's view, proportional to how much it surprises you when you read it [Shannon49][Shannon49]. Shannon linked irreversible the concept of entropy or disorder in gases as expressed by Boltzman with the information content in a signal by using Boltzman's formula.

Let $v_1, v_2, ..., v_n$ be the possible amplitude values of the samples of an audio signal. If for example, the sample size were 8 bits long, then the audio-samples would be integers in the range [-128,127]. Each v_i has the probability p_i to occur and the whole sequence $p_1, p_2, ..., p_n$ is a discrete Probability Density Function (PDF). Equation (3.1) must hold.

$$\sum_{i=1}^{n} p_i = 1 \tag{3.1}$$

The information content I in a value v_i also called "self information", depends only on its probability p_i to occur and it is denoted $I(p_i)$. The less likely a value is to appear, the more information it brings if it actually occurs. Equivalently, if a value is expected, it delivers very little self information when it arrives. Therefore, the self information is a monotonically decreasing function of the probability. In addition to this, $I(p_i)$ has to be computed in a way that if v_i depends on two or more independent events with probabilities $p_{i_1}, p_{i_2}, ...$, then the contribution to the information content of each event must be summed to be taken into consideration and the result must be $I(p_i)$, so it can be handled as information. Therefore if $p_i = p_{i_1} p_{i_2} ...$ then $I(p_i) = I(p_{i_1}) + I(p_{i_2}) + ...$ The one function to accomplish this is the logarithmic function [Shannon49]. That is why among other reasons that the self information is computed using Equation (3.2). Shannon used base-2 logarithms, however any base may be used for the issue of measuring entropy. The units for entropy are "nats" when the natural logarithm is used and "bits" when the base-2 logarithm is used [Gray90].

$$I(p_i) = ln\Big(\frac{1}{p_i}\Big) = -ln(p_i) \tag{3.2}$$

The entropy H in a sequence is the expected information content in it, so it is the average of all the information contents weighted by their probabilities to occur [Principe00], [Bercher00]. Shannon's entropy H is computed using Equation (3.3) and its continuous

version called "differential entropy" is computed with Equation (3.4).

$$H = E[I(p)] = \sum_{i=1}^{n} p_i I(p) = -\sum_{i=1}^{n} p_i ln(p_i) \tag{3.3}$$

$$H(X) = -\int_{-\infty}^{+\infty} p(x) ln[p(x)] dx \tag{3.4}$$

The entropy of a signal is a measure of how unpredictable it is, if the signal is constant at a fixed value k, then its probability density function (PDF) is a unitary impulse located at k, that is $p_i = \delta(k)$, its entropy or unpredictability would be zero as shown in Equation (3.5). Note that $0log(0)$ needs to be considered zero for this to be true. On the opposite case, if the signal has a uniform distribution then the entropy would be maximum, that is, if $p_i = 1/n$ for n possible values then its entropy would be $ln(n)$ as in Equation (3.6)

$$H_{min} = -\sum_{i} \delta(k) ln[\delta(k)] = -ln(1) = 0 \tag{3.5}$$

$$H_{max} = -\sum_{i=1}^{n} \frac{1}{n} ln(\frac{1}{n}) = -ln(\frac{1}{n}) = ln(n) \tag{3.6}$$

Consider for example that each audio sample was digitalized in words of 16 bits. The maximum entropy would then be 11.09 (i.e. $ln(2^{16})$). Of course in a real audio signal this level of entropy is nearly impossible since it would require that each possible value of the samples appeared the same number of times. For a frame of 2.9721 seconds at a sampling rate of 44,100 samples per second and a sample size of 16 bits, each possible value would have to appear exactly twice.

For two-dimensional data, Shannon's entropy is computed with Equation (3.7). For a uniform probability distribution, the 2D entropy is $2ln(n)$ as shown in Equation (3.8).

$$H = -\sum_{i=1}^{n} \sum_{j=1}^{n} p_{i,j} ln(p_{i,j}) \tag{3.7}$$

$$H_{max} = -\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{n^2} ln(\frac{1}{n^2}) = 2ln(n) \tag{3.8}$$

3.2. The Time-Domain Entropy Signature (TES)

By computing Shannon's entropy from the audio samples of every frame we obtained a sequence of entropy values. We will refer to this sequence as the entropy curve.

3.2.1. On Line Entropy Measurement

Computing the entropy of a signal requires some estimation of the Probability Density Function (PDF). Such estimation may be accomplished using Parametric methods, non parametric methods and histograms. Parametric methods [Bercher00] are advisable when the distribution is known a priori and the amount of data involved is not large. In parametric methods, first a distribution model is chosen and then its parameters are determined [Bercher00]. In non-parametric methods, no assumptions are made about the distribution the PDF belongs to, the PDF is shaped by the data which is in turn smoothed by some kernel. Non-parametric methods require also a large number of samples to be involved to make a good estimation of the PDF. The most popular non-parametric method is the *Parzen window estimation method* [Duda01]. However, non-parametric methods are computationally expensive and so not frequently used for realtime pattern recognition applications. For the on-line determination of the PDF of an audio stream we designed an algorithm based on histograms. The probability p_i for value v_i to be a sample read from the audio stream is computed using Laplace's formula as in Equation (3.9)

$$p_i = \frac{f_i}{N} \tag{3.9}$$

where f_i is the number of times that value v_i occurs in the sequence $x = x_1, x_2, .., x_N$ as in Equation (3.10). N is the frame size.

$$f_i = \sum_{j=1}^{N} \varphi(x_j, v_i) \tag{3.10}$$

where $\varphi(x, y) = 1$ if $x = y$ and $\varphi(x, y) = 0$ otherwise.

The certainty of the histogram method is ensured by the fact that thousands of audio-samples will be used at building the histogram. Furthermore, since the precision of the audio-samples is reduced to only 8 bits, then the resulting histogram is a table with only 256 entries. Algorithm 1 extracts the entropy curve from an audio-signal, this algorithm first builds a histogram for the very first frame of audio. The entropy for the first frame of audio is computed using Equation (3.3). The histogram is updated every time an audio-sample is read from the stream, this is done by increasing the frequency of the last audio-sample read and decreasing the frequency of the audio-sample that gets out of the frame (the oldest sample in the frame). The entropy is also updated for every audio-sample read. To reduce processing time, a lookup table L is used to avoid direct calls to the logarithm function. Six arithmetic operations and two memory access operations are performed for each audio-sample read from the stream. If an audio signal consists of M audio-samples, then approximately $6M$ operations will be performed by Algorithm 1, the complexity of this algorithm is therefore linear.

Algoritmo 1 Algorithm to obtain the Entropy curve

Input:

Audio stream with SampleSize (precision) reduced to 8 bits

Frame size N

Output:

Entropy curve

ExtractTES()

```
1  for i ← 1 to N
2    fifo.write(InputStream.read()) ; Read N samples and save them in buffer fifo
3  for i ← 1 to N
4    L[i] ← −(i/N)ln(i/N) ; Fill up lookup Table L of size N
5  for i = 1 to N
6    val ← fifo.read()
7    Hist[val] ← Hist[val] + 1 ; Build the histogram for the frame of audio saved on the buffer
8    fifo.write(val)
9  H ← 0
10 for i = 1 to 256
11   H ← H + L[Hist[i]] ; Compute entropy for the first frame of audio
12 OutputStream.write(H) ; Send H to the output stream
13 while There are more audio samples
14   SampleIn ← InputStream.read() ; Read one sample from the audio stream
15   fifo.write(SampleIn) ; Add it to the FIFO buffer
16   H ← H − L[Hist[SampleIn]] − L[Hist[SampleOut]] ; Subtract the old information
17   Hist[SampleIn] ← Hist[SampleIn] + 1 ; Update the histogram
18   Hist[SampleOut] ← Hist[SampleOut] − 1
19   H ← H + L[Hist[SampleIn]] + L[Hist[SampleOut]] ; Add the new information
20   if Number of samples read is multiple of N/2
21     then OutputStream.write(H) ; Send H to the output stream
22
23 return
```

The *Entropy curves* of several degraded versions of the song *Diosa del cobre* [1] are shown in Figure 3.1. Please note how similar the *entropy curves* look between the original, the lossy compressed version (i.e. mp3@32kbps), the low-pass filtered version (i.e. 1KHz cutoff) and the scaled version (i.e. 50 percent louder). The profile of these four *entropy curves* is almost identical, therefore we can safely use the sign of the derivative to build a binary string that we call the *Time-domain Entropy Signature (TES)*.

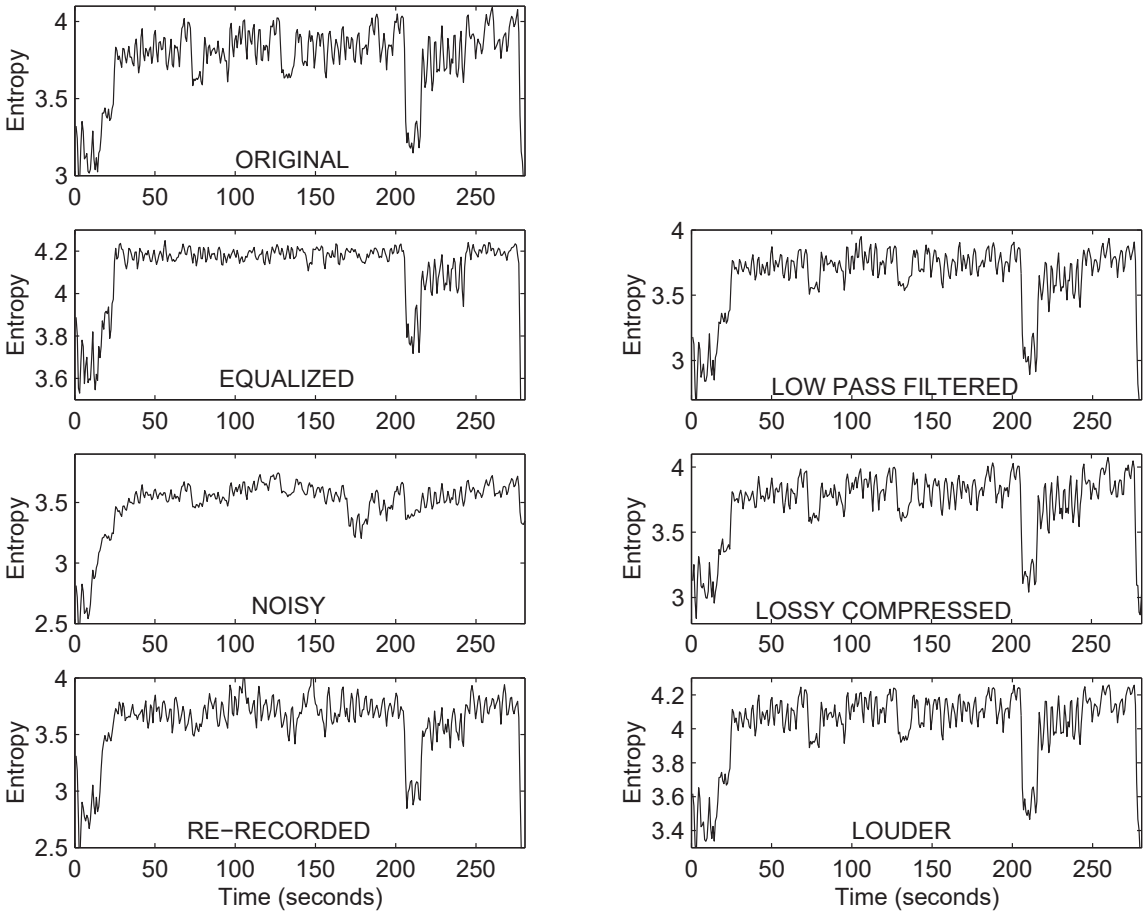


Figure 3.1: Entropy curves of several degraded versions of the song *Diosa del cobre*

[1] Author: Pavel Uriquiza, Perform: Miguel Bosé and Ana Torroja, Album: Girados en concierto, year: 2000

3.2.2. Codification

For every equally spaced interval of one second of the audio-signal two steps are performed:

1. Compute Shannon's entropy using a histogram of the audio-samples directly in time domain. Algorithm 1 is recommended for on-line applications and for efficiently computing entropy.

2. If the entropy of the current segment of audio is greater than the entropy of the previous segment of audio, set bit I to "1" otherwise, clear it (i.e. set I to "0"). Append bit I to the signature of the audio-signal (i.e. TES).

To keep only the sign of the entropy's time derivative makes the signature extremely compact. This way, instead of having 239 floating point values for a 4 minute song, only 30 bytes are needed. Taking the derivative and binary coding it is a key aspect of the technique because we want to compare the profile of the entropy curve of an audio-signal with the profile of the entropy curve of another audio-signal independently of the dynamic ranges of these entropy curves.

As reported in [Ibarrola06], TES is not only extremely compact and easy to compute but turned out to be very robust for the specific degradations of low-pass filtering, scaling and lossy compression. On the other hand, the *entropy curve* is severely deformed when the song is degraded by equalization, noise mixing and re-recording (i.e. Loudspeakers to microphone transmission in a noisy environment).

3.3. The Multi-Band Spectral Entropy Signature (MBSES)

The information content in the signal should be measured in the perspective of the human ear. The Bark scale defines 25 *critical bands*, each one of them of exactly one bark of bandwidth and each one of them corresponds to a section of the cochlea of about 1.3 mm [Zwicker90]. We decided to discard the 25th critical band which corresponds to the frequencies between 15.5 to 20 KHz since only the youngest and healthiest ears are able to perceive. After all, not only the young people are able to recognize a song. If the entropy

of the spectral coefficients corresponding to critical band k is computed for every frame of an audio signal, we obtain a sequence of entropy values. Let this sequence be denoted as $SE_k(t)$ for $t = 0, .., N-1$ where N is the number of frames in the audio-signal. If we plot $SE_k(t)$ we obtain the Spectral Entropy curve for the critical band k or simply the SE_k curve.

Remember from the preceding section how equalization deformed the *entropy curve* making TES practically unsuitable for this kind of degradation. The SE curves do not suffer such deformations where the audio signal is subject to equalization, to show this effect, Figure 3.2 shows the curves SE_4, SE_8, SE_{12}, SE_{16} and SE_{20} of the song *Diosa del cobre*. The curves at the left on Figure 3.2 correspond to the original song while the curves at the right correspond to the equalized version. Amazingly, the SE curves seem almost unaffected by equalization. Not all 24 critical bands are shown so the figure is not geometrically overcrowded, however, the other bands behave similarly. This early experiment was quite encouraging for the design of an audio-fingerprint based on Multi-Band Spectral Entropy. The Multi-Band Spectral Entropy Signature (MBSES).

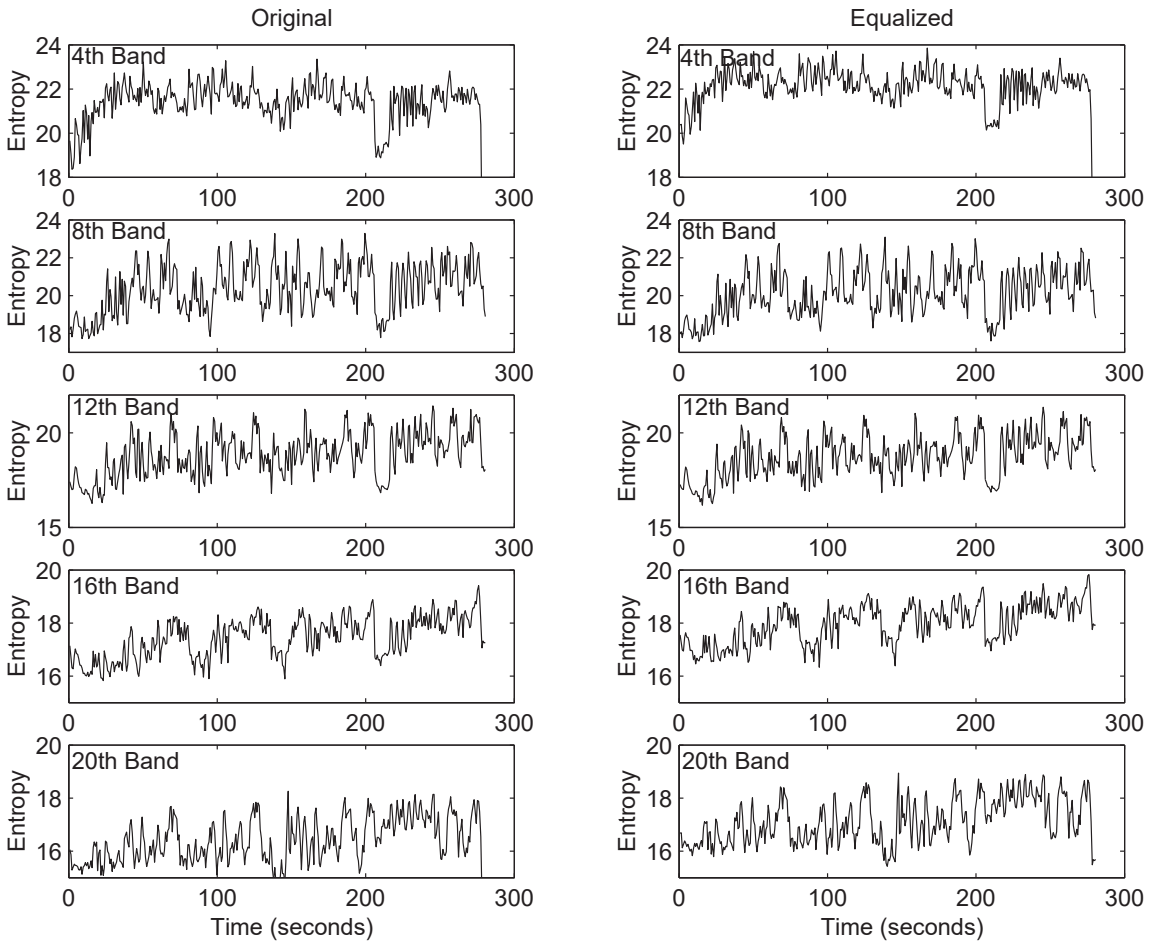


Figure 3.2: Curves SE_4, SE_8, SE_{12}, SE_{16} and SE_{20} of the song *Diosa del cobre* (from the top downwards) according to the Bark scale. Not all 24 critical bands are shown so the figure is not overcrowded: Left, original. Right, Equalized version

3.3.1. Entropygram Determination

For each frame of the audio-signal a vector with 24 entropy values is obtained. The sequence of vectors corresponding to a short excerpt of audio of a few seconds make a matrix of 24 rows and a number of columns that depends on the duration of the excerpt. Such a matrix may be shown as an image where the horizontal axis represents time, the vertical axis represents frequency and the gray levels represent the level of entropy for every band and frame. We call these images *Entropygrams*. In Figure 3.3 an entropygram determined from a piece of audio of 5 seconds is shown.

The firsts steps for the determination of the MBSES of a song are related to the

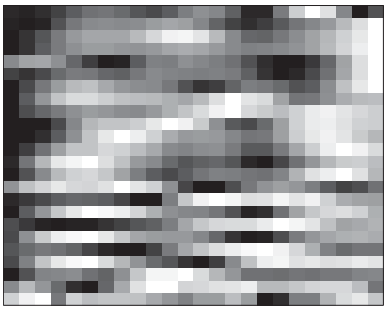


Figure 3.3: Entropygram of a piece of audio of 5 seconds

determination of the $SE_k(t)$ values, these steps correspond to the first blocks depicted in Figure 3.4 and are explained further below:

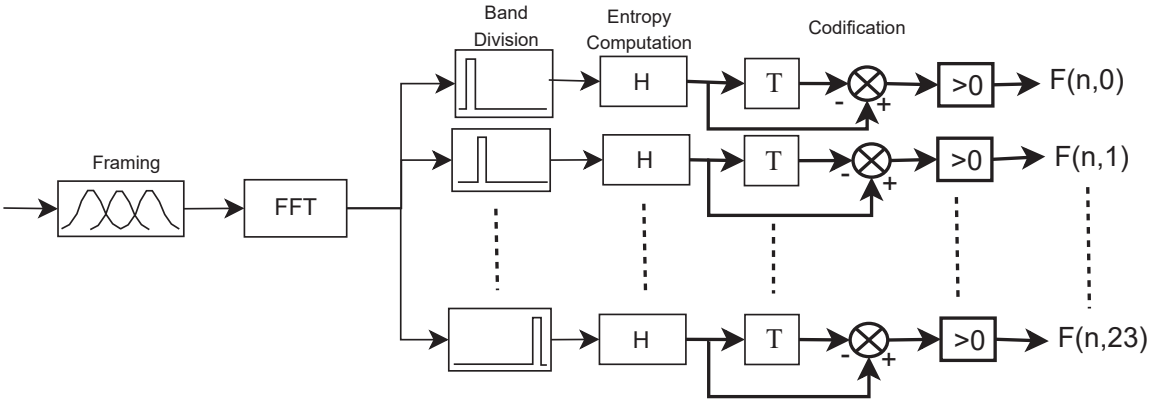


Figure 3.4: Signal processing for the determination of the MBSES of an audio signal

1. Stereo audio signals are first converted to monoaural by averaging both channels.

2. The signal is processed in frames of 370 ms, this frame size ensures an adequate time support for entropy computation according to our experiments. The frame sizes normally used in audio-fingerprinting ranges from 10 ms to 500 ms according to [Cano02]. The frame size used in [Haitsma02] is precisely 370 ms.

3. Our frames are overlapped fifty percent, therefore, 5.4 frames per second will be the frame rate for the MBSES extraction. A low frame rate like this will result in a compact audio-fingerprint.

4. To each frame the Hann window is applied and then its DFT is determined.

5. Shannon's entropy is computed for the first 24 critical bands according to the Bark scale, discarding only the 25th critical band (frequencies between 15.5 KHz and 20 KHz).

The use of histograms requires the use of a high amount of data to obtain a faithful estimation of the PDF. There are however very few spectral coefficients in the first critical bands (i.e. According to the Bark scale) of the spectrum. At 44,100 samples per second, a frame of 0.37 seconds is made out of 16,384 samples. The frequencies ranging from 0 to 22,050 Hz are then spread among 8,192 spectral coefficients. Therefore, the first critical band (i.e. 0-100 Hz) is conformed by only 37 coefficients, the same holds for next band (i.e. 100-200 Hz). See Table 3.1 for the other critical bands.

With as few spectral coefficients as 37, we decided to use a parametric method to estimate the PDF. The real and imaginary parts of the DFT of an audio signal can both be modeled as independent gaussian random variables according to [Martin01]. We will assume that the spectral coefficients corresponding to every critical band also follow a gaussian distribution. Appendix C shows how assuming the spectral coefficients of a single band being gaussian is not very far from reality, however, since using histograms and assuming gaussianity produce non identical results, the PDF estimation method used when building the fingerprint database must be the same method used for the extraction of the fingerprint at querying the database.

When a signal follows a gaussian distribution with mean zero and variance σ^2, then Equation (3.11) holds.

$$p(x) = \frac{e^{-x^2/2\sigma^2}}{\sqrt{2\pi}\ \sigma} \tag{3.11}$$

Replacing $p(x)$ into Equation (3.4), the formula for determining Shannon's entropy of a random variable with a gaussian distribution is deduced in Equation (3.12):

Table 3.1: Number of spectral coefficients available per critical band for PDF estimation purposes. The first critical bands are conformed by very few spectral coefficients according to the Bark scale. Audio signal sampled at 44,100 Hz in a frame of 0.37 sec

Bark	Hz	Initial Coefficient	Spectral Coefficients
1	0-100	2	37
2	100-200	39	37
3	200-300	76	37
4	300-400	113	37
5	400-510	150	41
6	510-630	191	45
7	630-770	236	52
8	770-920	288	55
9	920-1080	343	66
10	1080-1270	409	70
11	1270-1480	479	78
12	1480-1720	556	90
13	1720-2000	646	105
14	2000-2320	751	112
15	2320-2700	863	148
16	2700-3150	1011	161
17	3150-3700	1172	210
18	3700-4400	1382	260
19	4400-5300	1642	335
20	5300-6400	1977	408
21	6400-7700	2385	509
22	7700-9500	2894	643
23	9500-12000	3537	923
24	12000-15500	4460	1300

$$
\begin{aligned}
H &= -\int_{-\infty}^{\infty} \frac{e^{-x^2/2\sigma^2}}{\sqrt{2\pi}\ \sigma} ln\left[\frac{e^{-x^2/2\sigma^2}}{\sqrt{2\pi}\ \sigma}\right] dx \\
&= \frac{ln(\sqrt{2\pi}\ \sigma)}{\sqrt{2\pi}\ \sigma}\int_{-\infty}^{\infty} e^{-x^2/2\sigma^2} dx + \\
&+ \frac{1}{\sqrt{2\pi}\ 2\sigma^3}\int_{-\infty}^{\infty} x^2 e^{-x^2/2\sigma^2} dx \\
&= \frac{ln(\sqrt{2\pi}\ \sigma)}{\sqrt{2\pi}\ \sigma}\sqrt{2\sigma^2\pi} + \frac{4\sqrt{\pi}(\sqrt{2}\sigma/2)^3}{\sqrt{2\pi}\ 2\sigma^3} \\
&= \frac{1}{2}ln(2\pi) + ln(\sigma) + \frac{1}{2} \\
&= \frac{1}{2}ln(2\pi e) + \frac{1}{2}ln(\sigma^2)
\end{aligned} \tag{3.12}
$$

Similarly, for the n-dimensional case it is not difficult to prove that the entropy of a random variable with Normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{R})$ with zero mean and co-variance matrix $\mathbf{R}$ of size $n \times n$ is computed with Equation (3.13) [Mohammad-Djafari94].

$$H = \frac{n}{2} ln(2\pi e) + \frac{1}{2} ln[\det(\mathbf{R})] \tag{3.13}$$

We then compute entropy using Equation (3.13) with $n = 2$ as in Equation (3.14).

$$H = ln(2\pi e) + \frac{1}{2} ln(\sigma_{xx}\sigma_{yy} - \sigma_{xy}^2) \tag{3.14}$$

where σ_{xx} and σ_{yy} also known as σ_x^2 and σ_y^2 are the variances of the real and the imaginary part respectively and $\sigma_{xy} = \sigma_{yx}$ is the covariance between the real and the imaginary part of the spectrum in its rectangular form and so $\sigma_{xy}\sigma_{yx} = \sigma_{xy}^2$.

3.3.2. The codification step

It was shown on Figure 3.2 how the SE_k for any band k was practically not deformed when a song was equalized, the profile of the curve remained almost unchanged. There was however a change in the dynamic range due to a general decrease of entropy in attenuated bands. Based on this observation, it seems to be a good strategy to compare not the amplitudes but the slopes of the SE_k curves for each band k, better yet would be to compare only the signs of the slopes. The codification step consists of storing for each band only an indication of whether the spectral entropy is increasing or not in the current frame. Equation (3.15) states how the bit corresponding to band b and frame n of MBSES is determined using the entropy values of frames n and $n-1$ for band b. Only 3 bytes (i.e. 24 bits) are needed for each frame of audio signal, that was another reason for dropping the 25th critical band.

$$F(n,b) = \begin{cases} 1 & if \ \ [h_b(n) - h_b(n-1)] > 0 \\ 0 & Otherwise \end{cases} \tag{3.15}$$

Since the MBSES of a song is a binary matrix, it can be shown as a black and white image as the one shown in Figure 3.5 where a piece of the MBSES from the song *Diosa del cobre* is shown. In the same figure a 5 second excerpt is magnified.

Figure 3.5: Fragment of the MBSES of the song *Diosa del cobre*. A 5 sec excerpt is magnified

3.3.3. Time complexity for the MBSES extraction procedure

Assume that an audio-signal is formed by M audio-samples, extracting MBSES from this audio-signal requires the determination of the Fast Fourier Transform (FFT) of $2M/N$ frames of length N (Remember that frames are overlapped 50 %). Determining the FFT of a frame with N samples has a complexity $O(Nlog(N))$ [Stanley84]. Therefore, the complexity of the MBSES extraction procedure would be linear on M. After computing the means, variances and entropies from the 24 subsets of spectral coefficients, the MBSES extraction procedure complexity is $O(M)$.

3.4. Polyphonic Audio Matching

In Figure 3.6, the entropygrams of two performances of Tchaikovsky's *Nutcracker waltz of the Flower* look just alike. Encouraged by this observation we decided to include some experiments on the use of MBSES in the problem of matching musical performances. Figure 3.7 shows the MBSES of two performances of Mozart's Serenade number 13 Allegro.

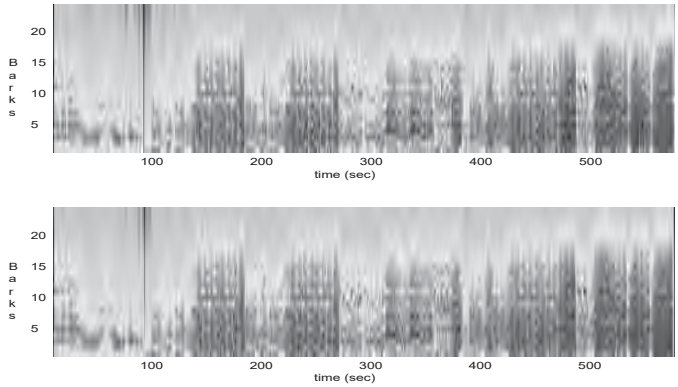


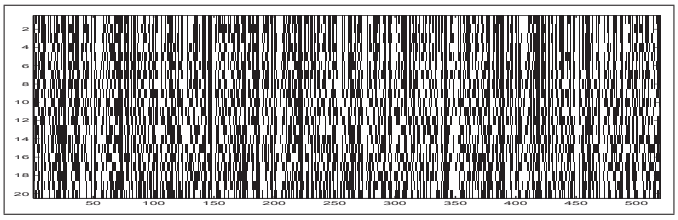
Figure 3.6: Entropygrams of the two performances of Tchaikovsky's *Nutcracker waltz of the flower*

3.5. Conclusions

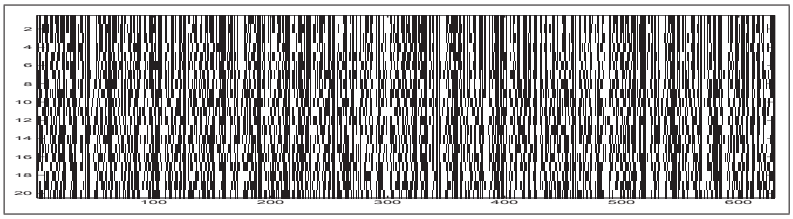
Since Time-domain Entropy Signature (TES) is conformed by a single bit per frame, then it is an extremely compact audio-fingerprint, it is also computed with very little computations since it is determined directly in time domain. Although TES seems to be robust by low-pass filtration, lossy compression, re-recording and scaling, it does not seem very robust by noise mixing and it does not seem robust to equalization at all. This is suspected by observing the entropy curves plotted in Figure 3.1 and it is expected to be confirmed by the experiments described in the next chapter.

The Multi-Band Spectral Entropy Signature (MBSES) is conformed by 3 bytes per frame of audio, Haitsma-Kalker's AFP is conformed by 4 bytes per frame. Since MBSES uses a frame rate of only 5.4 fps (frames per second) while Haitsma-Kalker's AFP works with a frame rate of 86.2 fps then MBSES results in a signature that is far more compact. The AFP of MPEG-7 stores 96 floating point values every second of audio, if the floating point values are stored using 4 bytes, then 3 kbps (kilo bits per second) is the required space for storing the AFP of MPEG-7, 2.6 kbps is the space needed to store Haitsma-Kalker's AFP and only 0.13 kbps to store MBSES. Therefore, even being 24 times bigger than TES, MBSES is still a very compact AFP.

By observing the SE_k curves of Figure 3.2, MBSES is expected to be robust to every degradation, this is will be confirmed by the experiments described in the next chapter.



(a) performed by the *Camerata Academica* conducted by Alfred Scholz



(b) performed by the *Slovak Philharmonic Orchestra*, conducted by Libor Pesek

Figure 3.7: MBSES of two performances of Mozart's *Serenade Number 13 Allegro*

Chapter 4

Experiments

In this chapter we explain the experiments for the evaluation of robustness of MBSES, TES, the AFP of MPEG-7 and Haitsma-Kalker's AFP.

4.1. Introduction

Four experiments were designed: In the first experiment, degraded whole songs were compared against each other searched to verify that the degraded versions of the same song had short distances between them and that different songs had large distances between them. In the second experiment, degraded whole songs are searched in a collection of 4000 MP3 files using the nearest neighbor criterion. In the third experiment, excerpts of only five seconds of the degraded songs were used to search in the collection of 4000 MP3 files. The fourth experiment is about searching in the collection of 4000 songs using other performances of 124 of them. Additionally, some experiments with speech signals are shown in Appendix D.

4.1.1. Degradations of the Audio Signals

For the experiments on robustness the following deformations were considered:

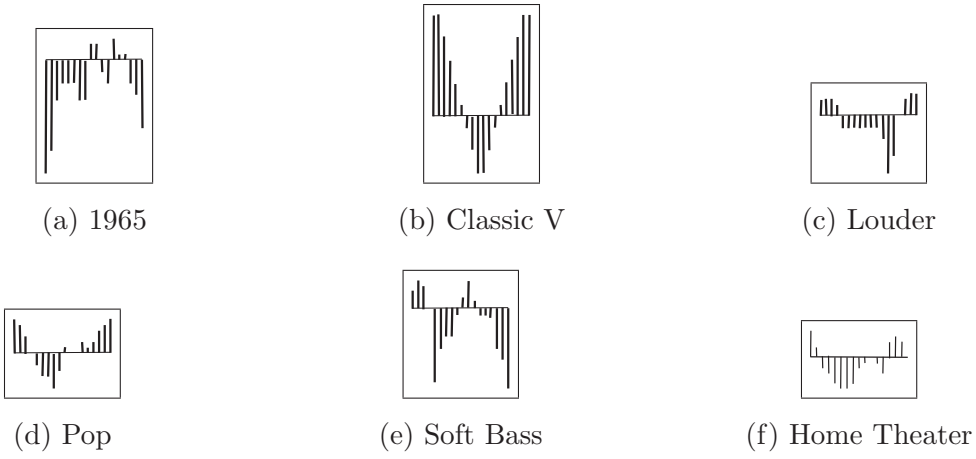
1. Cropping. The songs will be identified using excerpts of 5 seconds.

2. Desynchronization. The 5 seconds excerpt that is used to search a song is randomly

selected. Therefore, the probability that the initial frame of the excerpt begins at exactly the same time than any frame from the original song is practically zero (A real scenario was reproduced). This deformation is also known as *time shifting*.

3. Lossy compression. Compressing songs to MP3 implies a certain degradation of the audio-signal. The implicit degradation of MP3 compression is higher when a low bit rate is selected. In our experiments we used a bit rate of only 32 kbps. MP3 compression system also introduces a certain time shift. The "CD to MP3 freeware" [cd2] was used to generate these files directly from the CDs using the lowest allowed bit rate allowed by this utility which was precisely 32kbps.

4. Equalization. The equalization styles included are common equalization styles from [EQp06]. Figure 4.1 shows the Amplification/Attenuation diagrams corresponding to all the equalization styles used.



(a) 1965 (b) Classic V (c) Louder

(d) Pop (e) Soft Bass (f) Home Theater

Figure 4.1: Equalization styles used. Eighteen bars spread from the lowest band (i.e. leftmost) at 55 Hz to the highest band at 20 KHz. A bar above the horizontal axis indicates the amplification of its corresponding band. A bar below the horizontal axis indicates the attenuation of its corresponding band

To equalize songs, the utility used was xmms [xmm], the default output plugin libALSA.so (ALSA 1.2.10) was changed for libdisk_writer.so (Disk Writer 1.2.10), this library writes the equalized sound to a file instead of sending it to the speakers. The effect plugin libeq.so (EQ plugin 0.5) was used to implement the equalization effect. As

an example, the equalization fashion corresponding to Table 4.1 is specified in the following file "ClassicV.preset":

```
Preamp=0
Band0=20
Band1=10
Band2=0
Band3=-5
Band4=-10
Band5=-5
Band6=0
Band7=5
Band8=10
Band9=20
```

Table 4.1: Parameters for the equalizer

KHz	.06	.17	.31	.6	1	3	6	12	14	16
dB	20	10	0	-5	-10	-5	0	5	10	20

5. Mixing with white noise. This kind of noise contaminates all bands. Once mixed with white noise the songs have a Signal to Noise Ratio (SNR) between 3 and 5 dB. The SNR is determined by using Equation (4.1) where P_{signal} is the power of the original signal and P_{noise} is the power of the noise added to the signal.

$$SNR = 20log_{10}(\frac{P_{signal}}{P_{noise}}) \tag{4.1}$$

Noise produced by big fans fall in the range that is referred as white noise. The noise we used to contaminate songs can be accessed under the name of turbofan-hifi.wav at:

`http://www.asti-usa.com/skinny/sampler.html`.

Please note that colored noise is not as severe as white noise since colored noise affect only some bands of the signal.

To contaminate with noise we made use of the ecasound utility [Eca] specifically the command used was:

```
ecasound -a:1 -i cobre.wav -a:2 turbofan.ewf -eac:200,1 -eac:200,2
-t:279 -a:all -o cobreHiss.wav
```

where turbofan.ewf is the following text file with only two lines:

```
turbofan44100.wav
looping=true
```

Ecasound will take as input the file cobre.wav for chain one, for chain two the input as indicated in file turbofan.ewf will be the file with noise called turbofan44100.wav. The second line of file turbofan.ewf indicates that when the noise signal of turbofan44100.wav reaches an end, it starts over again, this was done because turbofan44100.wav is only 108 seconds long while the song to be contaminated is longer, it lasts 279 seconds. The duration of cobre.wav is specified in seconds with the -t option which causes to finish processing once this time is over (i.e. 279 seconds). Both chains are mixed (-a:all) and the output is sent to the file cobreHiss.wav.

6. Low pass filtering with a cutoff frequency of 1KHz.

   The Sox utility [Sox] was used for low pass filtering the song, used as in the following example where the audio signal in cobre.wav was filtered:

   ```
   sox cobre.wav cobre1000.wav lowpass 1000
   ```

   Sox uses a Butterworth filter with an attenuation of 20 dB/decade. An amplification in dB is defined as:

$$G_{dB} = 20log(V_{out}/V_{in}) \tag{4.2}$$

   Since the amplification is -20dB/decade (a negative amplification in dB is really an attenuation), then every decade the relation between the amplitudes of the output signal V_{out} and the input signal V_{in} would be given by:

$$\begin{aligned} -20 &= 20log(V_{out}/V_{in}) \\ -1 &= log(V_{out}/V_{in}) \\ 10^{-1} &= V_{out}/V_{in} \end{aligned} \tag{4.3}$$

Therefore, for a frequency that is ten times the cutoff frequency (i.e. a decade above) the amplitude of the output signal has declined to a tenth of the amplitude of the input signal.

$$V_{out} = V_{in}/10 \tag{4.4}$$

7. Loudspeaker-Microphone transmission (Ls-Mic). This degradation consisted of playing the music with the pair of loudspeakers of a multimedia system and recapturing it with an omnidirectional microphone with a sensibility of -54±3 dB and a frequency response of 50Hz to 16KHz in a noisy environment. The microphone was placed at a distance of 10 cm from the speakers.

   Again Ecasound was used for this degradation, the way how it was used is shown in the following example:

```
ecasound -b:256 -r -f:16,2,44100 -a:1 -i cobre.wav -o /dev/dsp
-t:279 -a:2 -i /dev/dsp -o cobreLsMic.wav
```

   Ecasound will take as input of the first chain the file cobre.wav and will send it to the speakers, the default output of the sound card (/dev/dsp). In the second chain the input is the microphone, the default input of the sound card (again /dev/dsp). With the -o option, the file cobreLsMic.wav was specified as the output. To minimize synchronization problems, a small buffer size is set to 256 samples with the -b:256 option. To ensure flawless recording, runtime priority is risen with the -r option

8. Scaling. The signal was amplified 50 percent without clipping prevention, in fact approximately 30 percent of the signal's peaks were clipped during this degradation.

   Here again the Sox utility is used as follows:

```
sox cobre.wav cobreLoud.wav vol 1.5
```

4.1.2. Sensitivity Analysis

When two degraded versions of the same song have a distance below some threshold *th* between them, we say that we are in the presence of a *true positive*, if those two degraded

versions of the same song have a distance above th between them, then we are dealing with a *false negative*. On the other hand, when comparing two different songs, if the distance between them falls below th then we call that a *false positive* and if the distance is grater than th, then is a *true negative*. Table 4.2 summarizes these definitions.

Table 4.2: Definitions for the sensitivity analysis

	$dist < th$	$dist > th$
Same songs	True Positive (TP)	False Negative (FN)
Different songs	False Positive (FP)	True Negative (TN)

Consider the following situation: Assume that 40,000 comparisons between songs were performed in an experiment and we know for a fact that 4,000 times a song was actually compared with another degraded version of the same song and the rest (36,000) comparisons were between different songs. Suppose that given a certain threshold th the audio-fingerprinting system computed a distance below th (considered them as a match) for 3900 out of the 4000, and the rest (i.e. 100) were mistakenly considered as being different songs (i.e. distance above th). On the other hand, 35950 out of the 36000 the system correctly considered them as being different and 50 incorrectly considered them to belong to the same class (i.e. song). In Table 4.3 these hypothetic results are summarized.

Table 4.3: Example of positive-negative result for a certain threshold th

	$dist < th$	$dist > th$	Total
Same songs	$TP = 3900$	$FN = 100$	4000 Positives
Different songs	$FP = 50$	$TN = 35950$	36000 Negatives
Totals	3950	36050	40000

The *True Positive Rate* (TPR) is the fraction of songs that the system correctly identifies (i.e. true positives) from all the songs the system should have (i.e. *Positives*). The TPR is also known as *sensitivity* or *recall* and it is estimated with Equation (4.5). TPR equals $1 - FRR$ where FRR is the well known *False Rejection Rate*.

$$TPR = \frac{TP}{TP + FN} \tag{4.5}$$

For the Example described above, $TPR = 3900/(3900 + 100) = 0.975$=97.5 %

The *False Positive Rate* (FPR) is a measure of how often the system mistakes a song for another and it is defined as in Equation (4.6). The FPR is also known as *False Alarm Rate* and equals $1 - specificity$

$$FPR = \frac{FP}{FP + TN} \tag{4.6}$$

For the Example described above, $FPR = 50/(50 + 35950) = 0.00125$=0.125 % and then *specificity*=$1 - 0.00125 = 0.99875$=99.875 %

The Receiver Operating Characteristics (ROC) space is the plane where the vertical axis is the TPR and the horizontal axis is the FPR, a single point in this plane represents the performance of the system for a given threshold. By varying the threshold a ROC curve is generated.

The *precision rate* is defined as the fraction of the correctly identified songs (i.e. true positives) over the number of queries performed (i.e. true positives plus false positives) [Fawcett03].

For the Example described above, *precision rate*=3900/(3900+50)=0.9873=98.73 %

4.1.3. Additional Information concerning the experiments

Some remarks about the experiments:

- The Hamming distance was used for TES, MBSES and Haitsma-Kalker's AFP. The Mahalanobis distance (2.4) was used for the AFPs of MPEG-7.
- Since TES is a signature designed for whole songs since it is extremely small, it was only included in experiments where whole songs were compared.
- Haitsma-Kalker's AFP was excluded from the Experiment 2 where whole songs were used to search the nearest neighbors from a collection of 4,000 songs. Haitsma-Kalker's AFP is too big for using it in such a way, this AFP was designed for searching small excerpts. The same could be said for Experiment 1. However, Experiment 1 implied much fewer comparisons than experiment 3 and so Haitsma-Kalker was not excluded from Experiment 1.

- Segments from several degraded versions of one of the songs of the test set are available at

  `http://lc.fie.umich.mx/~camarena/Audiofiles.html`

4.2. Experiment 1. Comparing Whole Songs

In this experiment, degraded songs were not only compared to original songs, but to other degraded versions as well. For example, the equalized version of a song will be compared with its noisy version. In the problem of querying by example the degraded versions are always compared with originals. However, this is not the case for other applications. For example, in radio broadcast monitoring, the audio signal that is going to be used as the reference for monitoring a specific announcement spot is normally captured in the same way as the audio signal to be monitored. As another example, consider a p2p application that is always looking in the network for audio files with better quality than the ones in the local host, this application would be comparing all kinds of degraded versions including those obtained from old tapes.

Thirty eight songs were used for this experiment, each one in six versions: Original, Equalized, Scaled, Noisy, Recaptured and Filtered (through Low pass filter). Each audio file were compared with every other one to fill up the *confusion matrix* of 228 rows and 228 columns. The 51,984 locations of the confusion matrix correspond to the same number of Hamming distances between AFPs that had to be computed.

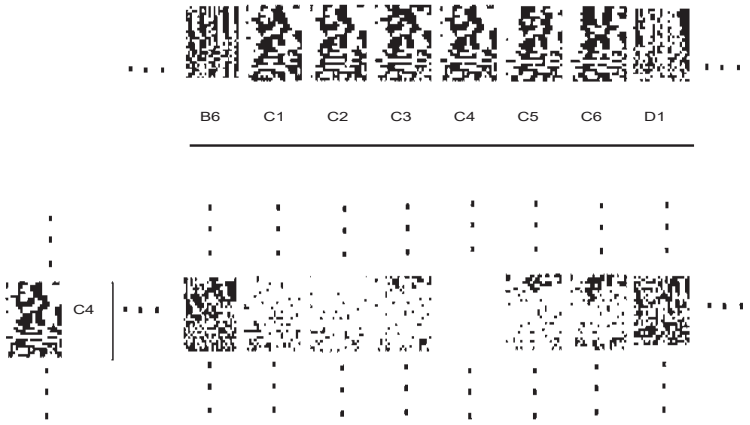


Figure 4.2: A piece of a row of a confusion matrix where audio file C4 (i.e. song C degraded version 4) is compared with 6 versions of song C (including itself) and two audio files from other songs (B and D)

In Figure 4.2, eight comparisons out of the 228 of a single row of the confusion matrix are shown. The degraded version 4 of song C is compared with the six degraded versions of the same song including itself (i.e. C1,..,C6) and two degraded versions of other songs (i.e. B6 and D1). The differences are shown graphically, a fuller image means a higher Hamming distance.

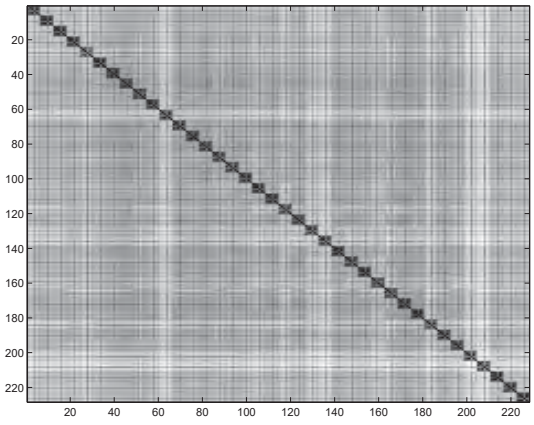


Figure 4.3: Confusion Matrix obtained when MBSES was used in Experiment 1

Figure 4.3 shows the whole confusion matrix that results from Experiment 1 using MBSES. Every pixel of Figure 4.3 has a gray level according to the distance it represents

(darker means closer). The first row of pixels represents the set of distances between the first audio file and every other one. The second row of pixels represents the distances between the second audio file and every one from the rest and so on. The 228 pixels along the diagonal are all black because the distance between any audio file and itself is always zero. The symmetry of the Hamming distance implies a symmetric confusion matrix. The name of every audio-file has a prefix according to the song's name and a suffix that denotes the kind of degradation the song suffered. The audio files are maintained in alphabetical order according to its name, for this reason, the ideal confusion matrix would be all white with 38 black squares along the main diagonal, being each black square of size 6×6 (i.e. six degradations).

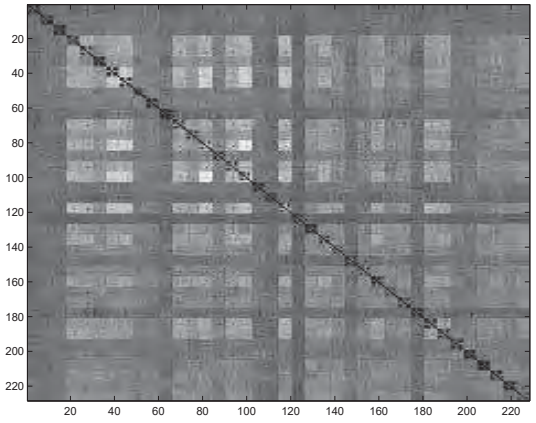


Figure 4.4: Confusion Matrix obtained when TES was used in Experiment 1

The confusion matrices that result from Experiment 1 using TES, Haitsma-Kalker's AFP and the AFP of MPEG-7 and are shown in Figures 4.4, 4.5 and 4.6 respectively. What is evident is that the expected 6×6 black squares along the diagonal are not as well defined in these figures as they are in Figure 4.3. Figure 4.3 even resembles the ideal confusion matrix described above, this fact reveals MBSES as the most robust audio fingerprint when compared with TES, Haitsma-Kalker's AFP and the AFP of MPEG-7.

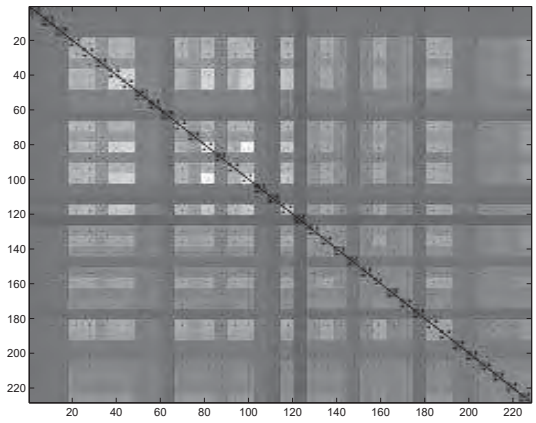


Figure 4.5: Confusion Matrix for Haitsma-Kalker's AFP (Experiment 1)

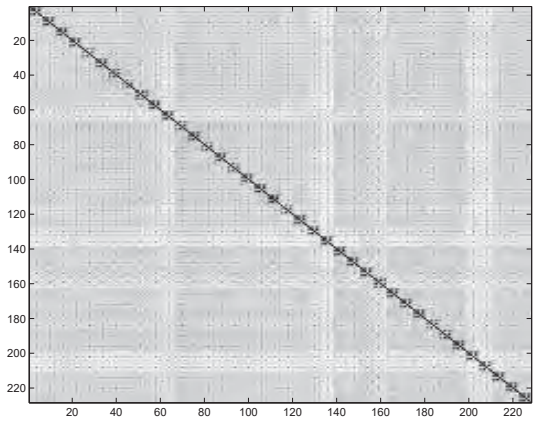


Figure 4.6: Confusion Matrix obtained when the AFP of MPEG-7 was used. Experiment 1

Figure 4.7 shows the ROC curves for AFP systems that use MBSES, TES , Haitsma-Kalker's AFP and the AFP of MPEG-7; there we clearly see that the area under the ROC curve for MBSES is greater than the area under the ROC curve of TES, Haitsma-Kalker's AFP or the AFP of MPEG-7.

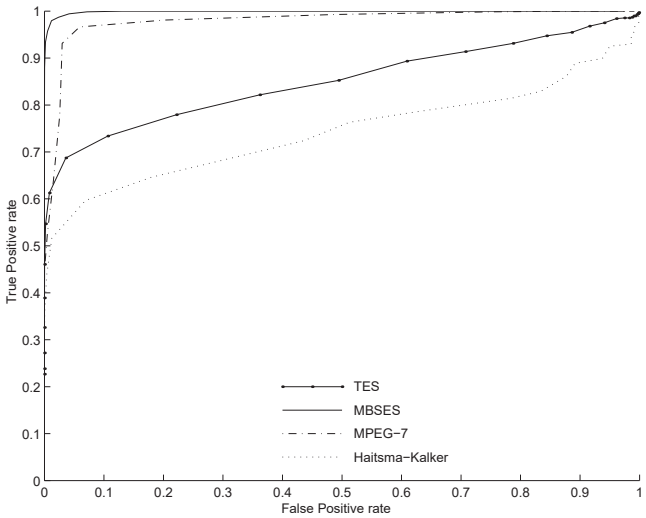


Figure 4.7: ROC curves for Experiment 1

Table 4.4: Precision rates of MBSES for its optimal threshold (Experiment 1)

	LowPass	EQ	Loud	Noisy	LsMic
Original	100 %	100 %	100 %	100 %	100 %
LowPass		100 %	100 %	97 %	100 %
EQ			100 %	97 %	100 %
Loud				100 %	100 %
Noisy					95 %

From the sensitivity analysis that generate the ROC curves, the optimal threshold for each system was also determined, this is the threshold that corresponds to the point of the ROC curve that is closest to the upper-left corner of the ROC plane. Using the optimal threshold, the precision rates for all possible combinations of degradations (e.g low pass filtered against equalized) using MBSES is shown on Table 4.4. Tables 4.5, 4.6 and 4.7 correspond to the precision rates for the AFP of MPEG-7, TES and Haitsma-Kalker's AFP respectively.

Table 4.5: Precision rates of MPEG-7 for its optimal threshold (Experiment 1)

	LowPass	EQ	Loud	Noisy	LsMic
Original	97 %	100 %	100 %	77 %	95 %
LowPass		100 %	100 %	71 %	90 %
EQ			100 %	76 %	87 %
Loud				77 %	95 %
Noisy					74 %

Table 4.6: Precision rates of TES for its optimal threshold (Experiment 1)

	LowPass	EQ	Loud	Noisy	LsMic
Original	87 %	52 %	100 %	74 %	85 %
LowPass		52 %	87 %	61 %	74 %
EQ			42 %	26 %	42 %
Loud				74 %	55 %
Noisy					37 %

Table 4.7: Precision rates of Haitsma-Kalker's AFP for optimal threshold (Experiment 1)

	LowPass	EQ	Loud	Noisy	LsMic
Original	87 %	61 %	100 %	47 %	29 %
LowPass		55 %	87 %	32 %	32 %
EQ			61 %	24 %	21 %
Loud				45 %	29 %
Noisy					18 %

4.3. Experiment 2. Searching in a larger collection

To verify if the system is scalable, that is, if the precision rates fall when dealing with thousands of songs, the following steps were followed:

1. The signatures (i.e. MBSES, TES and the AFP of MPEG-7) of 4 000 songs from all kind of genres (rock, pop, classical, etc.) were extracted.

2. Four hundred songs (i.e. ten percent) were subject to the six signal degradations numbered (3) to (8) at the beginning of subsection 4.1.1

3. The signatures of the 2,400 audio files, obtained in the previous step, were also extracted.

4. The signatures of the degraded songs were searched in the collection of 4 000 using the nearest neighbor criterion (i.e. that with the smallest distance).

Table 4.8 shows the precision rates for TES, MBSES and the AFP of MPEG-7 that resulted from this experiment. In this experiment MBSES showed high robustness to every considered degradation. The AFP of MPEG-7 showed high robustness to equalization, lossy compression an scaling. Finally, TES showed high robustness to Low-Pass filtering, lossy compression and scaling, its robustness to re-recording is also acceptable.

Table 4.8: Precision rate for different signal degradations using TES, MBSES, and the AFP of MPEG-7 without cropping (whole songs).

Degradation	TES	MBSES	MPEG-7
Equalization	53.7 %	100.0 %	100.0 %
Noise contamination (SNR=3.4dB)	63.2 %	100.0 %	55.3 %
Re-recording (LsMic)	92.1 %	100.0 %	80.0 %
Low-Pass filtering (1KHz)	100.0 %	100.0 %	72.1 %
Lossy Compression (32kbps)	100.0 %	100.0 %	100.0 %
Scaling (50 percent louder)	100.0 %	100.0 %	100.0 %

4.4. Experiment 3. Searching With Small Excerpts

Experiment 2 was done with whole songs, however, for some applications it is important to identify a song using only a small excerpt of it. To verify the robustness to the degradations considered in experiment 2, combined with cropping and desynchronization at the same time, the following steps were followed:

Figure 4.8: The piece of MBSES that was magnified in Figure 3.5 but in the same size as the used in Figures 4.10 and 4.11

1. The signatures (i.e. MBSES, Haitsma-Kalkers's AFP and the AFP of MPEG-7) of 4,000 songs from all kinds of genres (rock, pop, tropical, classical, etc.) were extracted and stored.

2. Four hundred of these songs were degraded in six different ways: Lossy compression, Equalization, Mixing with noise, Low pass filtering, Scaling, and Loudspeaker-Microphone transmission in a noisy environment.

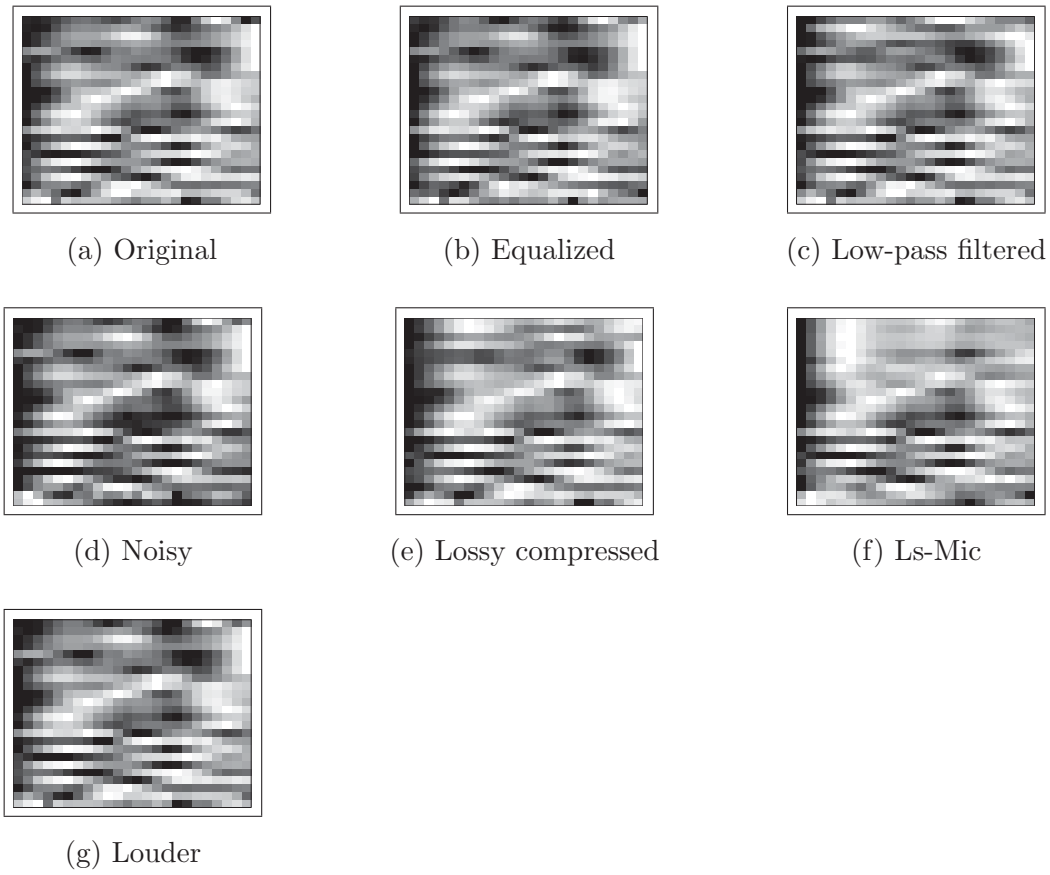
3. From each of the 2,800 audio files (including originals) obtained in the previous step an excerpt of 5 seconds was extracted, therefore all those degradations were combined with cropping and desynchronization at the same time.

4. The short signatures of the 2,800 excerpts that resulted from the previous step were determined. In Figure 4.9 the entropygrams of an excerpt of the song *Diosa del cobre* corresponding to the seven (including original) versions considered are shown, their corresponding signatures are shown in Figure 4.10.

5. All the short signatures determined in the previous step were searched inside every whole song's signature from the collection of 4,000 determined in the first step using the nearest neighbor criterion. For example, the nearest signature to those shown in

Figure 4.10 was found inside the piece of the whole song's signature that is magnified in Figure 3.5 and shown again in Figure 4.8.

The Hamming distance was used to establish how different the MBSES of two excerpts are from each other. The Hamming distance between two binary matrices can be conceived as a measure of fullness of the matrix that results from computing the absolute difference between them. Figure 4.11 shows the differences found between the degraded versions of an excerpt of the song *Diosa del cobre* and the nearest neighbor found inside that song, precisely the one shown in Figure 4.8.

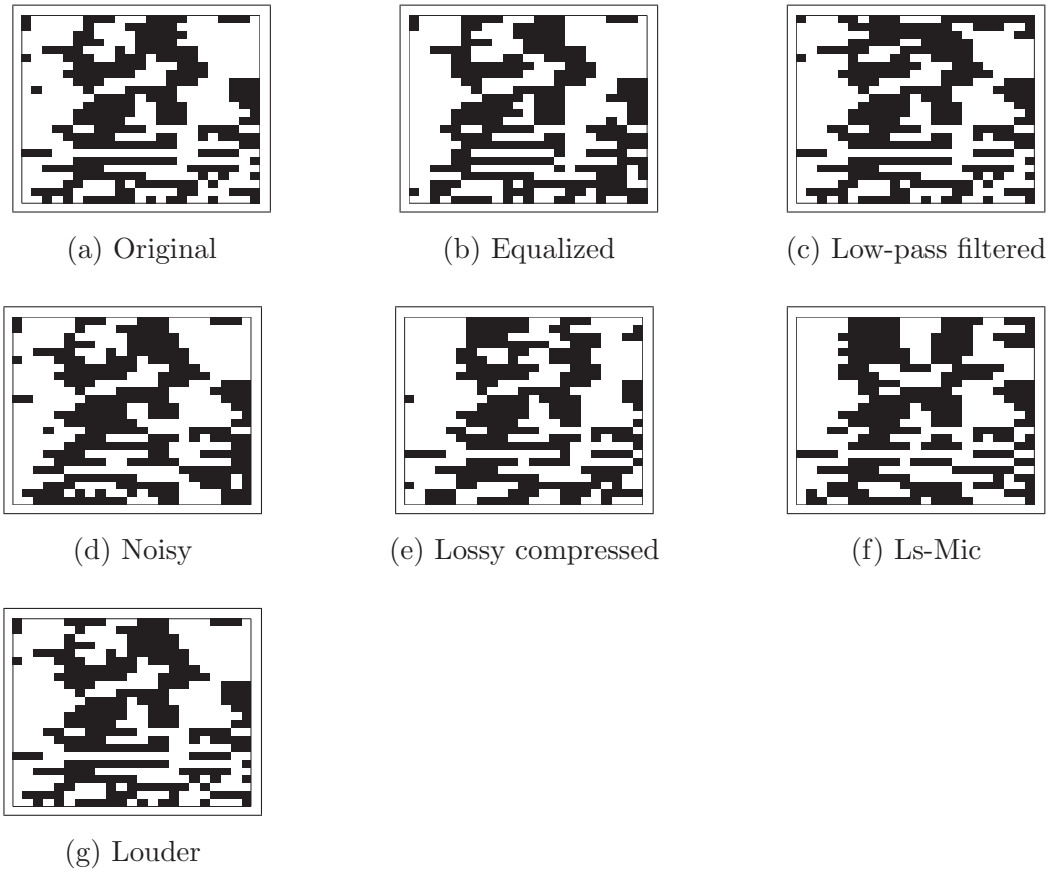


(a) Original (b) Equalized (c) Low-pass filtered

(d) Noisy (e) Lossy compressed (f) Ls-Mic

(g) Louder

Figure 4.9: Entropygrams of several degraded versions from a 5-second excerpt of the song *Diosa del cobre*



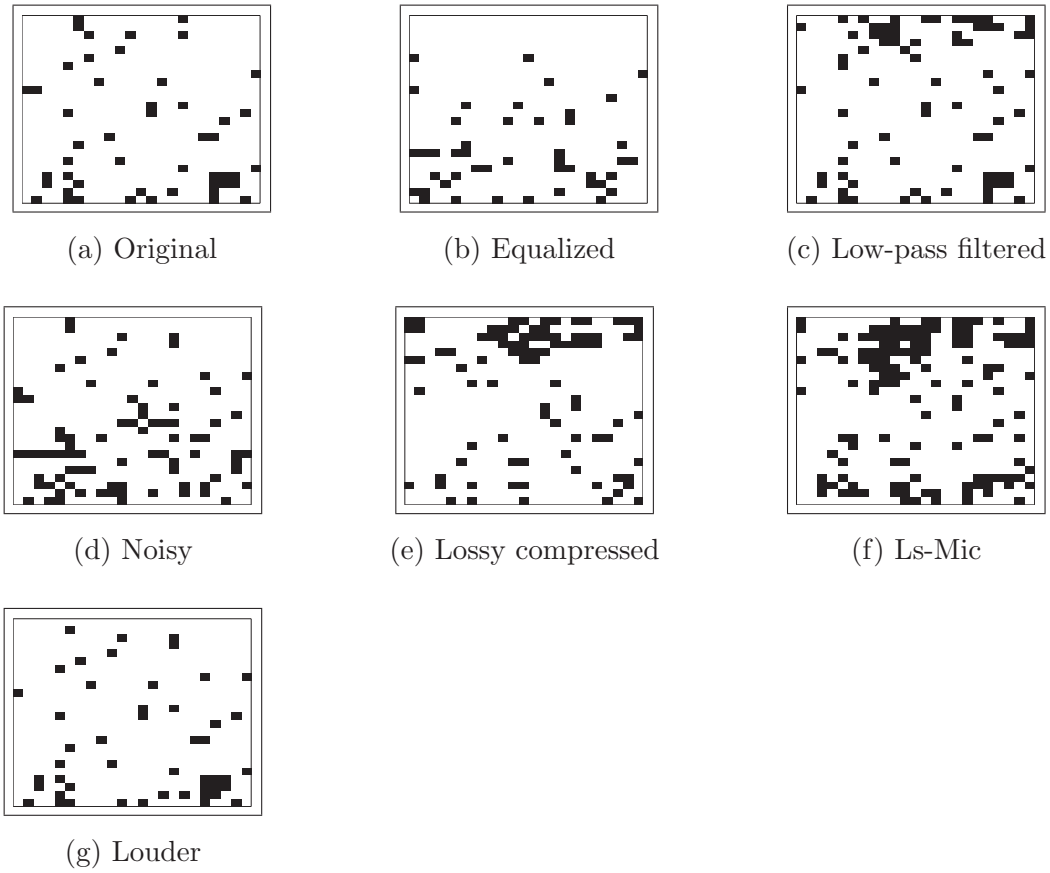
(a) Original (b) Equalized (c) Low-pass filtered

(d) Noisy (e) Lossy compressed (f) Ls-Mic

(g) Louder

Figure 4.10: MBSES of excerpts of five seconds of the degraded versions of the song *Diosa del cobre*

Not even the excerpts extracted from the original songs were found without errors. To understand this fact, consider that the probability for the first frame of the randomly selected excerpts to begin exactly at the same instant than any frame of the song is practically zero, so the experiment is reproducing a real scenario, this effect is known as desynchronization or time-shift. In Figure 4.12, the distance between a excerpt of a song and the most similar (i.e. closest Hamming distance) segment of audio inside the same song is plotted as a function of the time-shift. To generate the curve of Figure 4.12 a song at 44,100 samples per second was used, therefore a frame of 0.37 sec consists of 16,384 samples (i.e. the frame size). The first excerpt was extracted beginning at a position that was a multiple of the frame size (i.e. zero time-shift); this excerpt was of course found inside the song without errors (i.e. Hamming distance equal to zero) and corresponds to the first point



(a) Original (b) Equalized (c) Low-pass filtered

(d) Noisy (e) Lossy compressed (f) Ls-Mic

(g) Louder

Figure 4.11: Absolute differences, the fuller the images the greater the Hamming distances.

of the curve shown in Figure 4.12. The second excerpt was extracted beginning 100 samples (i.e. 2.2 ms) after the first excerpt, the most similar piece of audio inside the song was found with a normalized Hamming distance of 0.013, it corresponds to the second point of the curve shown in Figure 4.12. The third excerpt was extracted beginning 100 samples after the second excerpt and so on. Since the frames overlap fifty percent, a distance of zero is found again at a time-shift of 185 ms (i.e. half the frame size). Increasing the overlap btween frames surely results in an audio-fingerprint that is more robust to time-shift. Of course, a price in processing time and disk space would have to be paid if an increase in robustness is desired.

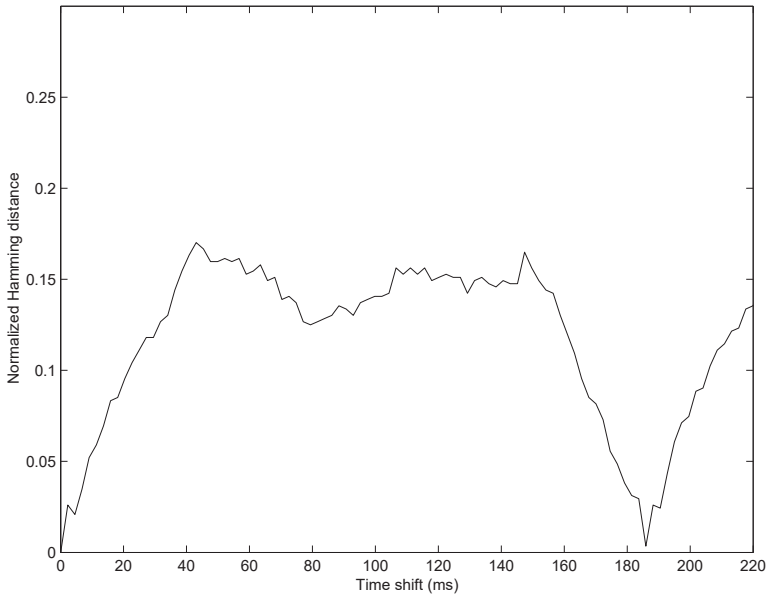


Figure 4.12: Hamming distance between an excerpt of a song and the most similar segment of audio inside the same song as a function of the time-shift

The MBSES of a song is a binary matrix with a number of rows that depends on the duration of the song and a fixed number of columns. For example, The MBSES of a song of 4 minutes and 39 seconds is a binary matrix of 1869×24. On the other hand, the MBSES of an excerpt is a binary matrix with a fixed size (i.e. 24×24 in our experiment) as long as the duration of the excerpts does not change.

In order to find the song to which the excerpt belongs to, we compared the short binary matrix (i.e. the MBSES) of the query excerpt with every possible sub-matrix with the same size inside every song's MBSES of the collection. For example, the nearest sub-matrix to those shown in Figure 4.10 was found inside the piece of the whole song's MBSES that is magnified in Figure 3.5 (and shown again in Figure 4.8). A brute force search procedure took approximately 20 seconds to answer a query in a 2.8 GHz Pentium 4 PC. The searching time was reduced to about 10 seconds using the following strategy: Instead of finishing the computation of the Hamming distance between any sub-matrix of the MBSES of any song and the MBSES of the query excerpt just to find that they are too different, just skip to the next sub-matrix as soon as the normalized Hamming distance between the first columns of the sub-matrix and the first columns of the query excerpt's MBSES is higher than 0.3, this

threshold was used because the normalized hamming distance between degraded versions of a song was never higher than 0.3.

Table 4.9 shows the precision rates for MBSES, Haitsma-Kalker's AFP and the AFP of MPEG-7 for the signal degradations considered. MBSES shows higher robustness to Noise addition, Re-recording, and Low-pass filtering.

Table 4.9: Precision rates of MBSES, MPEG-7 and Haitsma-Kalker's AFP for different signal degradations

Degradation	MBSES	MPEG-7	Haitsma-Kalker
Cropping and time-shift	100 %	100 %	100 %
Equalization, cropping and time-shift	100 %	100 %	40 %
Noise contamination, cropping and time-shift	100 %	63 %	20 %
Re-recording in noisy environment, cropping and time shift	100 %	79 %	10 %
Low-Pass filtering, cropping and time-shift	100 %	82 %	70 %
Lossy Compression,cropping and time-shift	100 %	100 %	80 %
Scaling, cropping and time shift	100 %	100 %	90 %

4.5. Experiment 4. Matching Musical Performances

The Audio-Fingerprints (i.e. TES, MBSES, Haitsma-Kalker's AFP, and the AFP of MPEG-7) from a collection of 4,000 audio files including several genres of music (i.e. classical, rock, pop, etc.) were extracted. For some of the songs or masterpieces of the collection, 124 to be accurate, we were able to obtain at least another musical rendition of it, for example for *Beethoven's Symphony Number 5 in C minor* performed by the *Berliner Philharmonische Orchester* conducted by Karajan we obtained the *Viena Philarmonic Orchestra* version conducted by Kleiber. The Audio-Fingerprints of these other performances were also extracted. Using the aligning techniques described on section 2.7, every song was searched in the collection of 4,000 songs to see if another performance of it could be found. The distance between every song from the collection of 124, and every song from the collection of 4,000 were stored for further studies (i.e. obtaining the ROC curves).

4.5.1. Specific details for Experiment 4

Those who want to reproduce these experiments should take into account the following considerations:

Using the Longest Common Subsequence Distance

The LCS distance produce short distances when the same sequence of acoustic events are present in both musical performances being compared. In this context a symbol represents an acoustic event. To use MBSES as a string, the symbols are considered to belong to an alphabet that is too large (2^{24}). It would be naive to consider two symbols as completely different just because they differ in one bit, remember that the symbols are made of bits that result from an analysis on unrepeatable audio segments. We considered two symbols as different only if the normalized Hamming distance was greater than 0.3, this threshold was used based on the knowledge that no frame of 24 bits was found to have changed in more than 7 bits when the songs were degraded. Finally, the costs of insertions and deletions used were both of 1.0.

Using the Levenshtein distance

Using a threshold to decide whether an acoustic event equals another may seem dangerous, therefore, instead of throwing away the normalized Hamming distance between feature frames, they are used as the substitution cost for the Levenshtein distance, while keeping the insertion and deletion costs to 1.

$$C_{i,j} = min \begin{cases} C_{i-1,j-1} + d(t_i, p_j) \\ C_{i,j-1} + 1 \\ C_{i-1,j} + 1 \end{cases} \tag{4.7}$$

where $d(t_i, p_j) = Hamming(t_i, p_j)/24$ since t_i and p_j are made from 24 bits and we want $d(t_i, p_j)$ to be a value between 0 and 1.

Matching Musical Performances with TES

The TES of a performance is a binary string, it is very easy to align with the TES of another performance, for example, a short segment of 9 seconds of audio coded as 11001001 would have a Levenshtein distance of 5.0 with the 10 seconds segment of audio coded as 110110010 (11001001 → 110101001 → 1101101001 → 110110001 → 1101100101 → 110110010).

Normalizing the distances

The Levenshtein distance between performances of length N and M cannot be greater than the length of the longest one of them, so in order to normalize the Levenshtein distance it was divided by $max(N, M)$. Once normalized, it was possible to set a threshold to decide whether two performances match or not. The LCS distance cannot be grater than $N + M$, so in order to normalize the LCS distance it was divided by $N + M$. The DTW distance was also divided by $N + M$.

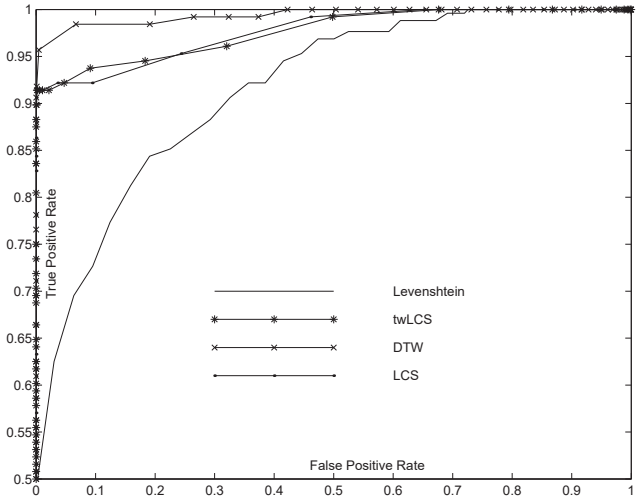


Figure 4.13: ROC curves that resulted from using Haitsma-Kalker's AFP for the considered aligning techniques

4.5.2. Results on Matching Musical Performances

Figure 4.13 shows the ROC curves that resulted from using Haitsma-Kalker's AFP. DTW turned out to be the best aligning technique for this AFP. The Levenshtein distance did not work well with Haitsma-Kalker's AFP.

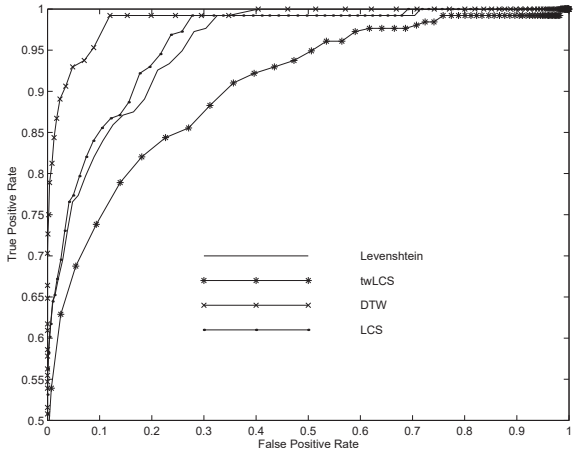


Figure 4.14: ROC curves that resulted from using the AFP of MPEG-7 for the considered aligning techniques

Figure 4.14 shows the ROC curves that resulted from using the AFP of MPEG-7. To use flexible string matching techniques, the AFP of MPEG-7 from every song (which consist of a sequence of floating point vectors) was converted into a string (i.e. a sequence of integer values) through vector quantization. However, for DTW the original floating point vectors were used. Therefore, in the case of this AFP the poor results of LCS, Levenshtein, and twLCS compared with DTW might be the consequence of the loss of precision associated with vector quantization. Levenshtein and LCS distances resulted with similar performances being LCS slightly better. Finally, twLCS turned out to be the worst choice for this audio-fingerprint.

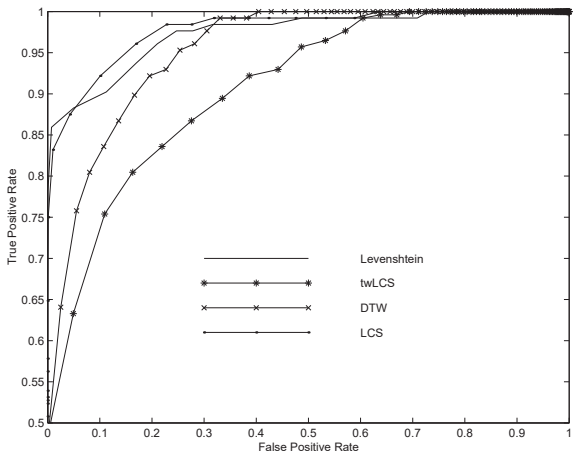


Figure 4.15: ROC curves that resulted from using TES for the considered aligning techniques

Figure 4.15 shows the ROC curves that resulted from using TES. The best aligning techniques turned out to be LCS and Levenshtein and twLCS had the worst performance.

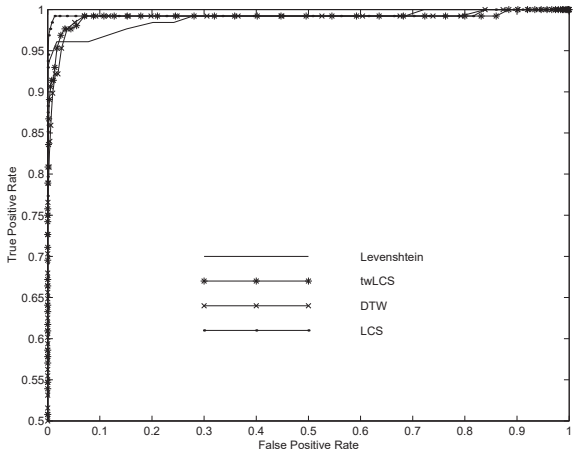


Figure 4.16: ROC curves that resulted from using MBSES for the considered aligning techniques

Figure 4.16 shows the ROC curves that resulted from using MBSES. With MBSES all the aligning techniques performed very well. It seems as if the selection of the aligning technique was not as important as the selection of the audio-fingerprint. Nevertheless, LCS outperformed the other aligning techniques.

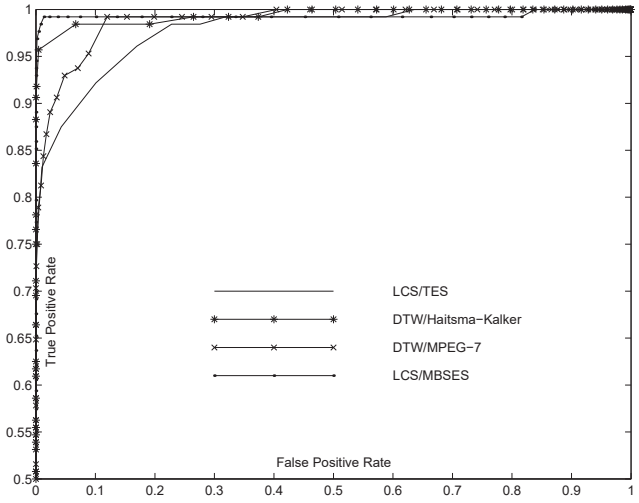


Figure 4.17: Best ROC curves for the considered AFP/Aligning technique combinations

In Figure 4.17 we clearly see that the best choice for aligning musical performances among those tried in this work is achieved by using MBSES on the front-end and LCS as the aligning technique.

The area under a ROC curve is frequently used for scoring the performance of classification systems, the greater the area under the ROC curve is, the better the classification system works. The left part of a ROC curve is much more relevant than the right part, this fact is disregarded when the area under the ROC curve is used for scoring classification systems. Therefore, we decided to list the true positive rates for the false positive rates of 0, 1 %, 2 %, and 5 % (i.e. the leftmost part of the ROC curves) on Table 4.10. The best precision rates for the considered Aligning-technique/Audio-signature combination are printed in boldface on Table 4.10.

Table 4.10: Precision Rate for the all the Aligning-Technique/Audio-Signature combinations used in the tests

Align/AFP	FPR=0	FPR=0.1 %	FPR=1 %	FPR=2 %	FPR=5 %
LCS/SES	77.34 %	**94.53 %**	**98.44 %**	**99.22 %**	**99.22 %**
twLCS/SES	76.56 %	81.25 %	91.41 %	96.88 %	97.66 %
Leven/SES	**85.94 %**	92.97 %	96.09 %	96.09 %	96.09 %
DTW/SES	67.97 %	77.34 %	90.63 %	92.19 %	98.44 %
LCS/E2S	82.81 %	91.41 %	91.41 %	92.19 %	92.19 %
twLCS/E2S	83.59 %	90.63 %	91.41 %	91.41 %	92.19 %
Leven/E2S	46.88 %	49.22 %	52.34 %	58.59 %	68.75 %
DTW/E2S	76.56 %	91.80 %	96.09 %	96.88 %	97.66 %
LCS/SFS	50.78 %	57.03 %	64.45 %	67.19 %	77.73 %
twLCS/SFS	46.88 %	48.44 %	58.20 %	62.89 %	68.75 %
Leven/SFS	50.78 %	57.03 %	65.23 %	67.19 %	76.56 %
DTW/SFS	68.75 %	72.66 %	82.03 %	88.28 %	93.75 %
LCS/TES	68.75 %	75.00 %	83.20 %	86.72 %	89.45 %
twLCS/TES	46.88 %	49.22 %	52.34 %	57.81 %	68.36 %
Leven/TES	70.70 %	82.81 %	86.72 %	87.89 %	88.28 %
DTW/TES	46.88 %	49.22 %	57.81 %	61.72 %	73.44 %

4.6. Conclusions of the chapter

The purpose of the experiments reported in this chapter was to evaluate the robustness of MBSES and TES with respect to state-of-the-art audio-fingerprints. The state-of-the-art AFP is the AFP of MPEG-7 but the most cited audio-fingerprint of Haitsma and Kalker was an obligated reference. In the experiments reported in this chapter, MBSES emerged as the most robust audio-fingerprint available.

Chapter 5

Conclusions and Future Work

In this thesis report two robust audio-fingerprints based on Shannon's entropy where introduced as TES and MBSES. Audio-fingerprints have to be evaluated from several angles, basically robustness, compactness, time complexity, granularity, and scalability. However, robustness is definitively the most important aspect of an audio-fingerprint, that is why all the experiments that were done in this research were designed to evaluate this relevant characteristic. Therefore, we begin this short chapter with some conclusions on robustness, and then add some comments on the other aspects of these AFPs.

1. Robustness.

   The Multi Band spectral entropy signature has proved to be highly robust to heavy degradations of the audio signals. MBSES was found to be more robust than TES specifically for equalization, noise contamination and loudspeaker to microphone transmission in a noisy environment (LsMic). MBSES was found to be more robust than the AFP of MPEG-7 specifically for Noise contamination, LsMic and low-pass filtration. Finaly, MBSES was found to be more robust than Haitsma-Kalker's AFP under equalization, noise contamination, LsMic, lowpass filtration, lossy compression and scaling.

   It is very interesting how the MBSES of the low pass filtered songs did not change significatively even when only 8 out of the 24 considered critical bands fall below the cutoff frequency of 1 KHz. To understand this effect, remember that a low-pass

filter attenuates the contents of the signal above the cutoff frequency gradually as the frequency increases. The Butterworth filter of second order (See [Proakis92]) used to filter the audio signals attenuates the signal at a rate of with -20 dB/decade, this means that the signal's amplitude declines to a tenth for a frequency that is ten times the cutoff frequency. Since the entropy value depends on the distribution of the spectrum for each considered band, disregarding its amplitude, then the entropy value does not change significatively except for the last 4 bands. You can see how the absolute differences shown in Figure 4.11(c) are not relevant below the first 20 bands, only the last 4 bands (at the top) seem affected.

2. Compactness.

   MBSES stores only 24 bits every 185 ms, it is a very compact finger-print of only 0.13 kbit/s. As a reference, Haitsma-Kalker's AFP [Haitsma02] requires 2.6 kbit/s. The AFP of MPEG-7 is an AFP with a variable resolution, its compactness is in the range of 0.76 to 4.6 kbit/s. TES is an extremely compact AFP of only 0.001 kbit/s but it was designed for whole songs only (not for small excerpts of audio).

3. Time complexity.

   As specified in Subsections 3.2.1 and 3.3.3, the time complexity of the TES and MBSES extraction algorithms are both linear on the length of the audio-signal. However, the time complexity of the algorithms for determining the AFP of MPEG-7 and Haitsma-Kalker's AFP are also linear on the length of the audio-signal. TES is determined in Time-domain and therefore faster to compute. MBSES, Haitsma-Kalker's AFP and the AFP of MPEG-7 are determined in the frequency domain and are determined in similar times. Table 5.1 shows the time it took to determine the considered audio-signatures of a song of 4 minutes and 39 seconds using a personal computer with processor Pentium 4 (2.8GHz) and 512 MB of RAM.

4. Granularity.

   In this thesis, we show the results of the experiments where excerpts of five seconds were used to identify a song. However, we experimented with excerpts of 10, 15 and 20

Table 5.1: AFP Extraction timing for a song of 4 minute and 39 seconds using a personal computer with a Pentium 4 processor (2.8 GHz)

Audio-Fingerprint	Time in seconds
TES	8.5
MPEG-7	15.5
MBSES	20
Haitsma-Kalker's	24.5

seconds as well. An elementary observation from such experiments is that the shorter the excerpt was, the higher the required resolution (i.e. greater overlap percentage) to identify a song.

5. Scalability.

   The first row of Table 4.4 is the accuracy rate of searching original songs among 228 audio files. In experiment 2, the songs were searched inside a collection of 4 000 songs, no decrease of the precision rate of MBSES is observed with the increase of the the database size. Metric indexing, as surveyed in [Chavez01], could be used to speed up searches.

6. Polyphonic Audio Matching.

   The audio fingerprint based on Multi-Band spectral entropy (MBSES) emerged as the more adequate AFP for the problem of matching musical performances among those included in our tests. The LCS distance performed better as an aligning technique and since it has the versatility of flexible string matching techniques, it is more adequate for applications like broadcast monitoring of musical performances. It was a surprise to discover that LCS worked better than twLCS since it was reported otherwise in [Guo04], two facts explain this result, the first is that the experiments reported in [Guo04] were carried out with music that was "monophonic in nature" (i.e. folk music) while we used polyphonic music instead. The second fact is that as described in [Guo04], they did not used real performances for they experiments, they "shortened and lengthened randomly selected notes to simulate inaccuracies in rhythm" and "stretched out the song by factors up to four" to obtain simulated performances. We

instead collected real musical performances for this work. We do not doubt that by repeating values of audio fingerprints twLCS might work well, unfortunately that does not occur with real musical performances.

5.1. Future work

There are several aspects that we are willing to work on in the future, they are:

1. Experiment with alternative ways of estimating the amount of information.

   In this thesis, Shannon's entropy was used intensively as a way of estimating the expected information content in a signal. Recent work on *Theory of Information* presented Renyi's entropy as an alternative to Shannon's entropy for information estimation purposes [Principe00]. Renyi's entropy of order α is computed with Equation (5.1).

   $$H_{R\alpha} = \frac{1}{1-\alpha} log\Big(\sum_{k=1}^{N} p_k^{\alpha}\Big) \tag{5.1}$$

   There is a relation between Renyi's entropy and Shannon's entropy given by Equation (5.2).

   $$\lim_{\alpha \to 1} H_{R_\alpha} = H_S \tag{5.2}$$

   Renyi's Entropy tends to Shannon's entropy when its order α tends to one. However, Renyi's entropy of second order or above might work better than Shannon's entropy for audio-fingerprinting purposes.

2. Shannon made rigorous the idea that the the entropy of a process is the amount of information in the process. He focused on memoryless sources whose probability distribution did not change with time and whose outputs were drawn from a finite alphabet. However, the measures of information have recently been extended to more general random processes [Gray90]. We are willing to find out if these new unrestricted ways of measuring the information content of random processes produce better results in robust audio-fingerprinting.

3. Use of Multi-Band Spectral Entropy for Automatic Speech Recognition (ASR).

   In Appendix D some experiments on the use of Entropy-Spectrograms for the characterization of utterances are reported. In these experiments, the Entropy-Spectrograms seems to be a slight improvement to Energy-spectrograms and LPC-12 coefficients. In the future we intend to validate this results by using the TIMIT database. Some experiments with noisy speech are also required, perhaps using the AURORA database. The use of the Continuous Fourier Transform explained in Appendix A apparently does not lead to any improvement according to our experiments, this might be due to a not so chaotic nature of the speech signal. Again this should be validate with more experiments. The use of predictograms using the kernel predictor explained in Appendix B came out as the worst choice, apparently, the gaussian nature of the speech spectrum makes the blind method of kernel predictability useless for speech recognition purposes.

4. Indexing techniques for Music Information Retrieval (MIR).

   To search a song among hundreds of thousands of them using only a small degraded excerpt as a query is a challenging problem. The search has to be done in milliseconds if thousands of queries per second have to be answered. Haitsma and Kalker use an inverted index [Haitsma02], Cheng Yang uses locally sensitive hashing (LSH) combined with the Hough Transform [Yang02]. We are working in the design of an index that should reduce the searching time of the state-of-the-art indexes while preserving the precision rate of the brute-force procedure that was used in our test for the assessment of robustness.

References

[Allamanche02] Allamanche, E., Herre, J., Helmuth, O., Froba, B., Kasten, T., and Cremer, M. Content-based identification of audio material using mpeg-7 low level description. *In International Symposium on Music Information Retrieval ISMIR*. 2002.

[Bank01] Bank, M., Podoxin, S., and Tsingouz, V. Estimation of non distortion audio signal compression. *World Scientific and Engineering Society Press*, pages. 104 – 110, 2001.

[Bartsch05] Bartsch, M. and Wakefield, G. Audio thumbnailing of popular music using chroma-based representations. *IEEE transactions on Multimedia*, 7:96–104, feb 2005.

[Batlle04a] Batlle, E., Masip, J., and Guaus, E. Amadeus: a scalable hmm-based audio information retrieval system. *In First International Symposium on Control, Communications and Signal Processing*, pages. 731– 734. March 2004.

[Batlle04b] Batlle, E., Masip, J., Guaus, E., and Cano, P. Scalability issues in an hmm-based audio fingerprinting. *In IEEE International Conference on Multimedia and Expo (ICME 2004)*, pages. 735 – 738. July 2004.

[Bercher00] Bercher, J. and Vignat, C. Estimating the entropy of a signal with applications. *IEEE Transactions on Signal Processing*, 48(6):1687–1694, June 2000.

[Bilmes98] Bilmes, J. A. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Tech report TR-97-021, Department of Electrical Engineering and Computer Science U.C. Berkeley, April 1998.

[Campos92] Campos, R. G. and Juarez, L. Z. A discretization of the continuos fourier transform. *Nuovo Cimento*, (107):703–711, 1992.

[Campos95] Campos, R. G. A quadrature formula for the hankel transform. *Numerical Algorithms*, 9(3-4):343–354, 1995.

[Cano99] Cano, P., Loscos, A., and Bonada, J. Score-performance matching using hmms. *In ICMC99*. Audiovisual Institute, Pompeu Fabra University, Spain, 1999.

[Cano02] Cano, P., Battle, E., Kalker, T., and Haitsma, J. A review of algorithms for audio fingerprinting. *Multimedia Signal Processing,IEEE Workshop on*, pages. 169–167, December 2002.

[cd2] cdtomp3freeware.exe.
URL `http://www.eusing.com/`

[Chavez01] Chavez, E., Navarro, G., Baeza-Yates, R., and Marroquin, J. Proximity searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[Dixon05a] Dixon, S. Live tracking of musical performances using on-line time warping. *In 8th International Conference on Digital Audio Effects (DAFx'05)*. Austrian Research Institute for Artificial Intelligence, Vienna, september 2005.

[Dixon05b] Dixon, S. and Widmer, G. Match: A music alignment tool chest. *In 6th International Conference on Music Information Retrieval (ISMIR)*. Austrian Research Institute for Artificial Intelligence, Vienna, 2005.

[Duda01] Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification*. John Wiley and Sons Inc., 2001.

[Eca] ecasound-2.4.5.tar.gz.
URL `http://eca.cx/ecasound/`

[EQp06] Foobar2000 equalizer presets eq_presets.zip, 2006.
URL `http://sjeng.org/ftp/fb2k/eq_presets.zip`

[Fawcett03] Fawcett, T. Roc graphs: Notes and practical considerations for researchers. Tech report HPL2003-4, HP Labs Tech, 2003.

[Fredriksson04] Fredriksson, K. and Navarro, G. Average-optimal single and multiple approximate string matching. *ACM Journal of Experimental Algorithmics (JEA)*, 9(1.4):1–47, 2004.

[Gomez-García06] Gomez-García, H. F., Marroquin, J. L., and Van Horebeeck, J. Image registration based on kernel predictability. 2006.

[Gray90] Gray, R. *Entropy and Information Theory*. Springer-Verlag, 1990.

[Group01] Group, M. A. *Text of ISO/IEC Final Draft International Standar 15938-4 Information Technology - Multimedia Content Description Interface - Part 4: Audio*. July 2001.

[Guo04] Guo, A. and Siegelmann, H. Time-warped longest common subsequence algorithm for music retrieval. *In 5th International Conference on Music Information Retrieval (ISMIR)*. 2004.

[Gusfield97] Gusfield, D. *Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology*. Cambridge University Press, 1997.

[Haitsma02] Haitsma, J. and Kalker, T. A highly robust audio fingerprinting system. *In International Symposium on Music Information Retrieval ISMIR*. 2002.

[Hellman72] Hellman, R. P. Asymmetry of masking between noise and tone. *Perception and Psychophysics*, 11:241–246, 1972.

[Hellmuth01] Hellmuth, O., Allamanche, E., Cremer, M., Kastner, T., Neubauer, C., Schmidt, S., and Siebenhaar, F. Content-based broadcast monitoring using mpeg-7 audio fingerprints. *In International Symposium on Music Information Retrieval ISMIR*. 2001.

[Herre01] Herre, J., Allamanche, E., and Hellmuth, O. Robust matching of audio signals using spectral flatness features. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages. 127–130, 2001.

[Hu03] Hu, N., Dannenberg, R. B., and Tzanetakis, G. Polyphonic audio matching and alignment for music retrieval. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[Ibarrola06] Ibarrola, A. C. and Chavez, E. A robust entropy-based audio-fingerprint. *In IEEE International Conference on Multimedia and Expo (ICME2006)*, pages. 1729–1732. July 2006.

[Kosugi00] Kosugi, N., Nishihara, Y., Sakata, T., Yamamuro, M., and Kushima, K. A practical query-by-humming system for a large music database. *ACM Multimedia*, pages. 333–342, 2000.

[Kurth03] Kurth, F. and Scherzer, R. A unified approach to content-based and fault tolerant music recognition. *In 114th AES Convention, Amsterdam, NL*. 2003.

[Lin06] Lin, H., Ou, Z., and Xiao, X. Generalized time-series active search with kullbak-leibler distance for audio fingerprinting. *IEEE Signal Processing Letters*, 13(8):465 – 468, August 2006.

[Logan00] Logan, B. Mel frequency cepstral coefficients for music model-

ing. *In International Symposium on Music Information Retrieval (ISMIR)*. oct 2000.

[Martin01] Martin, R. Noise power spectral density estimation based on optimal smoothing and minimum statistics. *IEEE Transactions on Speech and Audio Processing*, 9(5):504–512, July 2001.

[Mathews00] Mathews, J. H. and Fink, K. D. *Numerical Methods withMatlab 3rd Edition*. Prentice Hall, 2000. ISBN 84-8322-181-0.

[Mohammad-Djafari94] Mohammad-Djafari, A. Entropy in signal processing. *Traitement du signal*, pages. 87–116, 1994.

[Mus02] Musicbrainz trm musicbrainz-1.1.0.tar.gz, 2002.
URL `ftp://ftp.musicbrainz.org/pub/musicbrainz/`

[Navarro97] Navarro, G. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 1997.

[Navarro02] Navarro, G. and Raffinot, M. *Flexible Pattern Matching in Strings. Practical On-Line Search for Texts and Biological Sequences*, volume 17. Cambridge University Press, 2002.

[Pauws04] Pauws, S. Musical key extraction from audio. *In International Symposium on Music Information Retrieval ISMIR*. 2004.

[Pickens02] Pickens, J., Bello, J. P., Monti, G., Crawford, T., Dovey, M., Sandler, M., and Byrd, D. Polyphonic score retrieval using polyphonic audio queries. a harmonic modelling approach. *In 5th International Conference on Music Information Retrieval (ISMIR)*. 2002.

[Principe00] Principe, J., Xu, D., and Fisher, J. Information theoretic learning. *In* S. Haykin, ed., *Unsupervised Adaptive Filtering*. John Wiley & Sons, New York, 2000.

[Proakis92] Proakis, J., , and Manolakis, D. *Digital Signal Processing. Principles, Algorithms and Applications*. MacMillan, 1992.

[Rabiner78] Rabiner, L. and Schafer, R. *Digital Processing of speech signals*. Prentice Hall, Bell labs, AT&T, 1978.

[Ramalingam05] Ramalingam, A. and Krishnan, S. Gaussian mixture modeling using short time fourier transform features for audio fingerprinting. *In IEEE International Conference on Multimedia and Expo (ICME2005)*, pages. 1146 – 1149. July 2005.

[Sakoe78] Sakoe, H. and Chiba, S. Dynamic programming algortihm optimization for spoken word recognition. *IEEE transactions on Acoustics and Speech Signal Processing (ASSP)*, pages. 43–49, 1978.

[Seo05] Seo, J. S., Jin, M., Lee, S., Jang, D., Lee, S., and Yoo, C. D. Audio bingeprinting based on normalized spectral subband centroids. *In International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. 2005.

[Sert06] Sert, M., Baykal, B., and Yazici, A. A robust and time-efficient fingerprinting model for musical audio. *In IEEE Tenth International Symposium on Consumer Electronics (ISCE'06)*. 2006.

[Shalev-Shwartz02] Shalev-Shwartz, S., Dubnov, S., Friedman, N., and Singer, Y. Robust temporal and spectral modeling for query by melody. *In ACM SIGIR'02*. Hebrew University, Israel, August 2002.

[Shannon49] Shannon, C. and Weaver, W. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[Shen98] Shen, J.-L., Hung, J.-W., and Lee, L.-S. Robust entropy-based endpoint detection for speech recognition in noisy environments. *In International Conference on Spoken Language Processing*. Dec 1998.

[Shin02] Shin, S., Kim, O., Kim, J., and Choil, J. A robust audio watermark-

ing algorithm using pitch scaling. *In 14th International Conference on Digital Signal Processing*, volume 2, pages. 701 – 704. 2002.

[Shrestha04] Shrestha, P. and Kalker, T. Audio fingerprinting in peer-to-peer networks. *In 5th International Conference on Music Information Retrieval (ISMIR)*. 2004.

[Sigurdsson06] Sigurdsson, S., Petersen, K. B., and Lehn-Schioler, T. Mel frequency cepstral coefficients: An evaluation of robustness of mp3 encoded music. *In International Symposium on Music Information Retrieval (ISMIR)*. 2006.

[Sinitsyn06] Sinitsyn, A. Duplicate song detection using audio fingerprinting for consumer electronics devices. *In IEEE Tenth International Symposium on Consumer Electronics (ISCE'06)*. 2006.

[Sox] sox-12.18.2.tar.gz.
URL `http://sox.sourceforge.net/`

[Stanley84] Stanley, W., Dougherty, G., and Dougherty, R. *Digital Signal Processing. Principles*. Reston Publishing Company, 1984.

[Subramanya99] Subramanya, S., Simha, R., Narahari, B., and Youssef, A. Transform-based indexing of audio data for multimedia databases. *In International Conference on Multimedia Applications*. 1999.

[Sukittanon02] Sukittanon, S. and Atlas, E. Modulation frequency features for audio fingerprinting. *In IEEE, International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages. 1773–1776. University of Washington USA, 2002.

[Sukittanon05] Sukittanon, S., L.E., A., Pitton, J., and Filali, K. Improved modulation spectrum through multi-scale modulation frequency decomposition. *In IEEE, International Conference on Acoustics, Speech*

and Signal Processing (ICASSP), volume 4, pages. 517–520. University of Washington USA, 2005.

[Sungwoong07] Sungwoong, K. and Chang, D. Y. Boosted binary audio-fingerprint based on spectral subband moments. *In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2007)*. April 2007.

[VandePar06] Van de Par, S., McKinney, M., and Redert, A. Musical key extraction from audio using profile training. *In International Symposium on Music Information Retrieval ISMIR*. 2006.

[Viola95] Viola, P. and Wells, W. Alignment by maximization of mutual information. *In International Conference on Computer Vision*. 1995.

[xmm] xmss-1.2.10.tar.gz.
URL `http://www.xmms.org/`

[Yang02] Yang, C. Efficient acoustic index for music retrieval with various degrees of similarity. *In MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, pages. 584–591. ACM, New York, NY, USA, 2002. ISBN 1-58113-620-X. doi: http://doi.acm.org/10.1145/641007.641125.

[You04] You, H., Zhu, Q., and Alwan, A. Entropy-based variable frame rate analysis of speech signal and its applications to asr. *In International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2004.

[Zwicker90] Zwicker, E. and Fastl, H. *Psycho-Acoustics. Facts and Models*. Springer, 1990.

Glosario

AFP *Audio-Fingerprint*

ASR *Automatic Speech Recognition*

CFT *Continuous Fourier Transform*

DFT *Discrete Fourier Transform*

DTW *Dynamic Time Warping*

FN *False Negative*

FP *False Positive*

FPR *False Positive Rate*

fps *Frames Per Second*

FRR *False Rejection Rate*

kbps *kilo-bits per second*

LCS *Longest Common Subsequence*

LPC *Linear Prediction Coding*

LSH *Locally Sensitive Hashing*

MBSES *Multi-Band Spectral Entropy Signature*

MFCC *Mel-Frequency Cepstral Coefficients*

MIR *Music Information Retrieval*

MP3 *MPEG1,2 Layer 3*

MPEG *Moving Pictures Experts Group*

ROC *Receiver Operating Characteristics*

SCF *Spectral Crest Factor*

SFM *Spectral Flatness Measure*

SNR *Signal to Noise Ratio*

TES *Time-domain Entropy Signature*

TN *True Negative*

TP *True Positive*

TPR *True Positive Rate*

twLCS *time warped Longest Common Subsequence*

Appendix A

Continuous Fourier Transform

The Continuous Fourier Transform (CFT) is defined as in Equation (A.1), it may be applied to any signal. However, since the CFT has an infinite kernel its exact computation is not an option.

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt \tag{A.1}$$

The Discrete Fourier Transform (DFT) is defined as in Equation (A.2. The DFT has a kernel with the same size as the length of the signal as can be seen in Equation (A.3). When using the DFT, you are either assuming that the signal is periodic or accepting the fact that the DFT will consider the whole signal as a single period of a waveform that is periodic. However, the speech signal is not periodic, in fact, the unvoiced segments of the speech signal are particularly aperiodic.

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \tag{A.2}$$

$$\begin{bmatrix} X[0] \\ \hline X[1] \\ \hline X[2] \\ \hline \vdots \\ \hline X[N] \end{bmatrix} = \left[\begin{array}{c|c|c|c|c} 1 & 1 & 1 & \cdots & 1 \\ \hline 1 & W_N^1 & W_N^2 & \cdots & W_N^{N-1} \\ \hline 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)^2} \end{array}\right] \begin{bmatrix} x[0] \\ \hline x[1] \\ \hline x[2] \\ \hline \vdots \\ \hline x[N] \end{bmatrix} \tag{A.3}$$

Where $W_N = e^{-j2\pi/N}$

The CFT may be determined with high precision on the Hermite's zeroes as Campos demonstrated in [Campos92]. If enough values of the CFT are determined, a discretization of the CFT is obtained, Campos' discretization is computed using a finite kernel. In fact, the size of the kernel equals the number of the Hermite's zeroes where the CFT values are determined. The kernel for the discretization of the CFT is defined in Equation (A.4).

$$F_{k,l} = \frac{\pi}{\sqrt{2n}} \sqrt[4]{\frac{4n+3-x_l^2}{4n+3-x_k^2}}\, e^{jx_k x_l} \tag{A.4}$$

where:

- x is a vector with the roots of Hermite's Polynomial of degree n
- $f(x)$ is the signal at points x
- $g = Ff(x)$ is a vector with the CFT evaluated at the roots of the Hermite's Polynomial

To find the discretization of the CFT of a speech signal, we designed the following method:

1. Convert the sequence $0, 1, 2, ..., N-1$ into a sequence with N (the frame size) equidistant values from $-\pi$ to π
2. Find the coefficients of the trigonometric polynomial of degree $M < N/2$ given by Equation (A.5) that best adjusts to the speech signal waveform using Formulas (A.6) and (A.7) to find a_j and b_j respectively [Mathews00].

$$\frac{a_0}{2} + \sum_{j=1}^{M} [a_j cos(jx) + b_j sin(jx)] \tag{A.5}$$

$$a_j = \frac{2}{N} \sum_{k=1}^{N} [f(x_k) cos(jx_k)] \quad \forall \quad j = 0, 1, \ldots, M \tag{A.6}$$

$$b_j = \frac{2}{N} \sum_{k=1}^{N} [f(x_k) sin(jx_k)] \quad \forall \quad j = 1, 2, \ldots, M \tag{A.7}$$

3. Form vector x with the roots of the Hermite polynomial of degree P

4. Construct the Fourier's Kernel matrix F using Equation (A.8) according to [Campos95]. This Matrix is Hermitian so $F^{-1} = F^t$. Multiplying a signal vector by F is equivalent to finding a discretization of its CFT, multiplying by F^t would be a way of computing the inverse CFT.

$$F_{i,j} = \frac{\pi}{\sqrt{2n}} \sqrt[4]{\frac{4n+3-x_j^2}{4n+3-x_i^2}} [cos(x_i x_j) + jsin(x_i x_j)] \tag{A.8}$$

5. Evaluate the trigonometric polynomial found in step (2) in the hermite's zeros vector of step (3), call this vector f.

6. Compute $g = Ff$ so g will be the discretization of the CFT of f

Hermite's Polynomials are defined as in Equation (A.9).

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{x^2} \tag{A.9}$$

Hermite Polynomials form a complete orthogonal set on the interval with respect to the weighting function e^{x^2}, so they make an orthogonal system, meaning Equation (A.10) holds.

$$\int_{-\infty}^{\infty} e^{x^2} H_n(x) H_m(x) dx = 0 \qquad \forall n \neq m \tag{A.10}$$

The first ten Hermite's polynomials are written below, those of degree 2, 3 and 4 are plotted in Figure A.1.

$H_1(x) = 2x$

$H_2(x) = 4x^2 - 2$

$H_3(x) = 8x^3 - 12x$

$H_4(x) = 16x^4 - 48x^2 + 12$

$H_5(x) = 32x^5 - 160x^3 + 120x$

$H_6(x) = 64x^6 - 480x^4 + 720x^2 - 120$

$H_7(x) = 128x^7 - 1344x^5 + 3360x^3 - 1680x$

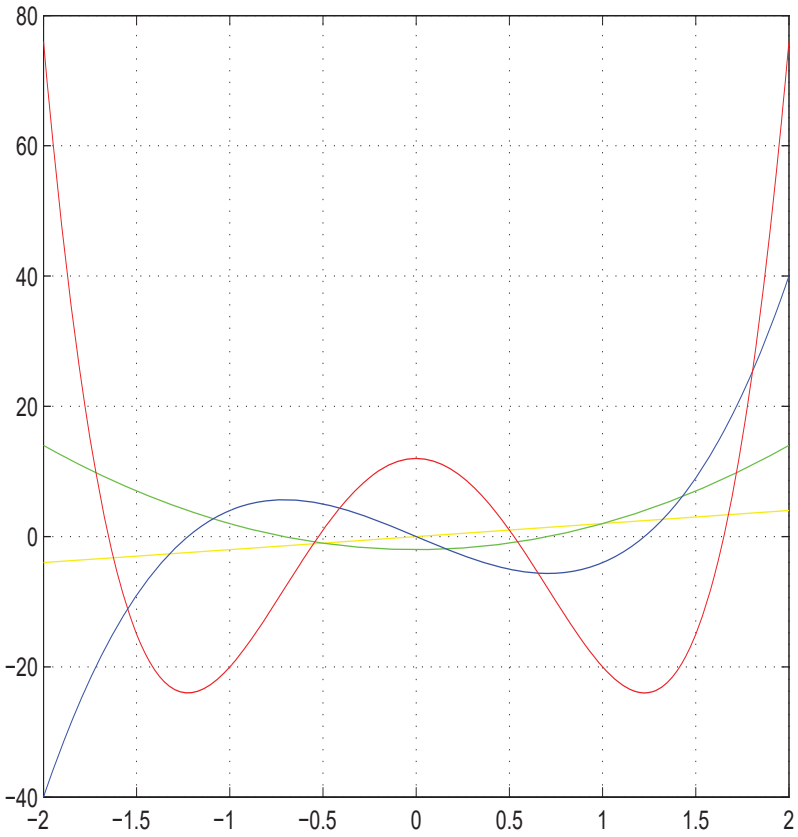


Figure A.1: Hermite's Polynomials

$H_8(x) = 256x^8 - 3584x^6 + 13440x^4 - 13440x^2$

$H_9(x) = 512x^9 - 9216x^7 + 48384x^5 - 80640x^3 + 30240x$

$H_{10}(x) = 1024x^{10} - 23040x^8 + 161280x^6 - 403200x^4 + 302400x^2 - 30240$

$\vdots$

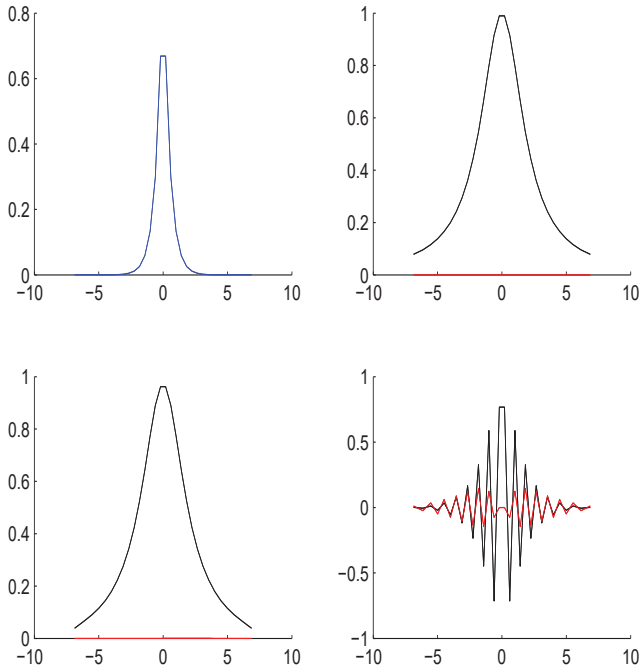


Figure A.2: Example of the discretization of the CFT.

As an example of the CFT, consider: $x(t) = e^{-2|t|}$ then:

$$X(j\omega) = \int_{-\infty}^{\infty} e^{-2|t|} e^{-j\omega t} dt = X(j\omega) = \frac{4}{\omega^2 + 4} \tag{A.11}$$

Figure A.2 shows $x(t)$, $X(j\omega)$, the discretization of the CFT of $x(t)$ and the DFT of $x(t)$. You can clearly see how the disretization of the CFT of $x(t)$ looks very similar to $X(j\omega)$ unlike the DFT of $x(t)$. Imaginary parts are plotted in red. The MATLAB code used to generate this figure is:

```
x=xiher(30); f=exp(-2*abs(x)); F=fourierker(30); g=F*f;
plot(x,real(g),'r'); plot(x,imag(g),'c'); gv=4./(x.^2+4);
plot(x,gv,'g') four=fft(f); plot(x,real(four),'k');
plot(x,real(four),'b');
```

As another example of the CFT, consider: $x(t) = sin(\frac{t^2}{2})$ then:

$$X(j\omega) = \pi[cos(\frac{\omega^2}{2}) + sin(\frac{\omega^2}{2})] \tag{A.12}$$

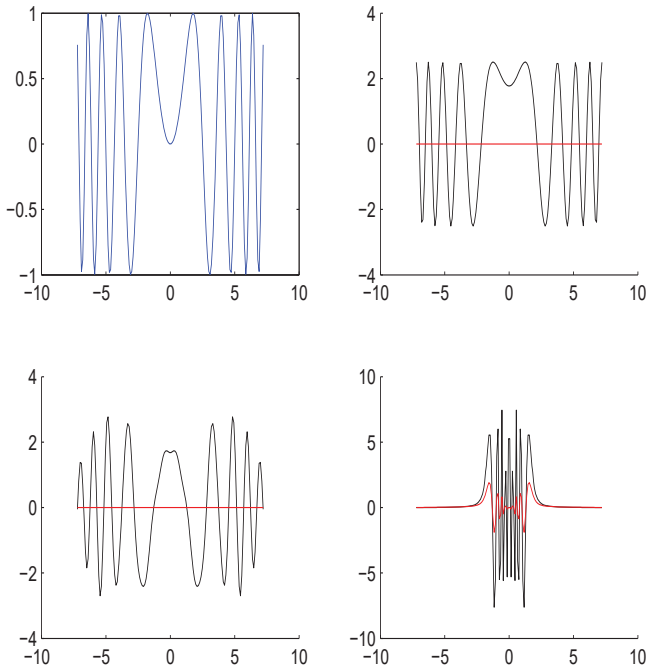


Figure A.3: Example of the discretization of the CFT

Figure A.3 shows $x(t)$, $X(j\omega)$, the discretization of the CFT of $x(t)$ and the DFT of $x(t)$. Again, you can clearly see how the disretization of the CFT of $x(t)$ looks quite similar to $X(j\omega)$ unlike the DFT of $x(t)$. Imaginary parts are plotted in red. The MATLAB code used to generate this figure is:

```
load her480;y=x;[F,xh,nf]=fkerasin(y);f=sin((xh.^2)/2);
g=F*f;plot(xh,abs(g),'r')
gv=(cos(xh.^2/2)+sin(xh.^2/2))*sqrt(pi);plot(xh,abs(gv),'g')
four=fft(f);plot(xh,abs(four),'b')
```

Hermite's polynomial roots are not spaced at equal distances. However, for Hermite's polynomials of high degree there is a linear range where the roots may be considered equidistant, see Figure A.4. We chose to use only the 170 roots of least magnitude of an Hermite's polynomial of degree 480.

When adjusting the speech signal to a trigonometric polynomial, we used one of degree 100 ($M = 100$) since using a lower degree affected the waveforms of unvoiced speech signals. The speech frames were made of 256 samples ($N = 256$) and the overlap of fifty

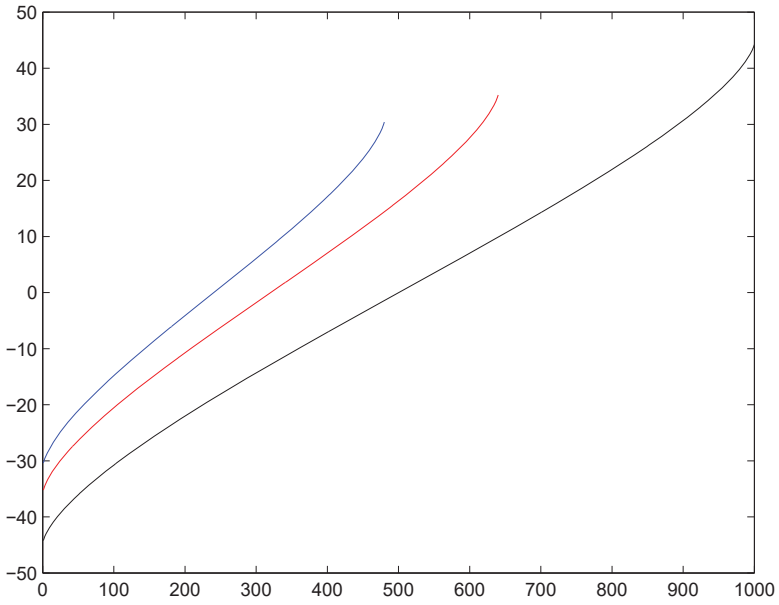


Figure A.4: In the Linear range, the zeroes of Hermite's polynomials are equidistant. blue: 480 Zeros Red: 640 Zeros Black 1000 Zeros

percent. No window was applied to the frames (which is one of the advantages of the CFT) and the sampling was made for telephonic quality (8 samples per second, 8 bits per sample, mono-aural). Algorithm 2 was used to determine the spectrogram of a utterance.

Figure A.5 shows the Energy-spectrograms determined using the CFT of two utterances of each word from the set of Spanish words: {"uno", "dos", "tres", "cuatro", "cinco"} spoken with a Mexican accent.

In order to compare Energy-Spectrograms determined using the CFT with the Energy-Spectrograms determined using the traditional DFT, Figure A.6 was included, it shows the Energy-Spectrograms determined using the DFT of two utterances of each word from the set: {"uno", "dos", "tres", "cuatro", "cinco"} spoken with a Mexican accent. Algorithm 3 was used to determine the Energy-spectrogram of a utterance using the DFT. This algorithm uses standard parameters for speech signal processing such as, frame size N of 30 ms, Hann Window and overlapping of 30 % between frames.

Algoritmo 2 Extraction of CFT spectrogram

Input:

Speech waveform.

Output:

Spectrogram

SpectrogramCFT()

1 Fill vector $\hat{x}$ with the roots of the Hermite's polynomial of degree n
2 Construct the finite kernel matrix F using $\hat{x}$
3 **while** *there are more frames*
4 Adjust the speech waveform to a trigonometric polynomial $p(x)$
5 Evaluate the trigonometric polynomial $\hat{x}$, call this vector f $(f = p(\hat{x}))$
6 Compute $g = Ff$ (CFT for this frame)
7 Add vector g to the spectrogram

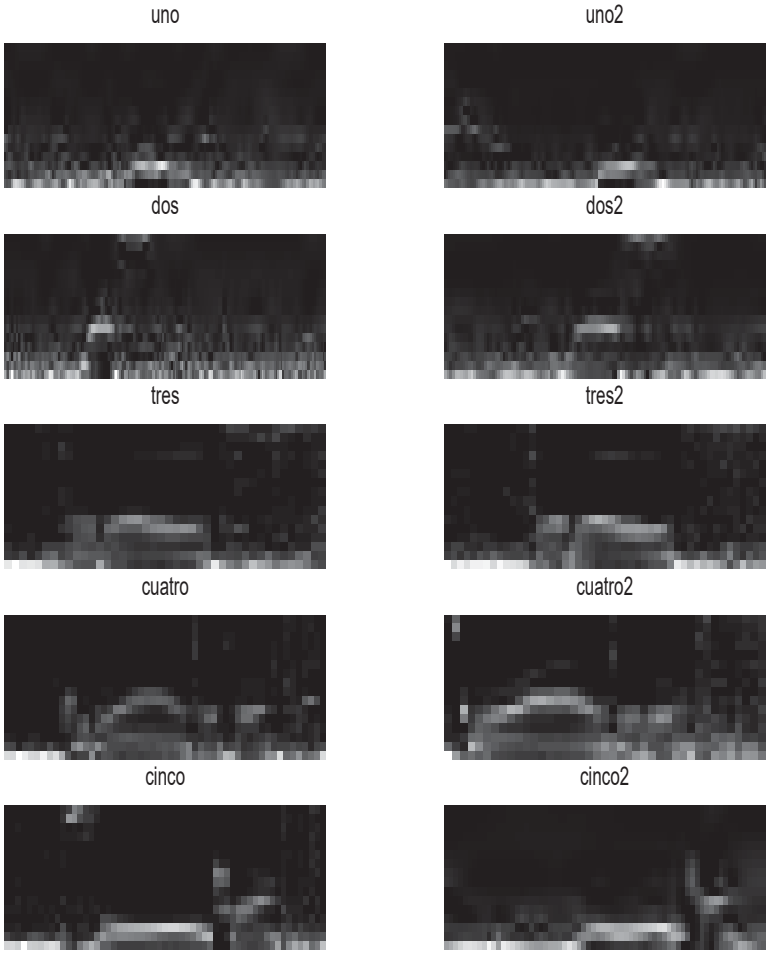


Figure A.5: CFT-Spectrograms of two utterances of 5 spanish digits spoken with a Mexican accent

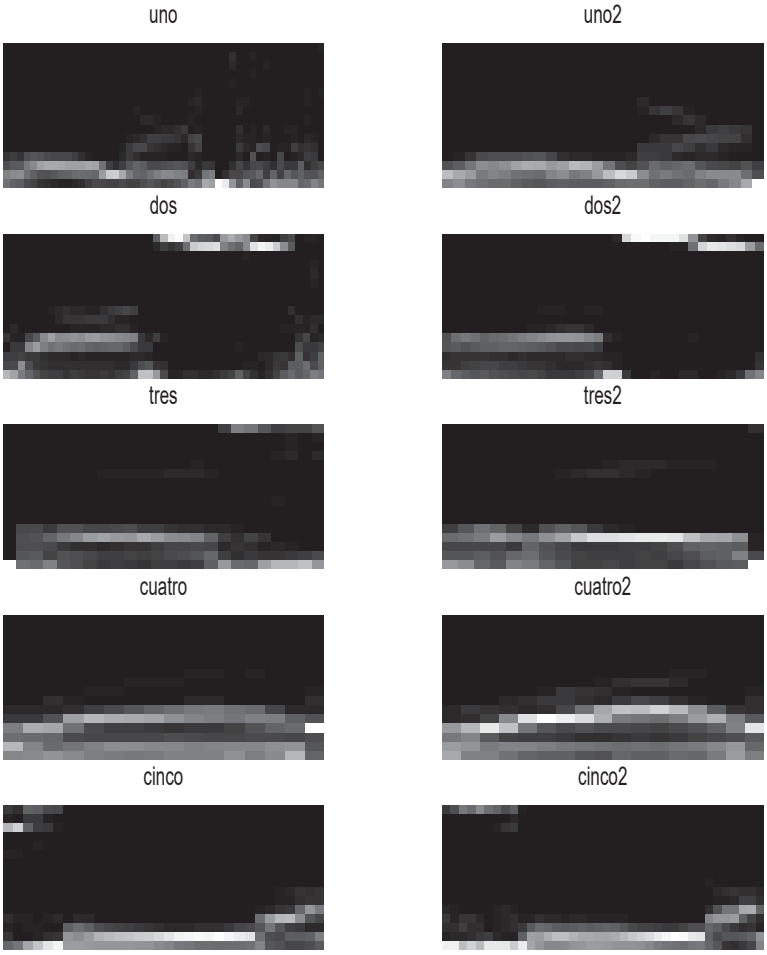


Figure A.6: DFT-Spectrograms of two utterances of 5 spanish digits spoken with a Mexican accent

Algoritmo 3 Algorithm to determine the spectrogram of a utterance

1. Read a Frame of size N

2. Multiply the frame by the Hamming Window

3. Determine the DFT of the resulting sequence

4. For the first nineteen critical bands according to Bark's scale:

   - Compute the energy of critical band b

     $$E = \sum_{i \in b} x_i^2 + y_i^2$$

     rem x and y are the real and imaginary parts of the DFT respectively.

   - Divide E by N and by wss

     rem wss (Window Squared and Summed) is the energy in the Hamming window

5. Add the vector of nineteen energies to the sequence of vectors (i.e. the spectrogram)

6. go to (1)

Appendix B

Kernel Predictors

A kernel predictor measures how predictable a signal is, it is opposite to the concept of Entropy. Kernel predictors were used for image registration in [Gomez-García06]. KP^1 was defined in [Gomez-García06] as Equation (B.1).

$$KP^1 = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} K(X_i, X_j) \tag{B.1}$$

where the kernel K is the gaussian:

$$G_\sigma(x_1, x_2) = exp(-\frac{||x_1 - x_2||^2}{2\sigma^2}) \tag{B.2}$$

σ is a free parameter.

The MATLAB code for computing this kernel predictor is:

```
function kp=kernelPredictor(x,sigma)
    n=length(x);
    suma=0;
    for i=1:n-1
        for j=i+1:n
            suma=suma+exp(-(x(i)-x(j))^2/(2*sigma^2));
        end
    end
    kp=suma*2/((n)*(n-1));
```

By computing the predictability for each critical band of every frame of a utterance, we build the predictogram of the utterance. Algorithm 4 may be used for extracting the predictogram of a speech signal.

Algoritmo 4 Extraction of a predictogram

Input:

Speech waveform.

Output:

Predictogram in matrix KPM

```
PREDICTOGRAM()
1  n ← 1
2  while there are more frames
3     Multiply the frame by the Hamming Window
4     Compute the DFT frame
5    for b ← 1 to 19
6       Compute KP^1 from the components of the DFT belonging to band b
7       Store result in KPM[n][b] (predictability for frame n and band b)
8    n ← n + 1
9  return KPM
```

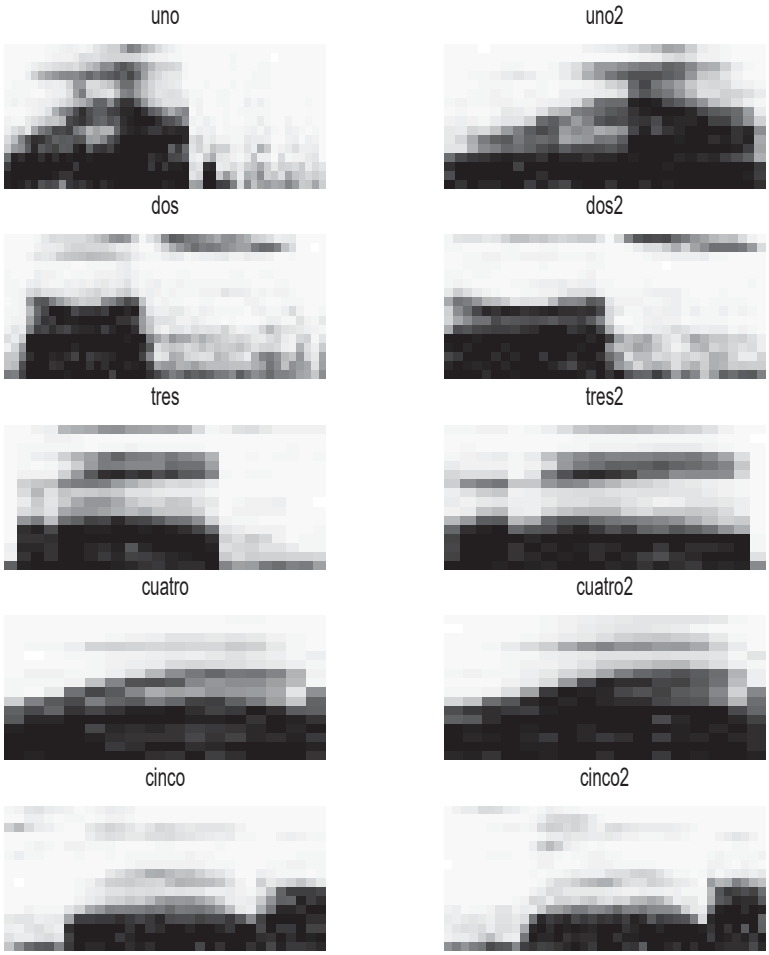


Figure B.1: Predictograms of two utterances of 5 spanish digits spoken with a Mexican accent

Figure B.1 shows the predictograms of two utterances of each word from the set of Spanish words: {“uno”, “dos”, “tres”, “cuatro”, “cinco”} spoken with a Mexican accent.

Appendix C

Gaussianity

Negentropy is a measure of the Gaussianity of a probability density function (PDF). If a signal is Gaussian it is considered to have a normal distribution and the Negentropy vanishes. The Negentropy states how far a PDF is from being normal, it is defined as:

$$J(p_x) = H(\phi_x) - H(p_x) \tag{C.1}$$

Where:
$J(p_x)$ is the Negentropy of the PDF p_x
$H(p_x)$ is the entropy of the PDF p_x
ϕ_x is the Gaussian density with the same mean and variance as p_x
$H(\phi_x)$ is the entropy of the PDF ϕ_x

To verify that the spectral coefficients of the critical bands according to the Bark scale follow a gaussian distribution, the negentropy of 1000 randomly selected frames of audio was computed. For each frame the Negentropy was computed as in the block diagram of Figure C.1

To build histograms some precautions were observed:

- The dynamic range of the whole spectrum was determined.

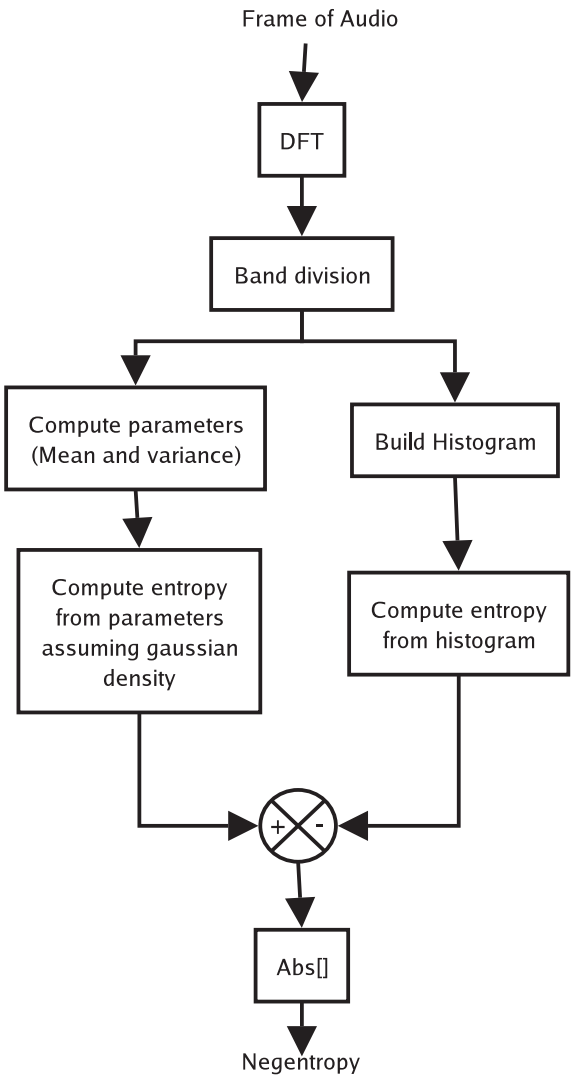


Figure C.1: Block diagram for computing Negentropy

- The number of bins in the histogram was fixed to 5 % of the number of spectral coefficients in the band.
- The width of each bin equals the dynamic range of the whole spectrum divided by the number of bins.

The Negentropy values obtained ranged between 0 and 1.1092, and since the entropy values ranged from 0 to 14.8051, then assuming gaussianity implies an error of 7.5 %. Figure C.2 shows two versions of the curve $BSES_{10}$ (Entropy vs time for the tenth critical band) for a segment of audio corresponding to 100 frames (37 seconds). The upper curve was generated assuming gaussianity and the lower curve was generated using histograms. The fact that these curves are not identical implies that the spectral coefficients corresponding to a single band are not strictly gaussian. In conclusion, since assuming gaussianity and building histograms produce different entropy values, the method for extracting the MBSES of the songs in the collection must be the same method used at extracting the MBSES for a querying song. These methods are not equivalent.

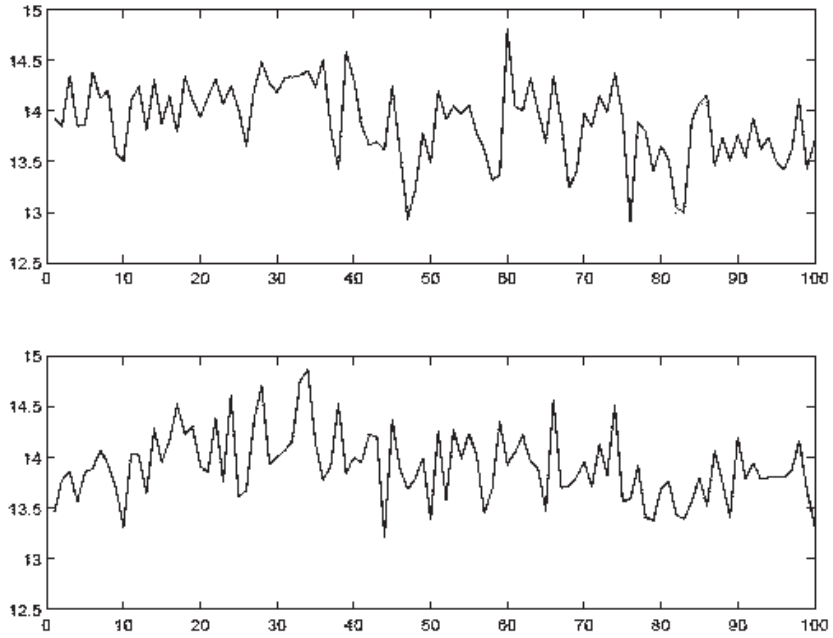


Figure C.2: $SE_{10}(t)$ obtained assuming gaussianity (top) and using histograms (down)

Appendix D

Use of Multi-Band Spectral Entropy in Isolated Word Recognition

To find out whether spectrograms would be of any use in Automatic Speech Recognition (ASR), the following experiment was designed:

Twenty one students read 34 words 4 times each, all those utterances were recorded to make a collection of 2772 audio files (i.e. wav files). The list of words consists of the ten digits in spanish (i.e. "cero", "uno",...,and "nueve", with a Mexican accent) as well as the names of the 24 greek letters: $\alpha, \beta, \gamma, \delta,\ \epsilon, \zeta, \eta, \theta,\ \iota, \kappa, \lambda, \mu,\ \nu, \xi, o, \pi, \rho,\ \sigma, \tau, \upsilon, \phi,\ \chi, \psi, \omega,$ (i.e. "alpha", "beta", "gamma", etc.). The Spectral entropy was determined only for the lower nineteen critical bands since human speech does not have spectral components above 4 kHz. The frame size was 50 ms and the frames were overlapped 75 % so that a vector of 19 spectral entropies was obtained every 12.5 ms. The sequence of such vectors make the speech entropygrams. Figure D.1 shows the entropygrams two utterances of each word from the set of Spanish words: {"uno", "dos", "tres", "cuatro", "cinco"} spoken with a Mexican accent.

The Linear Prediction Coding (LPC) coefficients [Rabiner78], and Energy-spectrograms are among the most commonly used features in Automatic Speech Recognition (ASR). They

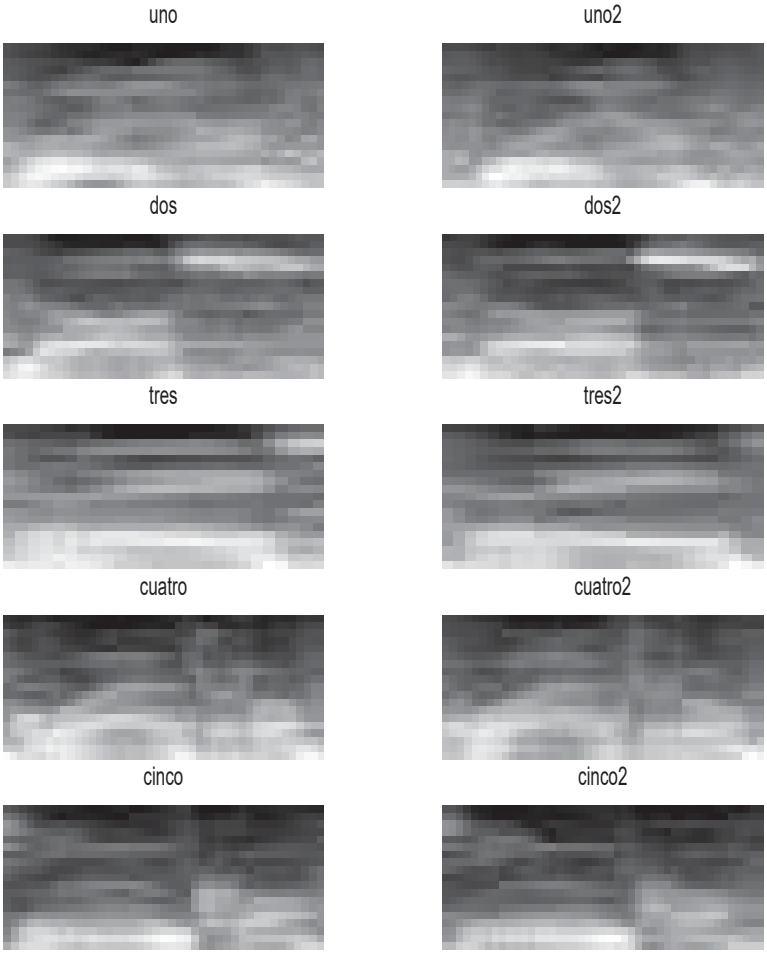


Figure D.1: Entropygrams of two utterances of 5 spanish digits spoken with mexican pronunciation

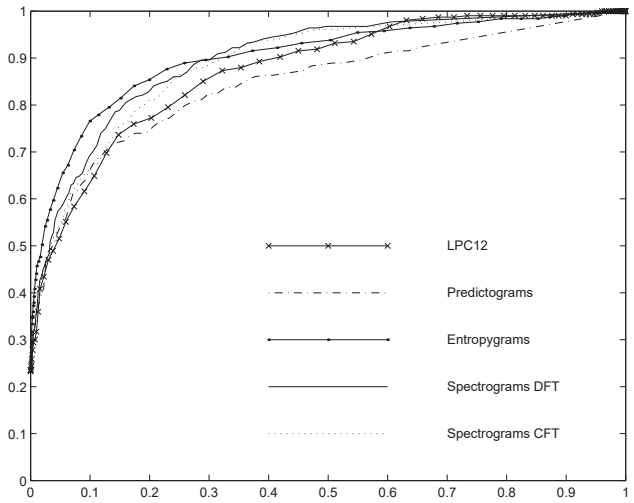


Figure D.2: ROC curves that resulted from the experiments on isolated word recognition

were therefore included in our tests. Spectrograms determined with the use of the Continuous Fourier Transform (CFT) as in [Campos92] were also included in the experiments. Finally predictograms were also considered as candidate features since predictability has been successful as the key feature of images in the field of *image registration* [Gomez-García06]. The features of every utterance was compared with those of every other utterance using the DTW distance for that matter. Figure D.2 shows the ROC curves that resulted from this experiments; entropygrams show a better accuracy rate than the other features since its ROC curve has a greater AUC (Area Under Curve).