TU/e technische universiteit eindhoven

Faculty of Electrical Engineering
Section Design Technology For Electronic Systems (ICS/ES)
ICS-ES 801

Master's Thesis

AUDIO FINGERPRINTING.

"A New Technology To Identify Music".

J.A. Haitsma

Coach:	Prof.dr. A.A.C.M. Kalker (Philips Research)
Supervisor:	prof.dr.ir. R.H.J.M. Otten
Date:	August 2002

Nat.Lab. Unclassified Report 2002/824

Date of issue: 13-08-2002

Audio Fingerprinting

"A New Technology To Identify Music"

J.A. Haitsma

Authors' address data: J.A. Haitsma WY701;Jaap.Haitsma@philips.com

Unclassified Report:	2002/824
Title:	Audio Fingerprinting "A New Technology To Identify Music"
Author(s):	J.A. Haitsma

Part of project: Audio Identification Technologies

Customer: Philips Research

Keywords: Audio Fingerprinting, Audio Recognition, Audio Identification, Proximity Searching, Database

Abstract: This report describes the core of the audio fingerprinting technology that was developed at Philips Research. An audio fingerprint is a unique code that uniquely identifies a segment of music like a human fingerprint identifies a human being.

Audio fingerprinting can be used for several applications:

- **Broadcast monitoring** to automatically generate playlists of radio stations
- **Connected audio:** users can for instance identify a song by pushing a button on their radio or by dialing a service number and letting their mobile phone listen to the song.
- **Filtering technology for P2P networks:** songs on the network can be identified irrespective of their metadata. Therefore copyrighted songs can for instance be blocked from the network.
- **Automatic Music Library Organization:** digital music archives with inconsistent, missing or erroneous meta-data can be automatically labeled with the correct meta-data

An audio fingerprint system basically consists of two parts: a fingerprint extraction algorithm and an algorithm to efficiently search such a fingerprint in a fingerprint database containing the fingerprints of many recordings. Fingerprint extraction is based on extracting bit sequences of 32 bits, so called sub-fingerprints, for every 11.8 ms of audio. These sub-fingerprints are obtained from energy differences along both the time and the frequency axis. A sequence of 256 sub-fingerprints (a fingerprint block), which corresponds to 3 seconds of audio, contains enough information to enable reliable identification.

The objective of the fingerprint search algorithm is to find as efficiently as possible a matching fingerprint block in the fingerprint database. This is not a trivial task, because a fingerprint block will in general not have a bit exact copy in the database. This is due to the fact that a fingerprint block that has to be identified is generally taken from an audio excerpt that has been processed (intentionally or unintentionally). Therefore a novel algorithm is devised, which uses the fact that sub-fingerprints sometimes do not contain any bit errors. In case they do, a more sophisticated algorithm can be used, which uses reliability information of the extracted fingerprint bits.

Conclusions:

The described audio fingerprint extraction algorithm is highly robust against audio signal degradation. Furthermore it is very reliable and needs only 3 seconds of audio to identify a song.

The fingerprint search algorithm is highly efficient. Both the extraction and search algorithm have been implemented and show that they enable all the mentioned applications with only limited hardware needs.

Future research will focus on investigating other fingerprint extraction methods and optimizing the current fingerprint search algorithm.

Table Of Contents

1 Introduction

Fingerprint systems are over one hundred years old. In 1893 Sir Francis Galton was the first to "prove" that no two fingerprints of human beings were alike. Approximately 10 years later Scotland Yard accepted a system designed by Sir Edward Henry for identifying fingerprints of people. This system relies on the pattern of dermal ridges on the fingertips and still forms the basis of all "human" fingerprinting techniques of today. This type of forensic "human" fingerprinting system has however existed for longer than a century, as 2000 years ago Chinese emperors were already using thumbprints to sign important documents. The implication is that already those emperors (or at least their administrative servants) realized that every fingerprint was unique. Conceptually a fingerprint can be seen as a "human" summary or signature that is unique for every human being. It is important to note that a human fingerprint differs from a textual summary in that it does not allow the reconstruction of other aspects of the original. For example, a human fingerprint does not convey any information about the color of the person's hair or eyes.

Recent years have seen a growing scientific and industrial interest in computing fingerprints of multimedia objects [1][2][3][4][5][6]. The growing industrial interest is shown among others by a large number of (start-up) companies [7][8][9][10][11][12][13] and the recent request for information on audio fingerprinting technologies by the International Federation of the Phonographic Industry (IFPI) and the Recording Industry Association of America (RIAA) [14].

The prime objective of multimedia fingerprinting is an efficient mechanism to establish the perceptual equality of two multimedia objects: not by comparing the (typically large) objects themselves, but by comparing the associated fingerprints (small by design). In most systems using fingerprinting technology, the fingerprints of a large number of multimedia objects, along with their associated meta-data (e.g. name of artist, title and album) are stored in a database. The fingerprints serve as an index to the meta-data. The meta-data of unidentified multimedia content are then retrieved by computing a fingerprint and using this as a query in the fingerprint/meta-data database. The advantage of using fingerprints instead of the multimedia content itself is three-fold:

1. Reduced memory/storage requirements as fingerprints are relatively small;

2. Efficient comparison as perceptual irrelevancies have already been removed from fingerprints;

3. Efficient searching as the dataset to be searched is smaller.

As can be concluded from above, a fingerprint system generally consists of two components: a method to extract fingerprints and a method to efficiently search for matching fingerprints in a fingerprint database.

This report describes the core audio fingerprinting algorithms which suitable for a large

number of applications. After defining the concept of an audio fingerprint in Section 2 and elaborating on possible applications in Section 3, we focus on the technical aspects of the proposed audio fingerprinting system. Fingerprint extraction is described in Section 4 and fingerprint searching in Section 5.

The described algorithms are also implemented in software, but this is beyond the scope of this report. The fingerprint search algorithm is now for example running on a cluster of 10 Linux PC's. It can store the fingerprints of approximately 500,000 songs and handle 250 fingerprint identification requests in parallel. With this Linux cluster it is for example possible to identify the audio of 250 radio stations in real-time. Currently there is also a Philips telephone number to which you can call with your mobile phone when you want to identify a song. After holding the mobile phone for a few seconds in front of the loudspeaker, you receive an SMS with the name and artist of the song.

2 Audio Fingerprinting Concepts

2.1 Audio Fingerprint Definition

Recall that an audio fingerprint can be seen as a short summary of an audio object. Therefore a fingerprint function F should map an audio object X, consisting of a large number of bits, to a fingerprint consisting of only a limited number of bits.

Here we can draw an analogy with so-called hash functions[1], which are well known in cryptography. A cryptographic hash function H maps an (usually large) object X to a (usually small) hash value (a.k.a. message digest). A cryptographic hash function allows comparison of two large objects X and Y, by just comparing their respective hash values $H(X)$ and $H(Y)$. Strict mathematical equality of the latter pair implies equality of the former, with only a very low probability of error. For a properly designed cryptographic hash function this probability is 2^{-n}, where n equals the number of bits of the hash value. Using cryptographic hash functions, an efficient method exists to check whether or not a particular data item X is contained in a given and large data set $Y=\{Y_i\}$. Instead of storing and comparing with all of the data in Y, it is sufficient to store the set of hash values $\{h_i = H(Y_i)\}$, and to compare $H(X)$ with this set of hash values.

At first one might think that cryptographic hash functions are a good candidate for fingerprint functions. However recall from the introduction that, instead of strict mathematical equality, we are interested in perceptual similarity. For example, an original CD quality version of 'Rolling Stones – Angie' and an MP3 version at 128Kb/s sound the same to the human auditory system, but their waveforms can be quite different. Although the two versions are perceptually similar they are mathematically quite different. Therefore cryptographic hash functions cannot decide upon perceptual equality of these two versions. Even worse, cryptographic hash functions are typically bit-sensitive: a single bit of difference in the original object results in a completely different hash value.

Another valid question the reader might ask is: "Is it not possible to design a fingerprint function that produces mathematically equal fingerprints for perceptually similar objects?" The question is valid, but the answer is that such a modeling of perceptual similarity is fundamentally not possible. To be more precise: it is a known fact that perceptual similarity is not transitive. Perceptual similarity of a pair of objects X and Y and of another pair of objects Y and Z does not necessarily imply the perceptual similarity of objects X and Z. However modeling perceptual similarity by mathematical equality of fingerprints would lead to such a relationship.

Given the above arguments, we propose to construct a fingerprint function in such a way that perceptual similar audio objects result in similar fingerprints. Furthermore, in order to be able discriminate between different audio objects, there must be a very high

[1] In the literature fingerprinting is sometimes also referred to as robust or perceptual hashing[5].

probability that dissimilar audio objects result in dissimilar fingerprints. More mathematically, for a properly designed fingerprint function F, there should be a threshold T such that with very high probability $||F(X)-F(Y)|| \leq T$ if objects X and Y are similar and $||F(X)-F(Y)|| > T$ when they are dissimilar.

2.2 Audio Fingerprint System Parameters

Having a proper definition of an audio fingerprint we now focus on the different parameters of an audio fingerprint system. The main parameters are:

- **Robustness**: can an audio clip still be identified after severe signal degradation? In order to achieve high robustness the fingerprint should be based on perceptual features that are invariant (at least to a certain degree) with respect to signal degradations. Preferably, severely degraded audio still leads to very similar fingerprints. The false negative rate is generally used to express the robustness. A false negative occurs when the fingerprints of perceptually similar audio clips are too different to lead to a positive match.

- **Reliability**: how often is a song incorrectly identified? E.g. "Rolling Stones – Angie" being identified as "Beatles – Yesterday". The rate at which this occurs is usually referred to as the false positive rate.

- **Fingerprint size**: how much storage is needed for a fingerprint? To enable fast searching, fingerprints are usually stored in RAM memory. Therefore the fingerprint size, usually expressed in bits per second or bits per song, determines to a large degree the memory resources that are needed for a fingerprint database server.

- **Granularity**: how many seconds of audio is needed to identify an audio clip? Granularity is a parameter that can depend on the application. In some applications the whole song can be used for identification, in others one prefers to identify a song with only a short excerpt of audio.

- **Search speed and scalability:** how long does it take to find a fingerprint in a fingerprint database? What if the database contains thousands and thousands of songs? For the commercial deployment of audio fingerprint systems, search speed and scalability are a key parameter. Search speed should be in the order of milliseconds for a database containing over 100,000 songs using only limited computing resources (e.g. a few high-end PC's).

These five basic parameters have a large impact on each other. For instance, if one wants a lower granularity, one needs to extract a larger fingerprint to obtain the same reliability. This is due to the fact that the false positive rate is inversely related to the fingerprint size. Another example: search speed generally increases when one designs a more robust fingerprint. This is due to the fact that a fingerprint search is a proximity search. I.e. a similar (or the most similar) fingerprint has to be found. Conceptually a fingerprint can be seen as a point in a very high dimensional fingerprint space. This means that finding

the most similar fingerprint implies finding the fingerprint point in the database that is nearest to the fingerprint (point) we want to identify. If the features are more robust the nearest point in the database will in general be nearer.

3 Applications

In this section we elaborate on a number of applications for audio fingerprinting.

3.1 Broadcast Monitoring

Broadcast monitoring is probably the most well known application for audio fingerprinting [2][3][4][5][12][13]. It refers to the automatic playlist generation of radio, television or web broadcasts for, among others, purposes of royalty collection, program verification, advertisement verification and people metering. Currently broadcast monitoring is still a manual process: i.e. organizations interested in playlists, such as performance rights organizations, currently have "real" people listening to broadcasts and filling out scorecards.

A large-scale broadcast monitoring system based on fingerprinting consists of several monitoring sites and a central site where the fingerprint server is located. At the monitoring sites fingerprints are extracted from all the (local) broadcast channels. The central site collects the fingerprints from the monitoring sites. Subsequently, the fingerprint server, containing a huge fingerprint database, produces the playlists of all the broadcast channels.

3.2 Connected Audio

Connected audio is a general term for consumer applications where music is somehow connected to additional and supporting information. As an example imagine the following service: you are listening to the radio and you hear a nice song of which you would like to learn the artist and title. In order to learn this, you just dial a certain service number on your mobile phone, let the phone listen to the music for a few seconds and finally you receive a text message with the artist and title of the song. This business is actually pursued by a number of companies [10][13]. The audio signal in this application is severely degraded due to processing applied by radio stations, FM/AM transmission, the acoustical path between the loudspeaker and the microphone of the mobile phone, speech coding and finally the transmission over the mobile network. Therefore, from a technical point of view, this is a very challenging application.

Other examples of connected audio are (car) radios with an identification button or fingerprint applications "listening" to the audio streams leaving or entering a soundcard on a PC. By pushing an "info" button in the fingerprint application, the user could be directed to a page on the Internet containing information about the artist. Or by pushing a "buy" button the user would be able to buy the album on the Internet. In other words, audio fingerprinting can provide a universal linking system for audio content.

3.3 Filtering Technology for File Sharing

Filtering refers to active intervention in content distribution. The prime example for filtering technology for file sharing was Napster [15]. Starting in June 1999, users who

downloaded the Napster client could share and download a large collection of music for free. Later, due to a court case by the music industry, Napster users were forbidden to download copyrighted songs. Therefore in March 2001 Napster installed an audio filter based on file names, to block downloads of copyrighted songs. The filter was not very effective, because users started to intentionally misspell filenames. In May 2001 Napster introduced an audio fingerprinting system by Relatable [8], which aimed at filtering out copyrighted material even if it was misspelled. Owing to Napster's closure only two months later, the effectiveness of that specific fingerprint system is, to the best of the author's knowledge, not publicly known.

In a legal file sharing service one could apply a more refined scheme than just filtering out copyrighted material. One could think of a scheme with free music, different kinds of premium music (accessible to those with a proper subscription) and forbidden music.
Although from a consumer standpoint, audio filtering could be viewed as a negative technology, there are also a number of potential benefits to the consumer. Firstly it can organize music song titles in search results in a consistent way by using the reliable meta-data of the fingerprint database. Secondly, fingerprinting can guarantee that what is downloaded is actually what it says it is.

3.4 Automatic Music Library Organization

Nowadays many PC users have a music library containing several hundred, sometimes even thousands, of songs. The music is generally stored in compressed format (usually MP3) on their hard-drives. When these songs are obtained from different sources, such as ripping from a CD or downloading from file sharing networks, these libraries are often not well organized. Meta-data is often inconsistent, incomplete and sometimes even incorrect. Assuming that the fingerprint database contains correct meta-data, audio fingerprinting can make the meta-data of the songs in the library consistent, allowing easy organization based on, for example, album or artist. For example, ID3Man [16], a tool powered by Auditude [7] fingerprinting technology is already available for tagging unlabeled or mislabeled MP3 files. A similar tool from Moodlogic [11] is available as a Winamp plug-in [17].

4 Fingerprint Extraction

4.1 Guiding Principles

Audio fingerprints intend to capture the relevant perceptual features of audio. At the same time extracting and searching fingerprints should be fast and easy, preferably with a small granularity to allow usage in highly demanding applications (e.g. mobile phone recognition). A few fundamental questions have to be addressed before starting the design and implementation of such an audio fingerprinting scheme. The most prominent question to be addressed is: what kind of features are the most suitable. A scan of the existing literature shows that the set of relevant features can be broadly divided into two classes: the class of semantic features and the class of non-semantic features. Typical elements in the former class are *genre, beats-per-minute,* and *mood.* These types of features usually have a direct interpretation, and are actually used to classify music, generate play-lists and more. The latter class consists of features that have a more mathematical nature and are difficult for humans to 'read' directly from music. A typical element in this class is *Audio Flatness* that is proposed in MPEG-7 as an audio descriptor tool [2]. For the work described in this paper we have explicitly chosen to work with non-semantic features for a number of reasons:

1. Semantic features don't always have a clear and unambiguous meaning. I.e. personal opinions differ over such classifications. Moreover, semantics may actually change over time. For example, music that was classified as *hard rock* 25 years ago may be viewed as *soft listening* today. This makes mathematical analysis difficult.

2. Semantic features are in general more difficult to compute than non-semantic features.

3. Semantic features are not universally applicable. For example, *beats-per-minute* does not typically apply to classical music.

A second question to be addressed is the representation of fingerprints. One obvious candidate is the representation as a vector of real numbers, where each component expresses the weight of a certain basic perceptual feature. A second option is to stay closer in spirit to cryptographic hash functions and represent digital fingerprints as bit-strings. For reasons of reduced search complexity we have decided in this work for the latter option. The first option would imply a similarity measure involving real additions/subtractions and depending on the similarity measure maybe even real multiplications. Fingerprints that are based on bit representations can be compared by simply counting bits. Given the expected application scenarios, we do not expect a high robustness for each and every bit in such a binary fingerprint. Therefore, in contrast to cryptographic hashes that typically have a few hundred bits at the most, we will allow fingerprints that have a few thousand bits. Fingerprints containing a large number bits allow reliable identification even if the percentage of non-matching bits is relatively high.

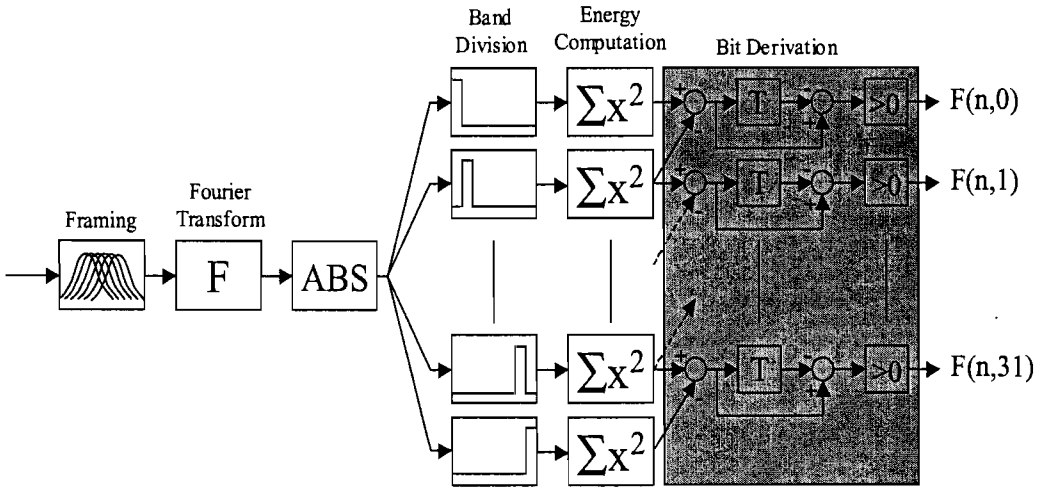

Figure 1. Overview fingerprint extraction scheme.

A final question involves the granularity of fingerprints. In the applications that we envisage there is no guarantee that the audio files that need to be identified are complete. For example, in broadcast monitoring, *any* interval of 5 seconds is a unit of music that has commercial value, and therefore may need to be identified and recognized. Also, in security applications such as file filtering on a peer-to-peer network, one would not wish that deletion of the first few seconds of an audio file would prevent identification. In this work we therefore adopt the policy of *fingerprints streams* by assigning *sub-fingerprints* to sufficiently small atomic intervals (referred to as *frames*). These sub-fingerprints might not be large enough to identify the frames themselves, but a longer interval, containing sufficiently many frames, will allow robust and reliable identification.

4.2 Extraction Algorithm

Most fingerprint extraction algorithms are based on the following approach. First the audio signal is segmented into frames. For every frame a set of features is computed. Preferably the features are chosen such that they are invariant (at least to a certain degree) to signal degradations. Features that have been proposed are well known audio features such as Fourier coefficients [4][5], Mel Frequency Cepstral Coefficients (MFFC) [18], spectral flatness [2], sharpness [2], Linear Predictive Coding (LPC) coefficients [2] and others. Also derived quantities such as derivatives, means and variances of audio features are used. Generally the extracted features are mapped into a more compact representation by using classification algorithms, such as Hidden Markov Models [3], or quantization [5]. The compact representation of a single frame will be referred to as a *sub-fingerprint*. The global fingerprint procedure converts a stream of audio into a stream of sub-fingerprints. One sub-fingerprint usually does not contain sufficient data to identify an audio clip. The basic unit that contains sufficient data to identify an audio clip (and therefore determining the granularity) will be referred to as a *fingerprint-block*.

The proposed fingerprint extraction scheme is based on this general streaming approach.

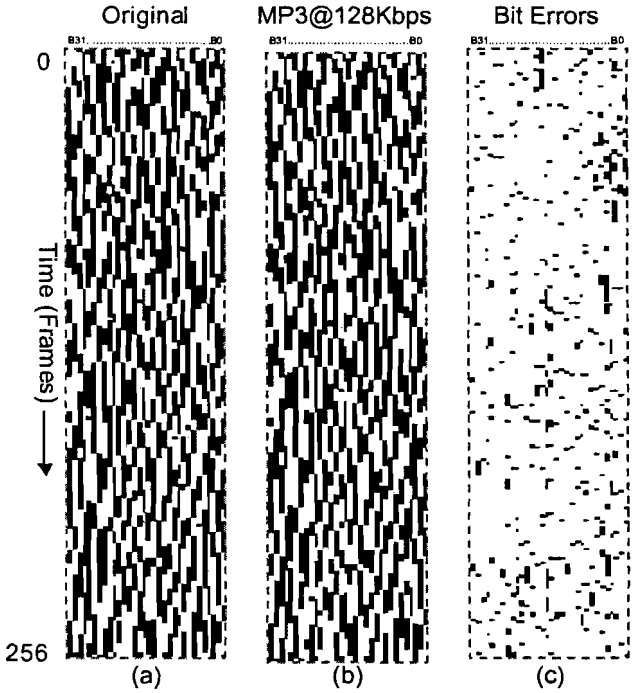

Figure 2. (a) Fingerprint block of original music clip, (b) fingerprint block of a compressed version, (c) the difference between a and b showing the bit errors in black (BER=0.078).

It extracts 32-bit sub-fingerprints for every interval of 11.8 milliseconds. A fingerprint block consists of 256 subsequent sub-fingerprints, corresponding to a granularity of only 3 seconds. An overview of the scheme is shown in Figure 1. The audio signal is first segmented into *overlapping* frames. The overlapping frames have a length of 0.37 seconds and are weighted by a Hanning window with an overlap factor of 31/32. This strategy results in the extraction of one sub-fingerprint for every 11.6 milliseconds. In the worst-case scenario the frame boundaries used during identification are 5.8 milliseconds off with respect to the boundaries used in the database of pre-computed fingerprints. The large overlap assures that even in this worst-case scenario the sub-fingerprints of the audio clip to be identified are still very similar to the sub-fingerprints of the same clip in the database. Due to the large overlap subsequent sub-fingerprints have a large similarity and are slowly varying in time. Figure 2a shows an example of an extracted fingerprint block and the slowly varying character along the time axis.

The most important perceptual audio features live in the frequency domain. Therefore a spectral representation is computed by performing a Fourier transform on every frame. Due to the sensitivity of the phase of the Fourier transform to different frame boundaries and the fact that the Human Auditory System (HAS) is relatively insensitive to phase, only the absolute value of the spectrum, i.e. the power spectral density, is retained.

In order to extract a 32-bit sub-fingerprint value for every frame, 33 non-overlapping frequency bands are selected. These bands lie in the range from 300Hz to 2000Hz (the

most relevant spectral range for the HAS) and have a logarithmic spacing. The logarithmic spacing is chosen, because it is known that the HAS operates on approximately logarithmic bands (the so-called *Bark* scale). Experimentally it was verified that the sign of energy differences (simultaneously along the time and frequency axes) is a property that is very robust to many kinds of processing. If we denote the energy of band m of frame n by $E(n,m)$ and the m-th bit of the sub-fingerprint of frame n by $F(n,m)$, the bits of the sub-fingerprint are formally defined as (see also the gray block in Figure 1, where T is a delay element):

$$F(n,m) = \begin{cases} 1 & \text{if } E(n,m) - E(n,m+1) - \big(E(n-1,m) - E(n-1,m+1)\big) > 0 \\ 0 & \text{if } E(n,m) - E(n,m+1) - \big(E(n-1,m) - E(n-1,m+1)\big) \leq 0 \end{cases} \quad (1)$$

Figure 2 shows an example of 256 subsequent 32-bit sub-fingerprints (i.e. a fingerprint block), extracted with the above scheme from a short excerpt of *"O Fortuna"* by Carl Orff. A '1' bit corresponds to a white pixel and a '0' bit to a black pixel. Figure 2a and Figure 2b show a fingerprint block from an original CD and the MP3 compressed (32Kbps) version of the same excerpt, respectively. Ideally these two figures should be identical, but due to the compression some of the bits are retrieved incorrectly. These bit errors, which are used as *the similarity measure* for our fingerprint scheme, are shown in black in Figure 2c.

The computing resources needed for the proposed algorithm are limited. Since the algorithm only takes into account frequencies below 2kHz the received audio is first down sampled to a mono audio stream with a sampling rate of 5kHz. The sub-fingerprints are designed such that they are robust against signal degradations. Therefore very simple down sample filters can be used without introducing any performance degradation. Currently 16 tap FIR filters are used. The most computationally demanding operation is the Fourier transform of every audio frame. However in the down sampled audio signal a frame has only a length of 2048 samples. If the Fourier transform is implemented as a fixed point real-valued FFT the fingerprinting algorithm has been shown to run efficiently on portable devices such as a PDA or a mobile phone.

4.3 False Positive Analysis

Two 3-second audio signals are declared similar if the Hamming distance (i.e. the number of bit errors) between the two derived fingerprint blocks is below a certain threshold T. This threshold value T directly determines the false positive rate P_f, i.e. the rate at which audio signals are incorrectly declared equal: the smaller T, the smaller the probability P_f will be. On the other hand, a small value of T will negatively effect the false negative probability P_n, i.e. the probability that two signals are 'equal', but not identified as such.

In order to analyze the choice of this threshold T, we assume that the fingerprint extraction process yields random i.i.d. (independent and identically distributed) bits. The number of bit errors will then have a binomial distribution (n,p), where n equals the number of bits extracted and p (= 0.5) is the probability that a '0' or '1' bit is extracted. Since n (= 8192 = 32 × 256) is large in our application, the binomial distribution can be

approximated by a normal distribution with a mean $\mu = np$ and standard deviation $\sigma=\sqrt{(np(1-p))}$. Given a fingerprint block F_1, the probability that a randomly selected fingerprint block F_2 has less than $T = \alpha n$ errors with respect to F_1 is given by:

$$P_f(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{(1-2\alpha)\sqrt{n}}^{\infty} e^{-x^2/2} dx = \frac{1}{2} \operatorname{erfc}\left(\frac{(1-2\alpha)}{\sqrt{2}} \sqrt{n} \right) \tag{2}$$

where α denotes the Bit Error Rate (BER).

However, in practice the sub-fingerprints have high correlation along the time axis. This correlation is due not only to the inherent time correlation in audio, but also by the large overlap of the frames used in fingerprint extraction. Higher correlation implies a larger standard deviation, as shown by the following argument.

Assume a symmetric binary source with alphabet {-1,1} such that the probability that symbol x_i and symbol x_{i+1} are the same is equals to q. Then one may easily show that

$$\mathrm{E}[x_i x_{i+k}] = a^{|k|}, \tag{3}$$

where $a = 2 \cdot q - 1$. If the source Z is the product of two such sequences X and Y, then Z is symmetric and binary and

$$\mathrm{E}[z_i z_{i+k}] = a^{2|k|}. \tag{4}$$

For N large, the standard deviation of the average $\overline{Z}_N$ over N consecutive samples of Z can be approximately described by a normal distribution with mean 0 and standard deviation equal to

$$\sqrt{\frac{1+a^2}{N(1-a^2)}}. \tag{5}$$

Translating the above back to the case of fingerprints bits, a correlation factor a between subsequent fingerprint bits implies an increase in standard deviation for the BER by a factor

$$\sqrt{\frac{1+a^2}{1-a^2}}. \tag{6}$$

To determine the distribution of the BER with real fingerprint blocks a fingerprint database of 10,000 songs was generated. Thereafter the BER of 100,000 randomly selected pairs of fingerprint blocks were determined. The standard deviation of the resulting BER distribution was measured to be 0.0148, approximately 3 times higher than the 0.0055 one would expect from random i.i.d. bits. Figure 3 shows the log Probability Density Function (PDF) of the measured BER distribution and a normal distribution with mean of 0.5 and a standard deviation of 0.0148. The PDF of the BER is a close approximation to the normal distribution. For BERs below 0.45 we observe some

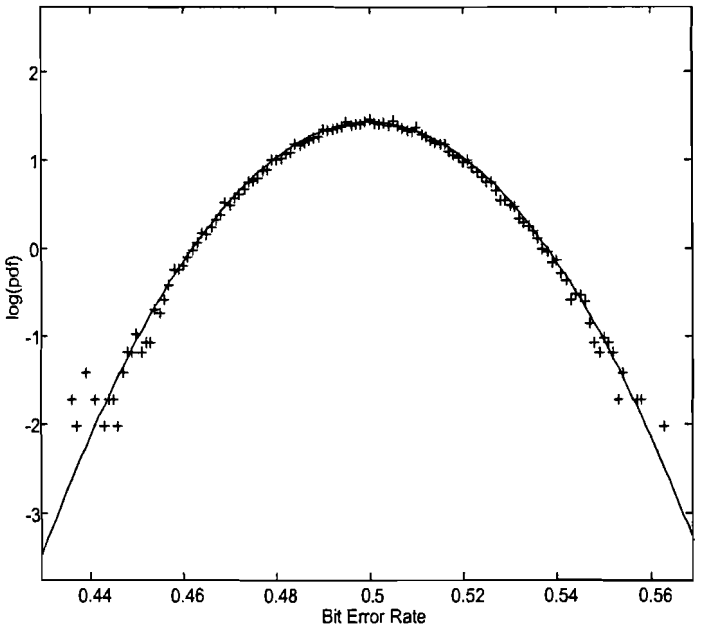

Figure 3. Comparison of the probability density function of the BER plotted as '+' and the normal distribution.

outliers, due to insufficient statistics. To incorporate the larger standard deviation of the BER distribution Formula (2) is modified by inclusion of a factor 3.

$$P_f(\alpha) = \frac{1}{2}\mathrm{erfc}\left(\frac{(1-2\alpha)}{3\sqrt{2}}\sqrt{n}\right) \tag{7}$$

The threshold for the BER used during experiments was $\alpha = 0.35$. This means that out of 8192 bits there must be less than 2867 bits in error in order to decide that the fingerprint blocks originate from the same song. Using formula (7) we arrive at a very low false positive rate of **erfc**(6.4)/2= $3.6 \cdot 10^{-20}$.

4.4 Experimental Robustness Results

In this subsection we show the experimental robustness of the proposed audio fingerprinting scheme. That is, we try to answer the question of whether or not the BER between the fingerprint block of an original and a degraded version of an audio clip remains under the threshold α (=0.35).

We selected four short audio excerpts (Stereo, 44.1kHz, 16bps) from songs that belong to different musical genres: *"O Fortuna"* by Carl Orff, *"Success has made a failure of our home"* by Sinead o'Connor, *"Say what you want"* by Texas and *"A whole lot of Rosie"* by AC/DC. All of the excerpts were subjected to the following signal degradations:

- **MP3 Encoding/Decoding** at 128 Kbps and 32 Kbps.

- **Real Media Encoding/Decoding** at 20 Kbps.
- **GSM Encoding at Full Rate** with an error-free channel and a channel with a carrier to interference (C/I) ratio of 4dB (comparable to GSM reception in a tunnel).
- **All-pass Filtering** using the system function: $H(z)=(0.81z^2-1.64z+1)/(z^2-1.64z+0.81)$.
- **Amplitude Compression** with the following compression ratios: 8.94:1 for $|A| \geq -28.6$ dB; 1.73:1 for $-46.4 \text{ dB} < |A| < -28.6$ dB; 1:1.61 for $|A| \leq -46.4$ dB.
- **Equalization** A typical10-band equalizer with the following settings:

Freq.(Hz)	31	62	125	250	500	1k	2k	4k	8k	16k
Gain(dB)	-3	+3	-3	+3	-3	+3	-3	+3	-3	+3

- **Band-pass Filtering** using a second order Butterworth filter with cut-off frequencies of 100Hz and 6000Hz.
- **Time Scale Modification** of +4% and -4% . Only the tempo changes, the pitch remains unaffected.
- **Linear Speed Change** of +1%, -1%, +2%, -2%, +3%, -3%, +4% and -4%. Both pitch and tempo change.
- **Noise Addition** with uniform white noise with a maximum magnitude of 512 quantization steps.
- **Resampling** consisting of subsequent down and up sampling to 22.05 kHz and 44.10 kHz, respectively.
- **D/A A/D Conversion** using a commercial analogue tape recorder.
- **Microphone.** The clip was played on PC loudspeakers and was recorded with the microphone in a PDA.

Thereafter the BERs between the fingerprint blocks of the original version and of all the degraded versions were determined for each audio clip. The resulting BERs are shown in Table 1. Almost all the resulting bit error rates are well below the threshold of 0.35, even for GSM encoding[2]. The only degradations that lead to a BER above threshold are large linear speed changes. Linear speed changes larger then +2.5% or –2.5% percent generally result in bit error rates higher than 0.35. This is due to misalignment of the framing (temporal misalignment) and spectral scaling (frequency misalignment). Appropriate pre-scaling (for example by exhaustive search) or an adaptation of the fingerprint extraction algorithm[3], which is more resilient to speed-ups, can solve this issue.

[2] Recall that a GSM codec is optimized for speech, not for general audio.

[3] Current research is focussing on such an algorithm.

5 Fingerprint Database

5.1 Search Algorithm

Finding extracted fingerprints in a fingerprint database is a non-trivial task. Instead of searching for a bit exact fingerprint (easy!), the *most similar* fingerprint needs to be found. We will illustrate this with some numbers based on the proposed fingerprint scheme. Consider a moderate size fingerprint database containing 10,000 songs with an average length of 5 minutes. This corresponds to approximately 250 million sub-fingerprints. To identify a fingerprint block originating from an unknown audio clip we have to find the most similar fingerprint block in the database. In other words, we have to find the position in the 250 million sub-fingerprints where the bit error rate is minimal. This is of course possible by brute force searching. However this takes 250 million fingerprint block comparisons. Using a modern PC, a rate of approximately of 200,000 fingerprint block comparisons per second can be achieved. Therefore the total search time for our example will be in the order of 20 minutes! This shows that brute force searching is not a viable solution for practical applications.

We propose to use a more efficient search algorithm. Instead of calculating the BER for every possible position in the database, such as in the brute-force search method, it is calculated for a few candidate positions only. These candidates contain with very high probability the best matching position in the database.

In the simple version of the improved search algorithm, candidate positions are generated

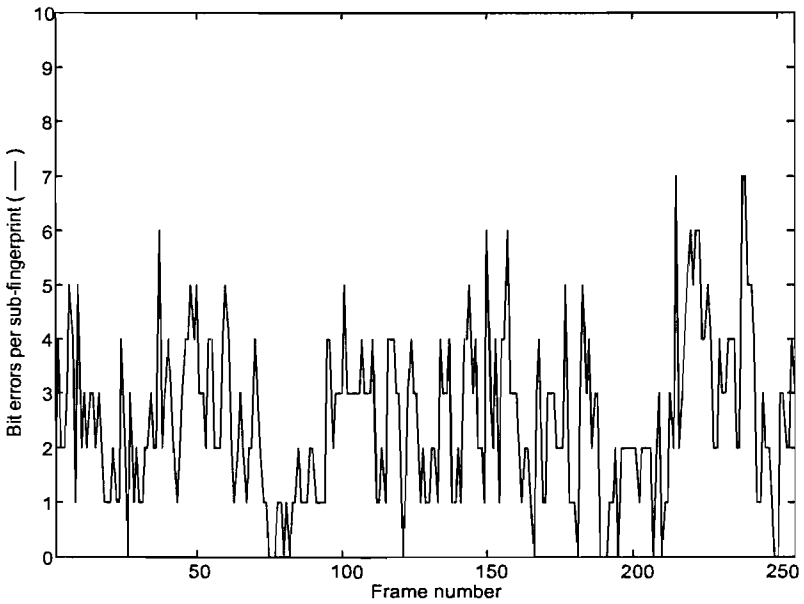

Figure 4. Bit errors per sub-fingerprint for the "MP3@ 128Kbps version" of excerpt of 'O Fortuna' by Carl Orff.

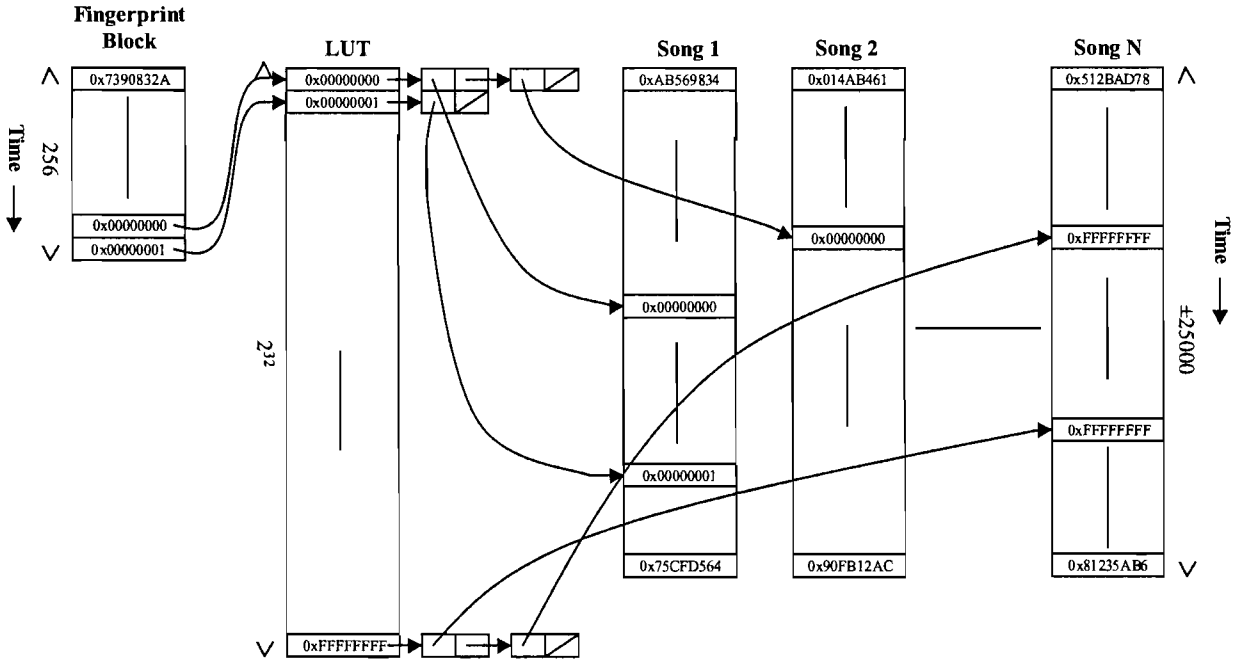

Figure 5. Fingerprint database layout.

based on the assumption that it is very likely that at least one sub-fingerprint has an exact match at the optimal position in the database [3][5]. If this assumption is valid, the only positions that need to be checked are the ones where one of the 256 sub-fingerprints of the fingerprint block query matches perfectly. To verify the validity of the assumption, the plot in Figure 4 shows the number of bit errors per sub-fingerprint for the fingerprints depicted in Figure 2. It shows that there is indeed a sub-fingerprint that does not contain any errors. Actually 17 out of the 256 sub-fingerprints are error-free. If we assume that the "original" fingerprint of Figure 2a is indeed loaded in the database, its position will be among the selected candidate positions for the "MP3@128Kbps fingerprint" of Figure 2b.

The positions in the database where a specific 32-bit sub-fingerprint is located are retrieved using the database architecture of Figure 5. The fingerprint database contains a lookup table (LUT) with all possible 32-bit sub-fingerprints as an entry. Every entry points to a list with pointers to the positions in the real fingerprint lists where the respective 32-bit sub-fingerprint are located. In practical systems with limited memory[4] a lookup table containing 2^{32} entries is often not feasible, or not practical, or both. Furthermore the lookup table will be sparsely filled, because only a limited number of songs can reside in the memory. Therefore, in practice, a hash table [19] is used instead of a lookup table.

Let us again do the calculation of the average number of fingerprint block comparisons per identification for a 10,000-song database. Since the database contains approximately

[4] For example a PC with a 32-bit Intel processor has a memory limit of 4 GB.

250 million sub-fingerprints, the average number of positions in a list will be $0.058(=250 \cdot 10^6 / 2^{32})$. If we assume that all possible sub-fingerprints are equally likely, the average number of fingerprint comparisons per identification is only 15 ($=0.058 \times 256$). However we observe in practice that, due to the non-uniform distribution of sub-fingerprints, the number of fingerprint comparisons increases roughly by a factor of 20. On average 300 comparisons are needed, yielding an average search time of 1.5 milliseconds on a modern PC. The lookup-table can be implemented in such a way that it has no impact on the search time. At the cost of a lookup-table, the proposed search algorithm is approximately a factor 800,000 times faster than the brute force approach.

The observing reader might ask: "But, what if your assumption that one of the sub-fingerprints is error-free does not hold?" The answer is that the assumption almost always holds for audio signals with "mild" audio signal degradations (See also Section 5.2). However, for heavily degraded signals the assumption is indeed not always valid. An example of a plot of the bit errors per sub-fingerprint for a fingerprint block that does not contain any error-free sub-fingerprints, is shown in Figure 6. There are however sub-fingerprints that contain only one error. So instead of only checking positions in the database where one of the 256 sub-fingerprints occurs, we can also check all the positions where sub-fingerprints occur which have a Hamming distance of one (i.e. one toggled bit) with respect to all the 256 sub-fingerprints. This will result in 33 times more fingerprint comparisons, which is still acceptable. However, if we want to cope with situations that for example the minimum number of bit errors per sub-fingerprint is three (this can occur in the mobile phone application), the number of fingerprint comparisons will increase with a factor of 5489, which leads to unacceptable search times. Note that the observed non-uniformity factor of 20 is decreasing with increasing number of bits being toggled. If for instance all 32 bits of the sub-fingerprints are used for toggling, we end up with the brute force approach again, yielding a multiplication factor of 1.

Since randomly toggling bits to generate more candidate positions results very quickly in unacceptable search times, we propose to use a different approach that uses soft decoding information. That is, we propose to estimate and use the probability that a fingerprint bit is received correctly.

The sub-fingerprints are obtained by comparing and thresholding energy differences (see bit derivation block in Figure 1). If the energy difference is very close to the threshold, it is reasonably likely that the bit was received incorrectly (an unreliable bit). On the other hand, if the energy difference is much larger than the threshold the probability of an incorrect bit is low (a reliable bit). By deriving reliability information for every bit of a sub-fingerprint, it is possible to expand a given fingerprint into a list of probable sub-fingerprints. By assuming that one of the most probable sub-fingerprints has an exact match at the optimal position in the database, the fingerprint block can be identified as before. The bits are assigned a reliability ranking from 1 to 32, where a 1 denotes the least reliable and a 32 the most reliable bit. This results in a simple way to generate a list of most probable sub-fingerprints by toggling only the most unreliable bits. More precisely, the list consists of all the sub-fingerprints that have the N most reliable bits fixed and all the others variable. If the reliability information is perfect, one expects that in the case where a sub-fingerprint has 3 bit errors, the bits with reliability 1, 2 and 3 are

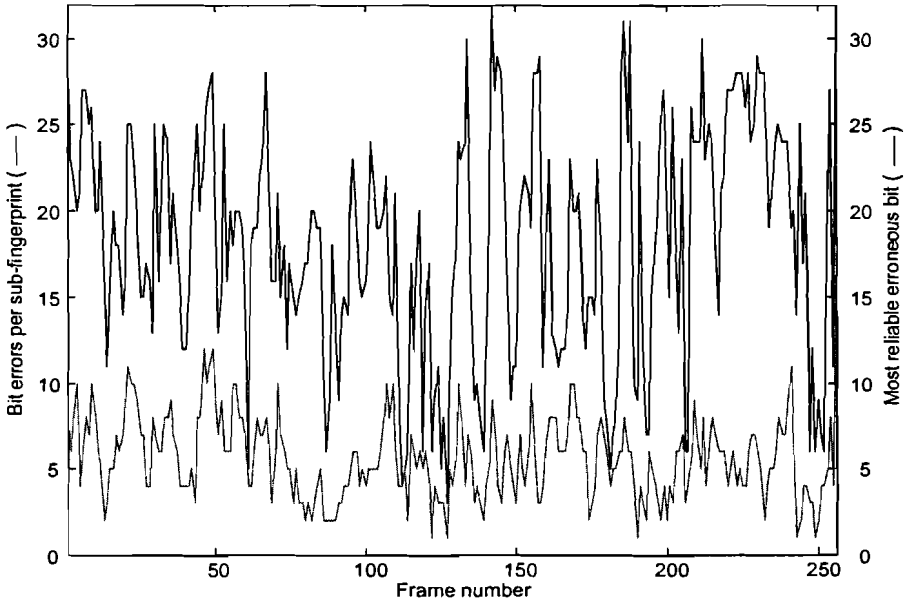

Figure 6. Bit errors per sub-fingerprint (gray line) and the reliability of the most reliable erroneous bit (black line) for the "MP3@32Kbps version" of 'O Fortuna' by Carl Orff.

erroneous. If this is the case, fingerprint blocks where the minimum number of bit errors per sub-fingerprint is 3, can be identified by generating candidate positions with only 8 ($=2^3$) sub-fingerprints per sub-fingerprint. Compared to the factor 5489 obtained when using all sub-fingerprints with a Hamming distance of 3 to generate candidate positions, this is an improvement with a factor of approximately 686.

In practice the reliability information is not perfect (e.g. it happens that a bit with a low reliability is received correctly and vice-versa), and therefore the improvements are less spectacular, but still significant. This can for example be seen from Figure 6. The minimum number of bit-errors per sub-fingerprint is one. As already mentioned before, the fingerprint block can then be identified by generating 33 times more candidate positions. Figure 6 also contains a plot of the reliability for the most reliable bit that is retrieved erroneously. The reliabilities are derived from the MP3@32Kbps version using the proposed method. We see that the first sub-fingerprint contains 8 errors. These 8 bits are not the 8 weakest bits because one of the erroneous bits has an assigned reliability of 27. Thus, the *reliability information* is not always *reliable*. However if we consider sub-fingerprint 130, which has only a single bit-error, we see that the assigned reliability of the erroneous bit is 3. Therefore this fingerprint block would have pointed to a correct location in the fingerprint database when toggling only the 3 weakest bits. Hence the song would be identified correctly.

We will finish this sub-section by again referring to Figure 5 and giving an example of how the proposed search algorithm works. The last extracted sub-fingerprint of the fingerprint block in Figure 5 is 0x00000001. First the fingerprint block is compared to

the positions in the database where sub-fingerprint 0x00000001 is located. The LUT is pointing to only one position for sub-fingerprint 0x00000001, a certain position *p* in song 1. We now calculate the BER between the 256 extracted sub-fingerprints (the fingerprint block) and the sub-fingerprint values of song 1 from position *p-255* up to position *p*. If the BER is below the threshold of 0.35, the probability is high that the extracted fingerprint block originates from song 1. However if this is not the case, either the song is not in the database or the sub-fingerprint contains an error. Let us assume the latter and that bit 0 is the least reliable bit. The next most probable candidate is then sub-fingerprint 0x00000000. Still referring to Figure 5, sub-fingerprint 0x00000000 has two possible candidate positions: one in song 1 and one in song 2. If the fingerprint block has a BER below the threshold with respect to the associated fingerprint block in song 1 or 2, then a match will be declared for song 1 or 2, respectively. If neither of the two candidate positions give a below threshold BER, either other probable sub-fingerprints are used to generate more candidate positions, or there is a switch to one of the 254 remaining sub-fingerprints where the process repeats itself. If all 256 sub-fingerprints and their most probable sub-fingerprints have been used to generate candidate positions and no match below the threshold has been found the algorithm decides that it cannot identify the song.

Table 2. Hits in the database for different kinds of signal degradations. First number indicates the hits for using only the 256 sub-fingerprints to generate candidate positions. Second number indicates hits when 1024 most probable candidates for every sub-fingerprint are also used.

Processing	Orff	Sinead	Texas	AC/DC
MP3@128Kbps	17, 170	20, 196	20, 179	19, 144
MP3@32Kbps	0, 34	10, 153	14, 147	5, 61
Real@20Kbps	2, 57	7, 110	1, 67	1, 41
GSM	1, 57	2, 95	1, 59	0, 31
GSM C/I = 4dB	0, 3	0, 12	0, 1	0, 3
All-pass filtering	157, 240	158, 256	146, 256	106, 219
Amp. Compr.	55, 191	59, 183	17, 71	44, 146
Equalization	55, 203	71, 227	33, 197	42, 148
Echo Addition	2, 36	12, 69	11, 75	4, 52
Band Pass Filter	123, 225	118, 253	103, 254	80, 214
Time Scale +4%	6, 55	7, 68	12, 68	6, 36
Time Scale –4%	17, 60	22, 77	22, 64	16, 44
Linear Speed +1%	0, 7	5, 88	0, 3	0, 8
Linear Speed -1%	3, 29	18, 170	3, 91	1, 16
Linear Speed +2%	0, 0	0, 0	0, 0	0, 0
Linear Speed –2%	0, 11	0, 64	0, 22	0, 1
Linear Speed +3%	0, 0	0, 0	0, 0	0, 0
Linear Speed –3%	0, 0	0, 11	0, 14	0, 1
Linear Speed +4%	0, 0	0, 0	0, 0	0, 0
Linear Speed –4%	0, 0	0, 0	0, 0	0, 1
Noise Addition	190, 256	178, 255	179, 255	114, 225
Resampling	255, 256	255, 256	254, 256	254, 256
D/A A/D	15, 149	38, 229	11, 109	31, 145
Microphone	0, 3	0, 21	0, 14	0, 8

5.2 Experimental Results

Table 2 shows how many of the generated candidate positions are pointing to the matching fingerprint block in the database for the same set of signal degradations used in the robustness experiments. We will refer to this number as the number of hits in the database. The number of hits has to be one or more to identify the fingerprint block and can be maximally 256 in the case that all sub-fingerprints generate a valid candidate position.

The first number in every cell is the number of hits in case only the sub-fingerprints themselves are used to generate candidate positions (i.e. no soft decoding information is used). We observe that the majority of the fingerprint blocks can be identified, because one or more hits occur. However a few fingerprint blocks, mainly originating from more severely degraded audio, such as at GSM with C/I of 4dB, do not generate any hits. This setting of the search algorithm can be used in applications, such as broadcast monitoring

and automated labeling of MP3's, where only minor degradations of the audio are expected.

The second number in every cell corresponds to the number of hits with a setting that is used to identify heavily distorted audio as, for example, in the mobile phone application. Compared to the previous setting the 1024 most probable sub-fingerprints of every sub-fingerprint are additionally used to generate candidates. In other words, the 10 least reliable bits of every sub-fingerprint are toggled to generate more candidate positions. The resulting number of hits are higher, and even the "GSM C/I = 4dB fingerprint blocks" can be identified. Most of the fingerprint blocks with large linear speed changes still do not have any hits. The BER of these blocks is already higher than the threshold and for that reason they cannot be identified even if hits occur. Furthermore one has to keep in mind that with appropriate pre-scaling, as proposed in Section 4.4, the fingerprint blocks with large linear speed changes can be identified rather easily.

5.3 Optimizing Fingerprint Storage

In this Section we introduce two ways to optimize fingerprint storage. The first, discussed in Section 5.3.1, optimizes the fingerprint storage in the fingerprint server, by using sub-sampling. By using this optimization more fingerprints (songs) can be stored inside a fingerprint database server. The second, discussed in Section 5.3.2, uses loss-less compression (i.e. Huffman coding) to compress fingerprint blocks when they are send to the server.

5.3.1 Optimizing Storage in the Fingerprint Database Server

The fingerprint extraction algorithm described in Section 4 requires approximately 100KB[5] of storage for fingerprint data for a 5-minute recording. The idea we introduce in this Section is to reduce the storage requirements while still maintaining robustness. We propose to still extract the sub-fingerprints from the recording to be identified as described in Section 4, but not to store all the sub-fingerprints in the database. Only a sub-sampled set (i.e. 1 out of every N sub-fingerprints) is stored in the database. This obviously yields a storage reduction by a factor N.

We now describe how fingerprint block matching with a sub-sampled database with a sub-sample factor N of 2 works. A full fingerprint block (no sub-sampling) of 256 sub-fingerprints is extracted from the unknown content. In the database only the sub-fingerprints of the even frames [0 2 ... 254] or odd frames [1 3 ... 255] are stored. The uncertainty about which sub-fingerprints are in the database is introduced by a possible time delay in the unknown recording. In order to determine if the extracted *full* fingerprint block of the unknown recording matches the sub-fingerprints in the database, the following procedure is followed. Firstly, the BER between both the sub-fingerprints [0 2 ... 254] and the sub-fingerprints [1 3 ... 255] and a (sub-sampled) fingerprint block in the database are calculated, respectively. Secondly, if at least one out of the two has a BER below threshold, the fingerprint block is identified. In general for a sub-sample

[5] The 100kB is computed as follows: 5(minutes) * 60(seconds) * 4(bytes per sub-fingerprint) / 0.0116(time per sub-fingerprint)

Table 3. The standard deviation σ of the BER versus the sub-sampling factor N for both a fingerprint block and a block consisting of random bits.

	Fingerprint Block			**Block consisting of random i.i.d. bits**		
Sub-sample factor N	**σ (N)**	**σ(N)/σ(1)**	**σ(N)/σ(N-1)**	**σ (N)**	**σ(N)/σ(1)**	**σ(N)/σ(N-1)**
1	0.0149	1.0000	---------	0.0055	1.0000	---------
2	0.0151	1.0096	1.0096	0.0078	1.4142	1.4142
3	0.0153	1.0273	1.0175	0.0096	1.7354	1.2271
4	0.0157	1.0497	1.0219	0.0110	2.0000	1.1524
5	0.0162	1.0870	1.0355	0.0124	2.2404	1.1202
6	0.0167	1.1187	1.0292	0.0136	2.4689	1.1019
7	0.0173	1.1561	1.0334	0.0147	2.6667	1.0801
8	0.0178	1.1922	1.0312	0.0156	2.8284	1.0607
9	0.0187	1.2513	1.0496	0.0167	3.0237	1.0690
10	0.0195	1.3043	1.0424	0.0177	3.2000	1.0583
11	0.0208	1.3948	1.0694	0.0184	3.3362	1.0426
12	0.0220	1.4739	1.0567	0.0193	3.4915	1.0465
13	0.0229	1.5325	1.0398	0.0203	3.6707	1.0513
14	0.0238	1.5961	1.0415	0.0208	3.7712	1.0274
15	0.0250	1.6766	1.0505	0.0214	3.8806	1.0290
16	0.0251	1.6831	1.0039	0.0221	4.0000	1.0308

factor of N only the sub-fingerprints [0 N 2N ...] are stored in the database and the BER is calculated for the N sub-sampled versions of fingerprint block of the recording to be identified. If 1 out of those N BER values is below a certain threshold, the recording is declared identified.

In general when a sub-sampling with a factor N is applied and the bits in a fingerprint block are random i.i.d. (independent and identically distributed), the standard deviation of the BER increases by a factor $\sqrt{N}$. This implies that either the robustness and/or the reliability is/are affected. If we keep the threshold on the BER the same the robustness is unaffected but the reliability decreases. If on the other side we decrease the threshold by the appropriate amount, the reliability stays the same but the robustness decreases.

However the bits in a fingerprint block have a large correlation along the time axis, which is introduced by the large overlap of the framing and inherent correlation in music. Therefore the standard deviation σ does not increase by a factor of $\sqrt{N}$ when a sub-sampling with a factor N is applied.

As a simple example of this phenomenon, consider a source that duplicates each symbol twice, that is a source that outputs sequences $A=[a_1\ a_1\ a_2\ a_2\ \ldots\ a_N\ a_N]$. If the values a_i are independently generated, matching a full sequence $B = [\ b_1\ b_1\ \ldots\ b_N\ b_N]$ with A or a sub-sampled sequence $[b_1\ b_2\ \ldots\ b_N]$ with a sub-sampled sequence $[a_1\ a_2\ \ldots\ a_N]$ clearly does not affect reliability of detection.

Experiments show that for small values of N the standard deviation σ does not even increase significantly at all. This is shown[6] in Table 3. For no sub-sampling (N=1) Table 3 shows that the standard deviation of the BER of a fingerprint block is approx. 3 times larger than for a block consisting of random bits (See also Section 4.3). This is due to the correlation of the sub-fingerprints along the time axis of a fingerprint block.

In the system with no sub-sampling the threshold on the BER is set to 0.35. If a sub-sampling with a factor of 4 is applied the threshold has only to be lowered to 0.342 (=0.5 – 0.15·1.0497). Therefore the decrease in robustness is insignificant, whilst the needed storage in the database has been decreased by a factor of 4. Furthermore, due to the fact that there are 4 times less sub-fingerprints in the database also approximately 4 times less candidate positions are generated. Hence the number of BER computations also decreases and consequently the time to search a fingerprint also decreases. There is however a caveat, also the probability that the valid candidate position is generated slightly decreases, simply because less candidate positions are generated. However the probability does not decrease with a factor of 4 because of the large correlation of the sub-fingerprints along the time axis. Hence in practice the advantages in storage and search speed outweigh the slightly decreased probability of generating a valid candidate position. If one insists on not having a lower probability of generating the valid candidate position, one can generate more candidate positions by generating more probable sub-fingerprints (i.e. more sub-fingerprint-bits are toggled). This will decrease the search speed again, but still keeps the advantage of less storage.

A sub-sample factor of 4 is currently used in the current fingerprinting system.

5.3.2 Fingerprint Compression

In the application scenarios discussed in Section 3 there generally is a central fingerprint database server to which fingerprint clients can connect. A device that is enabled with audio identification is equipped with a fingerprint client. When the device has to identify an excerpt of audio, the fingerprint client computes the fingerprint bits and their respective reliability information and sends these as a query to the fingerprint server. The fingerprint server then searches for a matching fingerprint in its database and subsequently sends the appropriate metadata back to the fingerprint client.

The communication between fingerprint client and server generally occurs via a TCP/IP connection on the Internet. A fingerprint query consists of 8192 fingerprint bits and 8192 bits of reliability information. The reliability information is send as a bitmap of 256 by 32 bits where a '1' bit indicates an unreliable bit and '0' bit a reliable. Therefore a fingerprint query has a total size of 2KB. When many clients connect to the server over the Internet, the required bandwidth can be substantial.

To reduce the required bandwidth the fingerprint query can be compressed on the client side and subsequently decompressed at the server side. The compression can be achieved by exploiting the correlation along the time axis of both the fingerprint bits (See also

[6] For sub-sampling factors N which are not a divider of 256 (=number of sub-fingerprints in a fingerprint block) the standard deviation of the BER was calculated on $\lfloor 256/N \rfloor$ sub-fingerprints, where $\lfloor x \rfloor$ is the floor of x

Figure 2) and the reliability bits. Furthermore in case of the reliability bits the fact that the majority of the bits will be reliable can be exploited. E.g. in a setting where the 256 most probable sub-fingerprints for every sub-fingerprint are used to generate candidate positions only 8 (2^8=256) out of every 32 bits will be a '1' bit.

The sub-fingerprints and the reliability information are compressed using the same algorithm. Therefore we only describe the algorithm for the sub-fingerprints. The first sub-fingerprint is not compressed. In order to exploit the correlation along the time axis, subsequent sub-fingerprints are first XORed with their previous sub-fingerprint. This will result in what we will call "XORed sub-fingerprints" of 32 bits of which the fast majority is '0'. Due to the fact that the statistics of the "XORed sub-fingerprints" are not very dependent on the underlying audio (this has been verified experimentally) we use Huffman coding with a static Huffman table to compress the XORed sub-fingerprints.

In order to keep the implementation simple we use a length limited Huffman code [20]. To limit the size of the Huffman encoding table to 256 entries the XORed sub-fingerprints are first divided into bytes. The Huffman codes for the XORed sub-fingerprint bytes are subsequently determined by running the algorithm of Larmore and Hirschberg [20] on the distribution of the XORed sub-fingerprint bytes. Furthermore a limitation of 16 bits for the Huffman codeword length is given as a second input of the Larmore and Hirschberg algorithm. This enables a simple Huffman decoding table with 2^{16} entries.

By using Huffman coding for both the sub-fingerprints and the reliability information experiments show that the query can be compressed by approximately a factor 3 resulting in a query size of 0.7KB.

6 Conclusions

In this report we presented a new approach to audio fingerprinting. The fingerprint extraction is based on extracting a 32-bit sub-fingerprint every 11.8 milliseconds. The sub-fingerprints are generated by looking at energy differences along the frequency and the time axes. A fingerprint block, comprising 256 subsequent sub-fingerprints, is the basic unit that is used to identify a song. The fingerprint database contains a two-phase search algorithm that is based on only performing full fingerprint comparisons at candidate positions pre-selected by a sub-fingerprint search. With reference to the parameters that were introduced in Section 2.2, the proposed system can be summarized as follows:

- **Robustness:** the fingerprints extracted are very robust. They can even be used to identify music recorded and transmitted by a mobile telephone.
- **Reliability**: in Section 4.3 we derived a model for the false positive rate, which was confirmed by experiments. By setting the threshold to 0.35 a false positive rate of $3.6 \cdot 10^{-20}$ is achieved.
- **Fingerprint size**: a 32 bit fingerprint is extracted every 11.8 ms, yielding a fingerprint size of 2.6kbit/s
- **Granularity:** a fingerprint block consisting of 256 sub-fingerprints and corresponding to 3 seconds of audio is used as the basic unit for identification. Therefore the granularity is only 3 seconds.
- **Search speed and scalability:** by using a two-phase fingerprint search algorithm a fingerprint database containing 20,000 songs and handling dozens of requests per second can run on a modern PC.

As already mentioned in Section 1 the both the fingerprint extraction and the search algorithm have been implemented in software. Extensive field testing have proven the high robustness and reliability of the fingerprint extraction algorithm and the search speed and scalability of the search algorithm.

Future research will focus on other fingerprint extraction techniques to increase robustness against large speedups and optimization of the search algorithm.

References

[1] Cheng Y., "Music Database Retrieval Based on Spectral Similarity", International Symposium on Music Information Retrieval (ISMIR) 2001, Bloomington, USA, October 2001.

[2] Allamanche E., Herre J., Hellmuth O., Bernhard Fröbach B. and Cremer M., "AudoID: Towards Content-Based Identification of Audio Material", 100th AES Convention, Amsterdam, The Netherlands, May 2001.

[3] Neuschmied H., Mayer H. and Battle E., "Identification of Audio Titles on the Internet", Proceedings of International Conference on Web Delivering of Music 2001, Florence, Italy, November 2001.

[4] Fragoulis D., Rousopoulos G., Panagopoulos T., Alexiou C. and Papaodysseus C., "On the Automated Recognition of Seriously Distorted Musical Recordings", IEEE Transactions on Signal Processing, vol.49, no.4, p.898-908, April 2001.

[5] Haitsma J., Kalker T. and Oostveen J., "Robust Audio Hashing for Content Identification, Content Based Multimedia Indexing 2001, Brescia, Italy, September 2001.

[6] Oostveen J., Kalker T. and Haitsma J., "Feature Extraction and a Database Strategy for Video Fingerprinting", VIS 2002,

[7] Auditude website <http://www.auditude.com>

[8] Relatable website <http://www.relatable.com>

[9] Audible Magic website <http://www.audiblemagic.com>

[10] Shazam website <http://www.shazamentertainment.com>

[11] Moodlogic website <http://www.moodlogic.com>

[12] Yacast website <http://www.yacast.com>

[13] Philips (audio fingerprinting) website: <http://www.research.philips.com/InformationCenter/Global/FArticleSummary.asp?lNodeId=927&channel=927&channelId=N927A2568>

[14] [RIAA-IFPI. Request for information on audio fingerprinting technologies, 2001. <http://www.ifpi.org/site-content/press/20010615.html>

[15] Napster website <http://www.napster.com>

[16] ID3Man website <http://www.id3man.com>

[17] Winamp website <http://www.winamp.com>

[18] Logan B., "Mel Frequency Cepstral Coefficients for Music Modeling", Proceeding of the International Symposium on Music Information Retrieval (ISMIR) 2000, Plymouth, USA, October 2000.

[19] Cormen T.H., Leiserson C.H., Rivest R.L., "Introduction To Algorithms", MIT Press, ISBN 0-262-53091-0, 1998

[20] L. L. Larmore and D. S. Hirschberg, "A Fast Algorithm for Optimal Length-Limited Huffman Codes", Journal of the ACM, Vol. 37, No. 3, July 1990, pp. 464-473.

Author(s): J.A. Haitsma

Title: Audio Fingerprinting
"A New Technology To Identify Music"

Distribution

Nat.Lab./PI	WB-5	
PRL	Redhill, UK	
PRB	Briarcliff Manor, USA	
LEP	Limeil-Brévannes, France	
PFL	Aachen, Germany	
CP&T	WAH	
Director:	F. Boekhorst	WY-21
Department Head	J.P. Linnartz	WY-71

Abstract

C. Conrado	Nat.Lab.	WY-71
W. Gispen	Nat.Lab.	WY-71
W. Jonker	Nat.Lab.	WY-71
F. Kamperman	Nat.Lab.	WY-71
T. Kevenaar	Nat.Lab.	WY-71
P. Koster	Nat.Lab.	WY-71
E. Kragt	Nat.Lab.	WY-71
P. Lenoir	Nat.Lab.	WY-71
G.J. Schrijen	Nat.Lab.	WY-71
P. Tuyls	Nat.Lab.	WY-71

Full report

T. Kalker	Nat.Lab.	WY-71
A. van Leest	Nat.Lab.	WY-71
M. van der Veen	Nat.Lab.	WY-71
F. Bruekers	Nat.Lab.	WY-71
J. Aprea	Nat.Lab.	WY-71
A. Lemma	Nat.Lab.	WY-71
J. Oostveen	Nat.Lab.	WY-71
K. Roberts	Nat.Lab.	WY-71
G. Langelaar	Nat.Lab.	WY-71

J. Talstra	Nat.Lab.	WY-71
T. Staring	Nat.Lab.	WY-71
T. Akkermans	Nat.Lab.	WY-71
J. Seo	Nat.Lab.	WY-71
R. Barto	Nat.Lab.	WY-71
E. Bos	Nat.Lab.	WY-71
S. Schenk	Nat.Lab.	WY-71
E. Rietjens	Nat.Lab.	WY-71
H. van Zeeland	Nat.Lab.	WY-71
S. Schimmel	Nat.Lab.	WY-71
E. de Ruijter	Nat.Lab.	WY-71
S. Egner	Nat.Lab.	WY-21
R. Collier	Nat.Lab.	WY-21
S. Baggen	Nat.Lab.	WY-63
A. van Luijt	Nat.Lab.	WB-55
M. Maes	IP&S	SFF-8
R. Maandonks	DN	WDB-3
R. Asveld	DN	WDB-3
G. Miedema	DN	WDB-1
K. Middeljans	DN	WDB-3
M. Barbieri	Nat.Lab.	WY-03
I. Paulussen	Nat.Lab.	WY-03
G. Mekenkamp	Nat.Lab.	WY-03
S. Pauws	Nat.Lab.	WY-21
J. Nesvadba	Nat.Lab.	WY-03
L. Tolhuizen	Nat.Lab.	WY-63