SPEED-CHANGE RESISTANT AUDIO FINGERPRINTING USING AUTO-CORRELATION

Jaap Haitsma and Ton Kalker

Philips Research Laboratories Eindhoven
Prof. Holstlaan 4
5656 AA Eindhoven, The Netherlands
Jaap.Haitsma@philips.com, Ton.Kalker@ieee.org

ABSTRACT

At ISMIR 2002 and CBMI 2001 the authors of this paper presented a new approach to audio fingerprinting. The proposed scheme, which we will refer to as the Streaming Audio Fingerprinting (SAF) system allows a very efficient database lookup and is also very robust against many different audio processing steps, including low bit rate audio coding, noise addition and amplitude compression. However it is not inherently robust against large linear speed changes (i.e. speed changes larger than 2%) where both the pitch and the tempo change. This is a potential problem, because some radio stations speed up by a few percent. In this paper we discuss a modification of the originally proposed fingerprinting algorithm, which is robust against large linear speed changes. The proposed modification has negligible effect on other aspects, such as robustness and reliability.

1. INTRODUCTION

In recent years we have seen a growing scientific [1][2][3][4][5] and industrial [6][7][8] interest in automatic identification of audio. The technology is generally referred to as audio fingerprinting. Although other names have also been used such as robust/perceptual hashing [2] and robust signatures.

Audio fingerprint systems can be used for a number of interesting applications such as automatic monitoring of radio broadcasts, identification of unknown songs using a mobile phone [6][7], filtering [1] for legal Napster-like services or automatically restoring ID3 tags of MP3 files [8][9].

The five main parameters of a fingerprint system are: (i) robustness, determining how severely an audio clip can be processed before it cannot be recognized anymore, (ii) reliability, expressing the probability that an audio clip is falsely identified, (iii) granularity, determining the minimal segment of audio needed for identification, (iv) fingerprint size, the number of bits of a fingerprint and (v) search speed determining how fast a fingerprint can be looked up in a large database. The authors of this paper showed that their system [1][2] performs well on all these five parameters. Although the system is very robust against almost all common audio processing steps, it is not inherently robust against large linear speed changes. Note, that we define a linear speed change as a speed change where both the pitch and the tempo change. E.g. an audio file that has been sampled at 44.1kHz but is played back at a sampling rate of for instance 44.2 kHz results in a linear speed change. In practice such a speed changes do occur in radio broadcasts. Broadcasters supposedly do this for two reasons. Firstly a shorter duration of songs enables the broadcast of more commercials. Secondly, the increased beat seems to be preferred by the audience, persuading them to stay with the selected station.

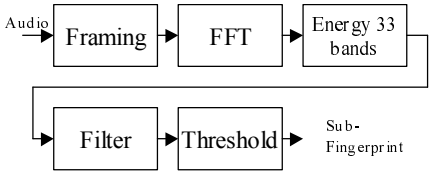

Figure 1 Overview existing fingerprint algorithm.

As the SAF algorithm presented in [1][2] is robust against speed changes of approximately 2%, we originally proposed to solve larger speed changes by storing the fingerprint at multiple speeds in the database or extracting the fingerprint query at multiple speeds and then perform multiple queries on the database. The main disadvantage of these methods is that either the storage requirement increases or the search speed decreases by several factors. In this paper we describe a modification of the existing algorithm that does not have this disadvantage, as it is inherently robust against larger speed changes.

2. EXISTING ALGORITHM

In this section we describe the existing SAF fingerprint extraction algorithm and explain why it is not robust against large linear speed changes. For a more detailed explanation the reader is referred to [1]. It is based on a general approach of computing a small fingerprint, i.e. a *sub-fingerprint*, for a time interval of in the order of 10ms. A sub-fingerprint, typically 32 bits large, does not contain sufficient information to identify an audio signal by itself, but a sequence of sub-fingerprints, which we refer to as a *fingerprint block*, does. A fingerprint block typically contains 256 sub-fingerprints (corresponding to approx. 3 seconds of audio) and consequently 8192 fingerprint bits.

To identify an audio signal a fingerprint block is extracted from the audio signal and subsequently sent to the fingerprint database as a query. The fingerprint database then responds with the metadata of the best matching fingerprint block in the database provided that the match is reliable enough. The matching criterion that we use is the Bit Error Rate (BER). If the BER between the query and the block in the database is below a

pre-determined threshold the match is said to be reliable. In [1] it is derived that a match can be qualified as reliable when the BER is below 35%. Therefore, for a reliable match, 2867 fingerprint bits out of the total of 8192 bits of a fingerprint block can be in error.

An overview of the algorithm is shown in Figure 1. First the audio is divided into frames, which have a length of approx. 0.37 seconds and a very large overlap of 31/32. This results in one frame and therefore also one sub-fingerprint for every 11.6 ms of audio. A spectral representation is obtained by applying a Fast Fourier Transform on every frame. From the resulting spectrum a vector of 33 energies is computed by determining the energy in 33 logarithmically spaced bands. The bands typically lie in the range of 300Hz to 2000Hz, which is perceptually the most relevant range. If we now put the energy vectors of subsequent frames as rows in a matrix, we obtain a spectrogram. The spectrogram is subsequently filtered by a simple 2D (time-frequency) filter with kernel F given by

$$F = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (1)$$

This results in values that represent energy differences along both the time and frequency axis. Finally the obtained energy differences values are converted to bits (i.e. one 32 bit sub-fingerprint per frame) by simple thresholding. If a value is larger than zero it is assigned a '1' bit, otherwise a '0' bit.

This scheme has proven to robust to most common audio processing steps, except large linear speed changes. This is due to the fact that speed changes cause misalignment along both the time and frequency axis.

Considering first the time misalignment, an audio excerpt subjected to a speed change of (for example) +2% causes the 250th sub-fingerprint of this excerpt to be extracted at the position of the 255th sub-fingerprint of the original excerpt. Fortunately, in order to be robust to shifting, the sub-fingerprints are constructed such that they possess a large correlation along the time-axis [1][2]. The large correlation is introduced by the large overlap of the framing. Therefore the BER between the original excerpt and the same excerpt with a speed change does not increase dramatically due to the temporal misalignment. This is demonstrated by experimental results. Figure 3 shows that the existing scheme is robust to large time scale modifications[1]. The main problem introduced by large speed changes seems to be the frequency misalignment. The above example of 2% speedup results in a scaling of the frequency axis of the spectrum that is obtained with the Fourier Transform. For example a tone of 500Hz then results in a tone of 510 Hz and a tone 1000 Hz results in a tone of 1020 Hz. As a result, speed change causes a shift of energy from one band to another band. The more energy shifts from one band into the next, the larger the probability that the extracted fingerprint bits (which are quantized energy differences) are erroneous.

[1] A processing step in which the tempo changes, giving temporal misalignment, but the pitch remains unaffected, i.e. no frequency misalignment.

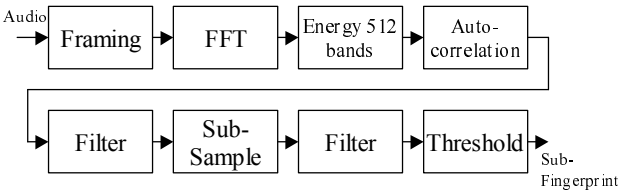

Figure 2 Overview modified algorithm

3. MODIFIED ALGORITHM

As shown in the previous section the main problem in the existing SAF algorithm seems to be the frequency misalignment caused by the speed change. In this section we present a modified version of the algorithm that is less susceptible to frequency misalignment and is based upon the auto-correlation of a densely sampled power spectrum. An overview of the modified scheme is shown in Figure 2. The first two steps (i.e. framing and FFT) are identical to the existing algorithm of Section 2. The third step is different: the energy of 512 bands instead of 33 is computed. The bands are still logarithmically spaced and in the range of 300 to 2000Hz. Thus the width of the bands is smaller. As already mentioned in Section 2, a speed-change results in a (possibly non-integer) shift of the computed energy vector. It is well known that auto-correlation is shift invariant, as shown with the following argument. Defining the auto correlation ρ_{ff} of the function $f(t)$ as:

$$\rho_{ff}(x) = \int_{-\infty}^{\infty} f(t) f(t+x) dt, \qquad (2)$$

The autocorrelation ρ_{gg} of the shifted function $g(t)=f(t+\alpha)$ is shown to be invariant by the following argument:

$$\begin{aligned} \rho_{gg}(x) &= \int_{-\infty}^{\infty} g(t) g(t+x) dt = \int_{-\infty}^{\infty} f(t+\alpha) f(t+\alpha+x) dt \\ &= \int_{-\infty}^{\infty} f(t) f(t+x) dt = \rho_{ff}(x). \end{aligned} \qquad (3)$$

Since we are not dealing with a continuous function but a discrete sequence of energies, we propose to approximate this behavior by correlating a subsequence of the energies with the complete sequence. More specifically we calculate the autocorrelation $\rho[x]$ of the energy vector $e[x]$ as follows:

$$\rho[x] = \sum_{j=1}^{M} e[K+j] e[x+j] \quad for \quad 1 \le x \le N - M \qquad (4)$$

where N denotes the length of the whole energy vector (in our case 512), M the length of the sub-sequence and K the position where the sub-sequence starts in the complete sequence. Typical settings for M and K are 64 and 96, respectively. To increase robustness, the resulting auto-correlation values are low pass filtered. The two last steps are identical to the steps of the existing algorithm. These need as input 33 values. However the low-pass filtered auto-correlation contains 512-64 = 448 values. Therefore the 448 low-pass filtered values are down-sampled to 33 values.

4. EXPERIMENTAL RESULTS

In order to assess the robustness of the modified fingerprint extraction algorithm, we repeated the experiments performed in [1] and compared the results to the existing algorithm. The experiments were performed by extracting fingerprint blocks from four audio clips belonging to different musical genres and from a set of processed versions of these clips. The set of processed versions consisted of audio coding at different bit rates, speech coding, filtering, amplitude compression, linear speed changes, time scaling (i.e. a speed change where the tempo changes but the pitch remains unaffected), echo addition, DA/AD conversion and recording with a microphone. To assess the robustness the bit error rates between the original version and their respective processed version was determined. The resulting bit error rates of all experiments were averaged over the four clips. The results of both the existing and the modified algorithm are shown in Figure 3. Experiments showed the standard deviation of the BER [1][2] of non-matching blocks of the modified algorithm is similar to the original algorithm. Therefore, for both algorithms, to be able to identify a fingerprint block the bit error rate has to be below the threshold of 0.35. The results clearly show that the robustness of both algorithms is very similar for most audio processing steps, but that the modified version is much more robust against linear speed changes. As the main focus of the modified algorithm was on improving the robustness against linear speed changes Figure 4 shows the bit error rate as a function of the speed change. It clearly shows that the modified algorithm can handle speed changes up to 6%, whilst the existing algorithm can only handle speed changes up to 2%.

It is important for any fingerprinting method that not only it results in a low BER, but also that it allows efficient searching. In [1] a search algorithm is presented that uses an index on sub-fingerprint level to search the fingerprint database for a matching fingerprint block. For all 256 sub-fingerprints in the query (recall that a fingerprint block is the typical unit for a query) a list of most probable sub-fingerprints is generated. These lists of probable sub-fingerprints are based on soft decoding information that is available during fingerprint extraction [1] and include the sub-fingerprint itself as the most probable sub-fingerprint. In order to find the best matching block in the database a match is performed at all the positions in the database where a sub-fingerprint occurs that matches one of the probable sub-fingerprints of the query. Therefore it is a prerequisite that one of the sub-fingerprints in the lists of the probable sub-fingerprints exactly matches the respective sub-fingerprint of the best matching block in the database. The positions where a certain sub-fingerprint occurs can be retrieved very efficiently by implementing the index on the sub-fingerprints as a hash-table.

One can draw an analogy with string searching over here. The analogy is to find the best matching string (fingerprint block) in a large database of characters (database of sub-fingerprints). The query string may contain erroneous characters, but for every character an estimate of the list of correct characters is available. Then the best match for the string in the database can be efficiently retrieved by only performing a match at positions in the database where either one of the characters or one of the probable characters of the string occurs. For a more detailed explanation of the search algorithm the reader is referred to [1].

In our experiments, a list of 1024 most probable candidates was created for each of the 256 sub-fingerprints of the fingerprint block. Figure 5 and Figure 6 show the number (again averaged over the four excerpts) of lists that contain a corresponding sub-fingerprint in matching fingerprint block in the database. This number is referred to as the number of database hits. Recall that in order to find the matching block in the database only one single database hit is needed to be successful. Figure 6 shows that in case of linear speed changes the number of database hits increases significantly with the newly proposed algorithm compared to the original algorithm. Figure 5 shows that the performance with respect to other common audio processing steps is similar. For most audio processing steps the number of database hits is much larger than one. Therefore the number of generated probable sub-fingerprints can be decreased significantly for most audio processing steps, which results in faster search times.

5. CONCLUSIONS

In this paper we presented a modified version of the SAF fingerprint algorithm that is more robust against linear speed changes. The robustness against speed changes is achieved by exploiting shift invariance of the auto-correlation function. Experiments show that the modified SAF algorithm is as robust as the original SAF algorithm in case of most audio processing steps. However it is far more robust in case of linear speed changes. With the modified algorithm the speed may vary between –6% and +6%. This is sufficient to handle speed changes applied in radio broadcasting (usually only a few percent).

6. REFERENCES

[1] J. Haitsma and T. Kalker, "A Highly Robust Audio Fingerprinting System," Proceedings of the International Conference on Music Information Retrieval (ISMIR) 2002, pp. 107-115, October 2002, Paris.

[2] J. Haitsma, T. Kalker, and J. Oostveen. "Robust audio hashing for content identification", Proceedings of the International Workshop on Content-Based Multimedia Indexing (CBMI) 2001, pp. 117-125, Brescia, Italy, 2001.

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

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

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

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

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

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

[9] MusicBrainz website <http://www.musicbrainz.org>

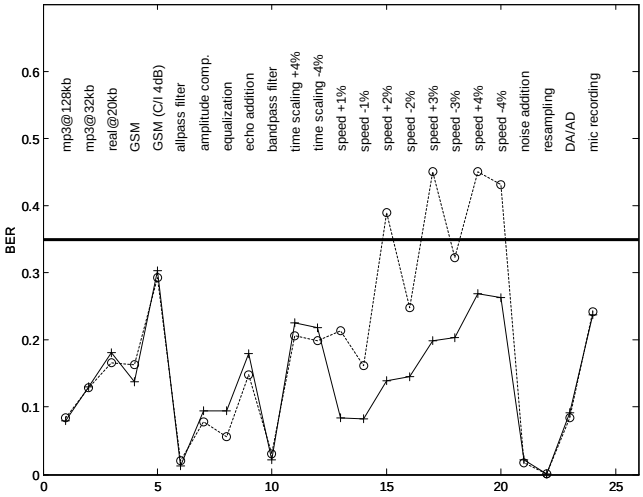

Figure 3 The resulting BER after different audio processing steps. The existing algorithm is plotted as a dotted line with 'o'. The modified algorithm as a solid line with '+'. The BER threshold for identification is depicted as thick solid line.

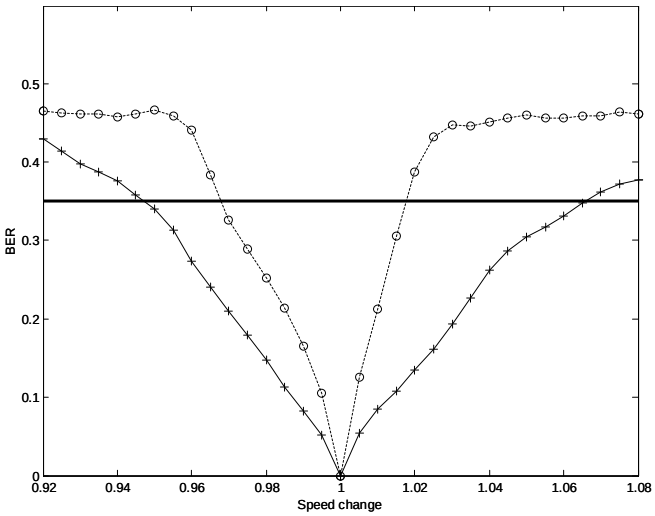

Figure 4. The resulting BER as a function of speed change. The existing algorithm is plotted as a dotted line with 'o'. The modified algorithm as a solid line with '+'. The BER threshold for identification is depicted as thick solid line.

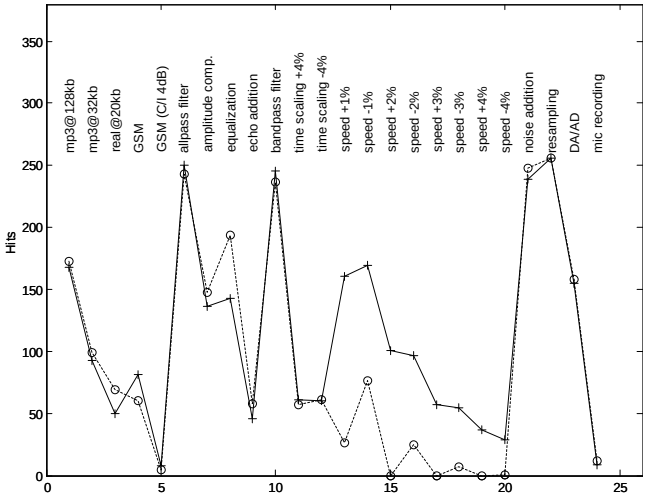

Figure 5. The number of hits in the database after different audio processing steps. The existing algorithm is plotted as a dotted line with 'o'. The modified algorithm as a solid line with '+'.

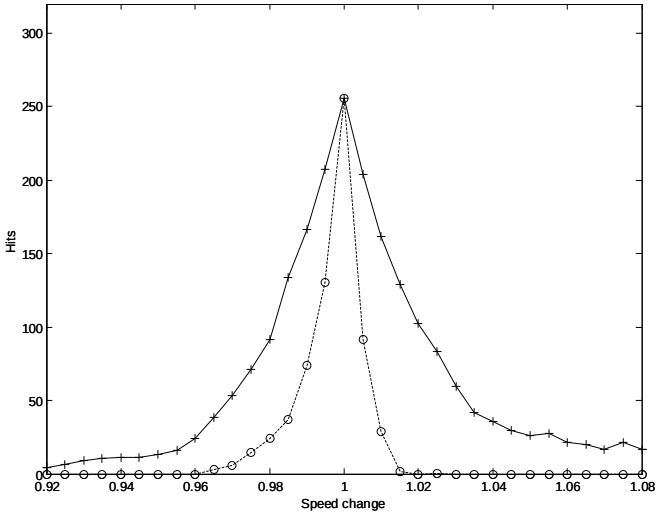

Figure 6. The number of hits in the database as a function of speed change. The existing algorithm is plotted as a dotted line with 'o'. The modified algorithm as a solid line with '+'