ROBUST AUDIO FINGERPRINTING FOR SONG IDENTIFICATION

F. Mapelli, R. Pezzano and R. Lancini

CEFRIEL – Politecnico di Milano
Via R.Fucini, 2 – 20133, Milano – ITALY
phone: +39 02 23954 209
E-mail: {rosa,mapelli}@cefriel.it

ABSTRACT

This paper presents a new algorithm for audio hashing. These kinds of techniques work analysing the signal taking into account its nature in order to extract distinguishing features robust to typical processing. These features describe in a compact way the signal and can be efficiently stored in a database. In our algorithm the features are analysed for short frames using a quantization approach. Moreover we propose a parameter for identification, which is also used to define a confidence. The confidence is used to identify the quality of the identification. The algorithm is tested under different processing operations: compression, cropping, noise addition, subsampling, stereo to mono conversion, etc. The results show that the identification can be performed also using a short excerpt of the song.

1. INTRODUCTION

The increasing number of multimedia data that can be accessed increases the interest in techniques for automatic retrieval of information. Two main classes of techniques can be identified [1]: invasive and non-invasive. While first approaches were mainly invasive – e.g. watermarking – in the last few years, non-invasive techniques for automatic recognition have received an increasing interest. Among them one of the most important techniques is hashing [2], also known as (passive) fingerprinting. The non-invasive techniques can be used in a lot of applications such as information retrieval, automatic content identification, authentication and so on.

In this paper we present a hashing technique as a method for automatic recognition of songs. Given a song or an excerpt we want to be able to identify which song it is with a good confidence. This has to be done without modifying the original signal but only analysing it. Many examples can be found in literature, see [2,3,5,6,7].

Our algorithm works in the frequency domain extracting three different features [3,5], that are proved to be robust to different operations. Robust means that they should not – so much – be affected by signal processing operation – like mp3 compression, noise addition and so on. Moreover the features have to be distinctive of the signal in order to distinguish between different songs. Features are analysed using adaptive quantization on short window of about two seconds. In this way we realize a compact representation of features that can be stored in a database.

In the identification phase we define a parameter to measure the similitude between two hashes. Moreover we define a confidence parameter starting from the similitude in order to check the reliability of the identification.

The algorithm is tested for different signal processing operations and the obtained results show that this goal can be achieved with very high confidence even if we do not have the whole song but only a little excerpt of few seconds.

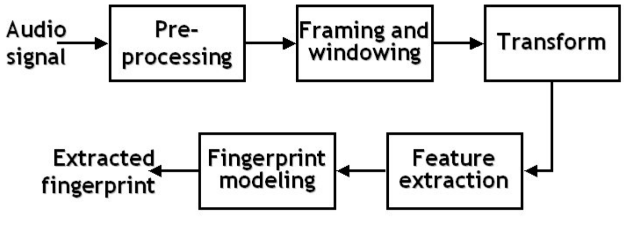

Figure 1 - Hash extraction algorithm.

2. HASH EXTRACTION PHASE

Figure 1 shows the main phases of the algorithm that are described in the following section taking into account the main parts of a hashing algorithm according to [2]. In the following paragraphs we explain the different phases.

2.1 Preprocessing

This first phase is important in order to improve the efficiency of the algorithm and to obtain a better modelling of the audio signal and features robust to audio processing. First of all the audio is converted to a mono signal, if necessary. One of the most important problems to take into account is the loss of some frequencies. The first reason could be the *resample* – for example for lower bit-rate coding – that is solved resampling the signal to 44.1kHz. The second reason is the transmission over band limited network – e.g. phone network – so we limit our analysis to the most significant frequency range: 300-3400 Hz.

2.2 Framing and windowing

In this phase the original signal is split in shorter parts called *frames*. This is important for some different reasons. Firstly the whole signal could not be available. Moreover we know that the audio signal is not stationary, but the shorter is the frame the more stationary can be considered the signal. Lastly working on short frames allows faster operation reducing computational burden. The chosen frame length is

32768 – this is a power of 2 in order to allow the use of fast algorithm for the transform.

Many signal processing operations – like shifting, time stretch and so on – can cause loss of alignment between frames in original and modified songs. This effect can be reduced introducing an overlap between frames: the frames are overlapped for 63/64 samples. To reduce discontinuities each frame is windowed using a Hamming window.

It is almost impossible to identify correctly each single frame, for this reason we have to choose the minimum identifiable shot. This shot is about 2 seconds long and includes 64 frames and is called hash window.

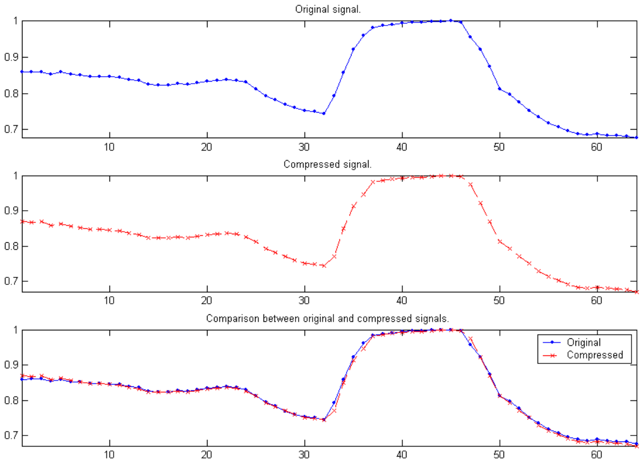

Figure 2 - Comparison of extracted feature from original and compressed versions.

2.3 Transform

To choose the transform we have to decide which parameters model the fingerprint. We are interested in three parameters: the energy of each window, the SCF (Spectral Crest Factor) and SFM (Spectral Flatness Measure). They are computable in the frequency domain: we use the FFT. For this reason in the previous paragraph we have chosen the size of the frame as a power of 2.

2.4 Features extraction

In order to extract a robust fingerprint we use the following parameters [3,4,5,6]:

$$E = \frac{1}{N}\sum_{1}^{N} \left|S(f)\right|^2 \; ;$$

$$SFM = 10\log\left(Mg\left(\left|S(f)\right|^2\right)/\, Ma\left(\left|S(f)\right|^2\right)\right);$$

$$SCF = 10\log\left(Max\left(\left|S(f)\right|^2\right)/\, Ma\left(\left|S(f)\right|^2\right)\right);$$

where *s(t)* the signal in the time domain, *S(f)* the signal in the Fourier domain, *Ma* is the arithmetic mean and *Mg* the geometrical mean. An example of the efficiency of the above parameter can be seen in Figure 2. It plots the parameter *E* computed for two different versions of the same song: original version E_O – first graph – and mp3 compressed version E_C – second graph. It is noticeable that E_O and E_C are almost similar as shown in the third graph.

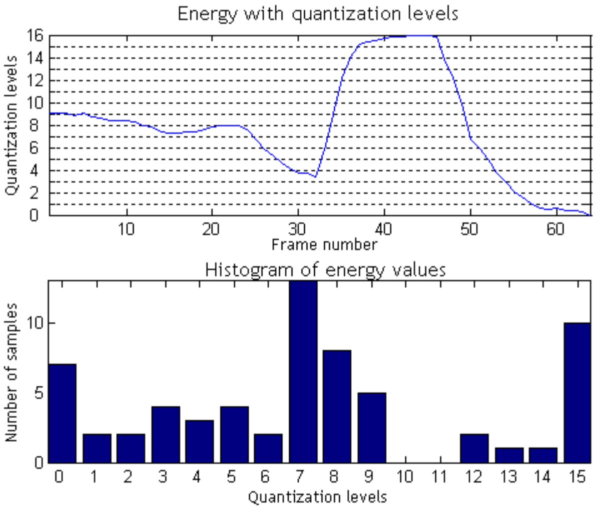

Figure 3 - Feature plot with linear quantization levels (above) and histogram of values distribution (below).

2.5 Fingerprint modelling

It is important in this phase to select the correct way to analyse the chosen parameters. We decide for a quantization approach: each parameter is quantized using 4 bit per level. In this way we reduce the "noise" due to signal processing. The problem is the choice of the quantization. Using a linear quantization we obtain a distribution for the value as in Figure 3.

The distribution varies a lot with modification on the signal. For example supposing a high peak is inserted, the quantization process tends to quantize the whole signal assigning low values. This reduces the identification capabilities of our system. This reason leads us to develop an adaptive approach for the quantization. The goal is to obtain a more uniform distribution of the histogram. The first step is to analyse the obtained values in order to choose custom – non-uniform – intervals for quantization. To achieve this we order in increasing way the values, than we take the middle and split them in two groups. Then in each group we repeat the procedure, and so on until we obtain 16 intervals. In this way we highly reduce the dependence on signal modification.

Figure 4 shows the quantization levels obtained for the same feature of Figure 3. Obviously in this case we have the same number of samples for each quantization levels, so the histogram is constant.

In this way we quantize each feature in each hash window – i.e. 64 frames. We use 4 bits for the quantized values, this means that we have a hash window – 2 seconds of song – represented by 768bits – 64 (values per hash window) * 4 (bits per values) *(3 features).

3. IDENTIFICATION PARAMETER

As the identification parameter we use the following similitude parameter compute between the two hashes we are comparing:

$$d = \frac{1}{N_Q}\sqrt{\frac{1}{N_F}\sum (H_2 - H_1)^2} \;;$$

where H_1 and H_2 are the hash values to compare, N_F=64 the number of value in each hash and N_Q=16 the number of quantization levels. Obviously the lower is d the higher is the similarity between the two hashes. By definition the value of d is in [0,1].

It is important to determine the threshold T for the identification. If $d>T$ we decide that the two compared hashes belong to different songs, otherwise if $d<T$ we have the same song. Experimentally we found that a good value for the threshold is T=0.3.

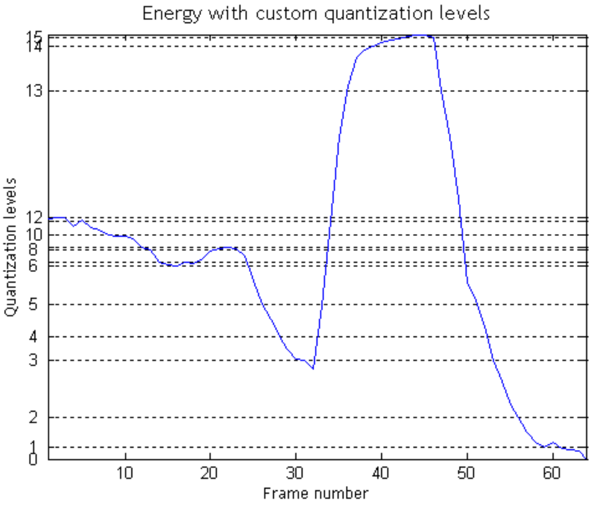

Figure 4 - Feature plot and adaptive quantization levels.

4. SONGS IDENTIFICATION

The identification process is presented in this section. The first frame – i.e. 32768 samples, about 0.74 seconds – is extracted and analysed. Then the window analysis is moved 1/64 –i.e. 512 samples – ahead on the next frame, and so on. Once analysed the first 64 frames we have a complete hash window, so we quantize the coefficients and compute the first distance in order to identify which is the song. At the first step the retrieved song is the one with the lowest d.

Then the algorithm goes on analysing next frames and windows. And the identification is performed again. We compute also the confidence of the results considering two parameters: d and the number of consecutive windows where the song has hashes below T. This allows recovering from errors. For example take the following values:

S1	0.1	0.08	0.15	0.07	0.05	0.5	0.04	0.2	0.11	0.13
S2	0.5	0.48	0.6	0.35	0.4	0.1	0.5	0.6	0.47	0.38

They represent the d for two different songs S1 and S2. It can be noticed that S1 has hash d below threshold except the 6th value, while S2 is always above threshold except the 6th value. Checking only each single hash without considering previous results we should say that we have: S1 for 5 times, than S2, than S1 for other 4 times. But checking the confidence we can say that the 6th result is S2 with very low confidence.

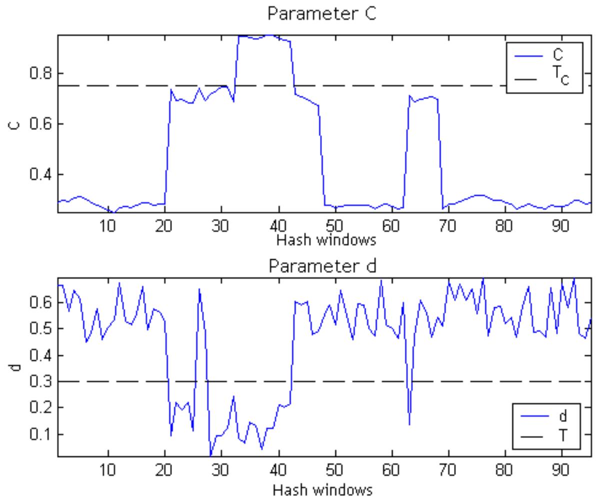

Figure 5 – Confidence C (upper graph) computed on parameter d (lower graph).

The confidence is computed as:

$$C = \frac{1}{2}\left(1 + \frac{1}{N_T}\sum_{N_T}(1 - d_T) - \frac{1}{N_{NT}}\sum_{N_{NT}}(1 - d_{NT})\right);$$

where the parameters represent:

- N_T is the number of hash values with $d \leq T$ and d_T is the value of d for these hashes;
- N_{NT}: the number of hash values with $d>T$ and d_{NT} is the value of d for these hashes.

The parameter C is in [0,1]. Typical plot of C has peaks in presence of consecutive d values below threshold, and rapidly decrease when value below T appears.

An example of C plot is in Figure 5. In the graph we report a threshold – named T_C – that represents a good experimental value to avoid false positive. A combined analysis of d and C provides the better results.

5. TEST RESULTS

A lot of different tests are performed in order to check the efficiency of our algorithm. In our tests we take different songs both of same and different genre. The reported results are intended as a mean for each window analysed – e.g. d=0.1 means that the mean of all d computed for each window of the song is 0.1. The test are performed using a database of more than 1000 songs.

First of all we test the original songs in order to verify that different songs provide different hashes. In this test we have obviously d=0 comparing the song with itself, while we have about d=0.73 comparing different songs. The confidence is very high, always above 0.9 for any song.

The second part of tests is mainly focused on compression. In this phase we test the identification after compression down to 32kbps. The compression can be combined with subsampling and/or conversion from stereo to mono. As an example in Table 1 is reported the results obtained after mono conversion and compression to 64 kbps comparing nine example songs. The reported number are the parameter *d* obtained comparing the compressed version AS* with the original S*. The obtained confidence is high also in this case, above 0.8.

	S1	S2	S3	S4	S5	S6	S7	S8	S9
AS1	0,19	0,50	0,61	0,56	0,70	0,62	0,60	0,67	0,56
AS2	0,54	0,14	0,39	0,83	0,70	0,51	0,52	0,53	0,48
AS3	0,61	0,48	0,16	0,75	0,56	0,63	0,43	0,39	0,56
AS4	0,45	0,72	0,71	0,11	0,62	0,53	0,73	0,89	0,48
AS5	0,64	0,71	0,50	0,55	0,24	0,46	0,56	0,63	0,53
AS6	0,56	0,58	0,52	0,43	0,45	0,19	0,61	0,74	0,39
AS7	0,67	0,67	0,41	0,71	0,51	0,66	0,25	0,46	0,71
AS8	0,73	0,57	0,41	0,90	0,66	0,77	0,45	0,19	0,69
AS9	0,44	0,38	0,51	0,57	0,56	0,46	0,61	0,67	0,12

Table 1 – Parameter *d* between original song (S*) and attacked version (AS*).

The highlighted cells on diagonal represents the comparison between a compressed song and its original. It is noticeable that all these values are below the threshold value *T*=0.3. Moreover all the values outside the diagonal are higher than the threshold. This means that all songs can be correctly retrieved and there are no error cases.

Figure 6 shows an example of parameter *d* for comparison between 100 compressed songs and original version. The continuous line represents the comparison of the songs with their original. The dashed line represents comparison between compressed versions with an original of a different song. It is noticeable that the continuous line is always below the line of the threshold *T*=0.3, while the dashed line is always above. Also in this case we do not have any error.

The last part of tests is focused on different attacks like cropping, noise addition and subsampling. These attacks do not cause problems to our algorithm. In any case the song can be identified with high confidence higher than 0.75.

The main features of our algorithm are the short excerpt of song needed for identification and the computation of a confidence parameter. In all our tests we do not find any errors so the performances are comparable with other algorithms in literature. More tests are going on increasing the size of our database in order to test the reliability of the system.

6. CONCLUSIONS

In this paper we presented an audio fingerprinting algorithm. This is based on a quantization approach of different features in the frequency domain. This technique provided good results in term of identification of attacked songs. The confidence as a function of the identification parameter is also studied and it provides an additional control on the identification.

Our theoretical work is now focused on different aspects: first of all we are trying to study a better modelling of error probability. On the other side we are working on the use of error correcting codes to improve the performance of the database search reducing the number of data to compare.

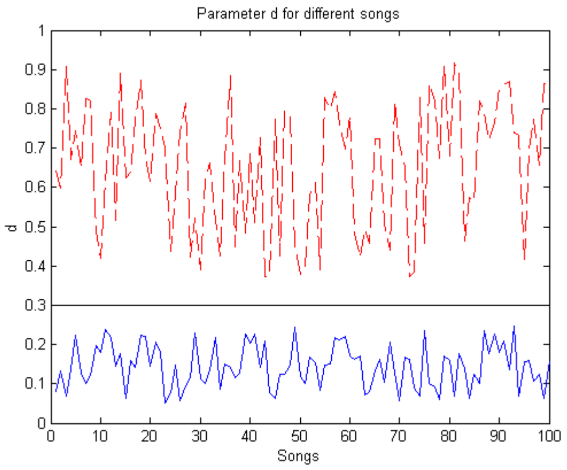

Figure 6 - Value of *d* for different mp3 compressed songs in case of comparison with its original (continuous line) or different songs (dashed line).

REFERENCES

[1] T.Kalker, J.Haitsma, and J.Oostveen, “Issues with Digital Watermarking and Perceptual Hashing”, *SPIE Conference on Multimedia Systems & Applications*, August 2001.

[2] P.Cano, E.Batlle, T.Kalker, J.Haitsma, “A Review of Algorithms for Audio Fingerprinting”, *IEEE International workshop on MMSP*, 2002.

[3] J.Haitsma, T.Kalker, and J.Oostveen, “Robust audio hashing for content identification”, in *Proc. of Content-Based Multimedia Indexing*, Brescia, Italy, September 2001.

[4] E.Wold, T.Blum, D.Keislar, and J.Wheaton, “Content-based Classification Search and Retrieval of Audio*”, IEEE Multimedia, Vol. 3 Issue 3, pp. 27 -36,* Fall 1996.

[5] J.Herre, O.Hellmuth and M.Cremer, “Scalable Robust Audio Fingerprinting Using MPEG-7 Content Description” *IEEE International workshop on MMSP*, 2002.

[6] P.Cano, E.Batlle, H.Mayer, and H.Neuschmied, “Robust sound modelling for song detection in broadcast audio”, in *Proc AES 112th Int. Conf.*, Munich, Germany, May 2002.

[7] M.L.Miller, M.A. Rodriguez and I.J.Cox, “Audio Fingerprinting: Nearest Neighbor Search in High Dimensional Binary Spaces”, *IEEE International workshop on MMSP*, 2002.