Fast content-based audio retrieval algorithm

César Pedraza - Jaime Vitola
Telecommunications Engineering - Electronics Engineering
Universidad Santo Tomas
Colombia
cesarpedraza, jaimevitola@usantotomas.edu.co

Johanna Sepúlveda
Microelectronics Laboratory
University of São Paulo
São Paulo, Brazil
jsepulveda@lme.usp.br

Jose I. Martínez
DATCCIA
Universidad Rey Juan Carlos
Madrid, Spain
joseignacio.martinez@urjc.es

***Abstract*—Fingerprinting is one of the most used techniques for searching and identification audio with a wide spectrum of applications. Different algorithms defines different fingerprint extraction and the match techniques, with different efficiency, computational load, robustness, response time and location search. Nowadays music audio retrieval faces two main challenges in order to be efficient: robustness and speed. This article proposes a fast algorithm to the audio content-based retrieval with the fingerprint technique, based on the extraction of the frequency features of the audio and a hash function. Experiments determined a high success rate and a response time lower than other techniques, optimal to real time applications like monitoring radio stations or songs identifying.**

***Keywords*—Hash function, Audio Retrieval, Fingerprinting .**

I. Introduction

In the last years, the world has experimented an exponential increase of audio an video consuming and production over different mass media. Because of this volume of information, it is necessary to develop different mechanisms to protect intellectual property, in order to guard it in radio broadcasting, market research, song identification, data mining, related contents search, and others [6].

Audio contents retrieval has been proposed as one of the solutions to this problem [3], by the recognition of audio tracks with copyright in different media in radio, television, internet, etc [13]. Decades ago, audio contents retrieval became one of the most important topics research in digital signal processing, and authors have proposed strategies such tagging, by adding some text into the audio stream, hashing by creating a key in order to extract some features from audio, watermarking that includes a non-perceptible mark into the audio stream, encryption watermarking, that includes the same mark but available only for some users, and finally, fingerprinting that extracts important characteristics from a audio to a set of numbers, to be distinguish from other tracks [15].

Many authors has selected fingerprinting as an effective technique to implement audio retrieval, in which audio frames of any length (windows) are reduced to a single fingerprint that identify their characteristics. This fingerprints numbers allow to perform a faster and efficient searching in a database. Despite there are many fingerprint techniques, the design of an algorithm with a low computational load and high reliability are one of the challenges for some authors around the world [4].

According to this, the reliability of an fingerprint algorithm faces some problems such as track speed changes, pitch variations, noise and others.On the other side, an algorithm should be efficient with the required computational load, in order to be executed in low-cost platforms and a high scalability for more complex applications [9].

In this article, an fingerprint algorithm for audio contents retrieval is proposed, based in the *shazam* technique, showing a high reliability and low cost implementation [16]. Section II presents the most important works around this problem and contemplates the computational load problem. Next, section III shows the proposed algorithm, in order to improve the implementation on low-cost platforms with a high reliability and efficiency. Section IV presents the experiments made to validate the performance of the algorithm and, finally in section V, conclusions and future work are showed.

II. Related Methods

Content based audio retrieval can be performed by different strategies, that define the way the extraction of the features and searching are carried out. E.g Espectral flux [17] verifies the speed of changing of the spectrum between two different audio frames, resulting in a high noise immune algorithm.

Seidel et al [13] designed the 2D chroma algorithm that extracts the frequencies with the maximum energy, in order to place them in a 2D map and to calculate a unique number through hash tables. The calculated number will be use to make the comparisons in the search process.

Venkatachalam [14] proposed the spectral subband centroids algorithm that extracts the component values for 16 bands of frequency between 500 Hz and 5300 Hz, to make a search process by using the euclidean distance. Other techniques use the MFCC (Mel Frequency cepstral coefficient) to represent simplified spectral information with a set of coefficients, which subsequently are used to compare with Dynamic Time Warping that determines the similarity between frames of data [2].

In the other hand Kamaladas [8] implements audio fingerprinting by using, frame division and superpositioning to avoid discontinuities. A 5-layer wavelet transform is applied to determined the main components in time-frequency, variance, zero crossing and energy centroid. This parameters are used to calculate a hash number that accelerates the search process in a database.

978-1-4799-1121-9/13/ $31.00

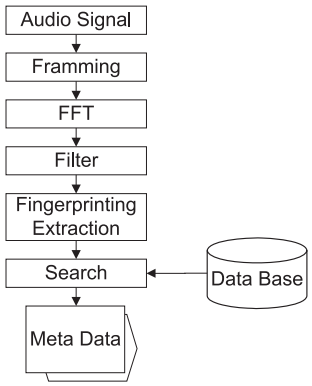

Fig. 1. General Algorithm.

Other techniques, involves training and learning algorithms to calculate an optimal euclidean distance, taking the original audio and altered audio (amplitude distortion, frequency, etc) to determine weather the two audio tracks are the same [7].

Gaussian mixture models (GMM) are used to extract features such as: shannon entropy, Rényi entropy, the spectral centroid, spectral bandwidth, Spectral band energy, Spectral flatness measure, Spectral crest factor y the MFCC (Mel-frequency cepstral coefficients). These parameters are intended to calculate a probability density[11] that distinguish one track from anothers.

[5] implements an audio fingerprinting technique called AudioDNA, which plays with AudioGenes to extract the information. This technique starts computing some characteristic values of 10ms frames of the audio tracks. After, carries on with a Hamming window convolution, an FFT, computing the MFCCs (Mel-Frequency Cepstral Coefficients) and a CDT. The AudioGenes are described as HMM (Hidden Markov Models) [1].

From all mentioned techniques, it can be inferred that fingerprinting algorithms are composed by two steps: features extracting and search process. In the same way, almost all algorithms can be represented in a generic way. Figure 1 shows a general front end for fingerprinting algorithms: first the audio is divided in groups by a framming step and then some characteristics are extracted by any data transform (usually to frequency domain through fast Fourier transform). In the next step, the extracted characteristics are selected to calculate a fingerprint that represents that window or set of samples. A comparison strategy is carried out to find the calculated fingerprint in a pre-processed fingerprints database.

III. Proposed Algorithm.

The proposed algorithm is presented in figure 2. As mentioned before, the frontend is based in the *shazam* algorithm, that extracts the most important frequency components of an audio window, in order to calculate a hash function that represents the features of it. The system is composed by two main parts: the first one, extracts the characteristics of the audio track to be search, represented by a set of numbers or fingerprint. The second one, performs the search of this numbers in a fingerprint database, and determines weather the track is copyright protected or not.

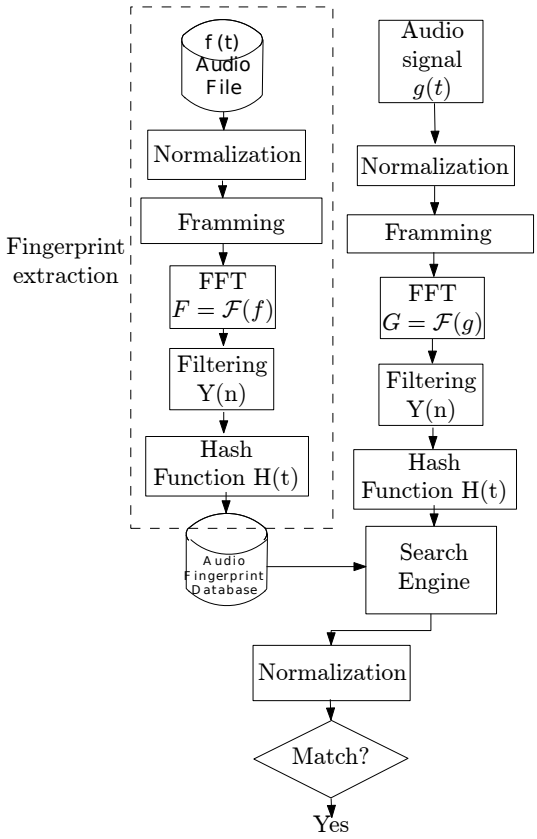

Fig. 2. Algorithm frontend.

As soon as the audio signal is digitalized and stored, all the frames are divided in groups of 2048 samples in order to make the frequency analysis. In this step, a fast Fourier transform is applied to each window, to obtain the frequencies where the signal energy is concentrated. Also, al filtering process is applied to discard redundant frequencies in some musical genres, such as electronic music. Finally, the most important frequencies (usually three) are selected to determine the features of the window.

A. Hash function.

After the extraction of the main characteristics of the audio signal, it is necessary to calculate a set of numbers (fingerprint) that represents it. In our case, a single fingerprint for a time (t), is composed by a set of 8 numbers. Each number was calculated by a Hash function that incorporates pairs of frequencies. Equation 1 describes the structure of the fingerprint. It shows that the main frequency for each time ($f_1(t)$) is paired with the most important frequencies in $t-1$, $t-3$, $t-5$ and $t-7$.

$$H_n(t) = \sum_{i=1,3,5,7} = f_1(t_n) \cdot f_1(t_n - i) + f_1(t_n) \cdot f_2(t_n - i) \quad (1)$$

B. Search algorithm.

After a fingerprint is calculated, a search process in a database is executed. In order to improve this process, a hash

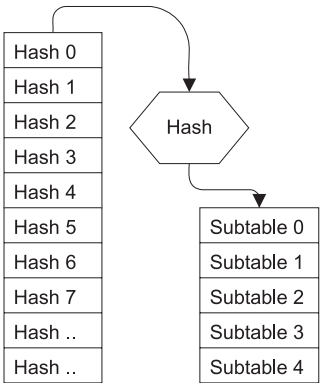

Fig. 3. Table Fragmentation.

number was calculated, allowing to divide the search space. This hash value is composed by three digits, calculated from the first number of the fingerprint. It will determine the first data with the same hash in the database, figure 3 shows the way the search process is accomplished.

IV. Discussions and Results.

A. *Experiment setup.*

To determine the quality of the algorithm, an experimental environment was made up of different real time tests on audio tracks of 5, 10, 20 y 30 seconds with a 4000 tracks database, which is equivalent to 330 hours of audio approximately. These track lengths was selected taking into account the typical durations of television and radio advertisements, in order to determine optimal lengths which the algorithm works.

On the other hand, tests were taken with the same audio tracks with different white random gaussian noise levels, to determine the robustness of the algorithm under channel distortions, modulation effects, and other broadcasting facts.

B. *Time response.*

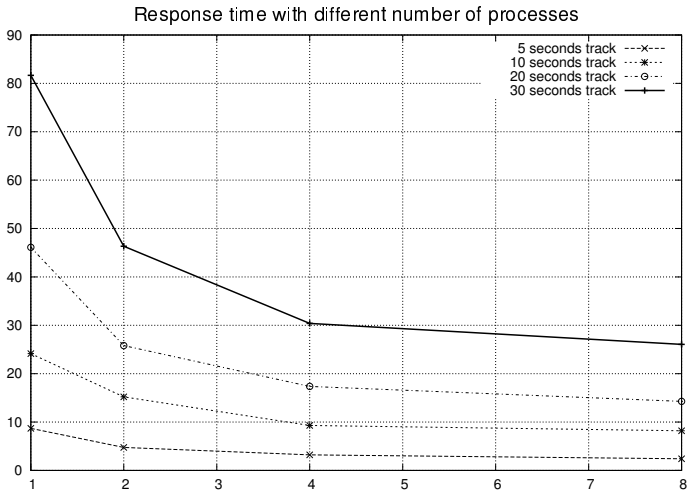

Fig. 4. Coincidences of a positive search with white noise.

To obtain time response values, the algorithm was executed in a Intel core i7, 4GB personal computer. Figure 4 shows the values for a search process with 5, 10, 20 and 30 seconds tracks in the database, and creating 1, 2, 4 and 8 executing processes. The number of processes depends on the number of cores available for this processor family.

Table I shows the response time of the algorithm for a search in a 3600 seconds database, in order to compare the performance against with other implementations implementing correlation and Dynamic Time Warping (DTW) [10]. It can be observed a better performance of the proposed algorithm compared against the previous implementations, with response times of some milliseconds, which is more suitable for real time applications with a high volume information.

TABLE II. Speedup for different number of processes against one process.

Track length	2 processes	4 processes	8 processes
5	1.8	2.66	2.6
10	1.53	2.55	2.87
20	1.76	2.7	3.06
30	1.78	2.73	3.03

Table II shows the results of executing the algorithm in multi-process mode. The speedup number is shown for 2, 4 and 8 processes against 1 process execution time. It can be observed a better performance when more than one process is executed, but this improvement is not linear, due the communications load generated during the execution of the algorithm.

Figure 4 shows the efficiency when the program is executed with up to 8 processes, determined by the number of cores of the processor. This characteristic makes clear the scalability of the system when executed in multiple processing cores. A minor performance is noticed when it is executed from 4 to 8 processes, this behaviour can be explained in the Intel Hyper Threading Technology ©.

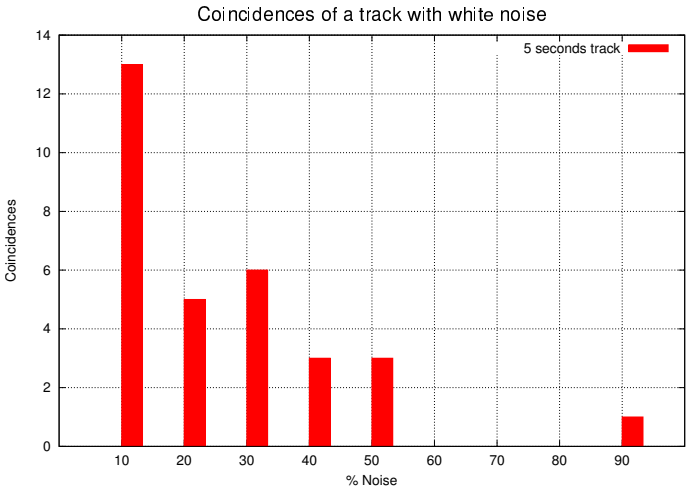

Fig. 5. Coincidences for a positive search with white noise.

In the noise robustness tests, different levels of noise where added to the same testing tracks and the number of fingerprints coincidences in the database were calculated. This test permitted to determine a maximum of noise allowed by the algorithm in order to operate correctly.

TABLE I. RESPONSE TIME FOR A SINGLE SEARCH IN A 3600 SECONDS DATABASE.

Track [s]	TR algorithm	TR algorithm DTW [12]	TR algorithm XCORR [10]
5	27.27 ms	37.25	193.72
10	75.76 ms	58.28	383.42
20	139.4 ms	101.62	763.55
30	248.4 ms	141.13	1140.59

Figure 5 shows the number of coincidences when 10% to 90% of noise were added to a 5 seconds track. It can be observed a high sensibility to detect tracks when total of noise added exceed 20% of the track amplitude.

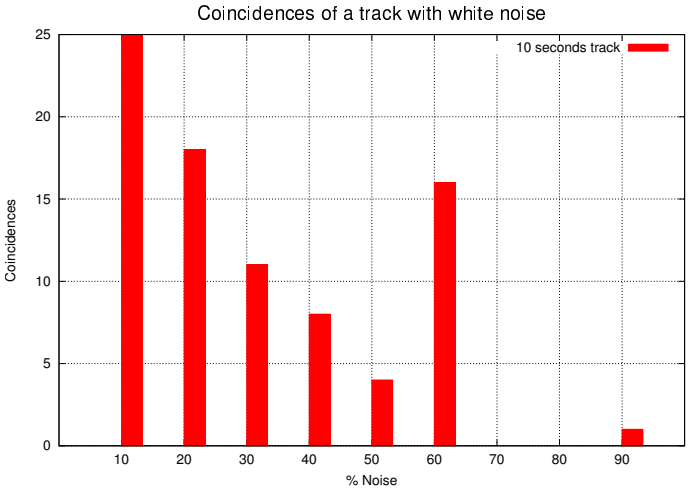

Fig. 6. Coincidences for a positive search with white noise.

In the same way, figure 6 shows the behaviour of the algorithm when different levels of noise were added to a 10 seconds track. It is clear a positive detection of the track goes difficult when noise levels exceed 40% of the amplitude.

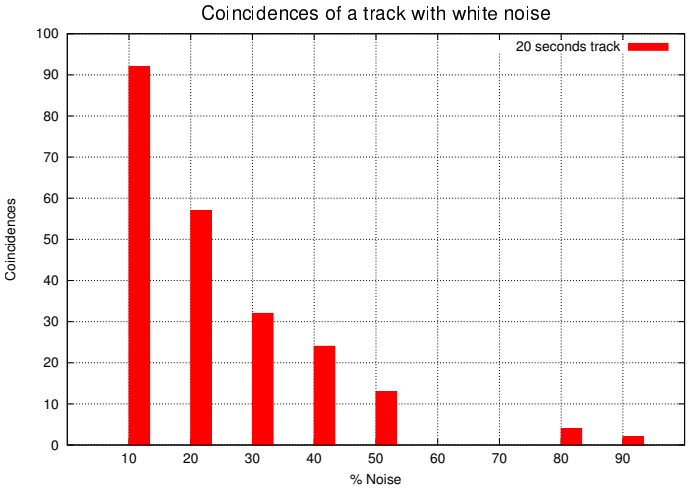

Fig. 7. Coincidences for a positive search with white noise.

Figure 7 and 8 shows the behaviour for 20 and 30 seconds tracks. In the first case, tolerance to noise is observed up to 50% of amplitude noise. In the second, it is noticed a good behaviour up to 50% or 60% of noise.

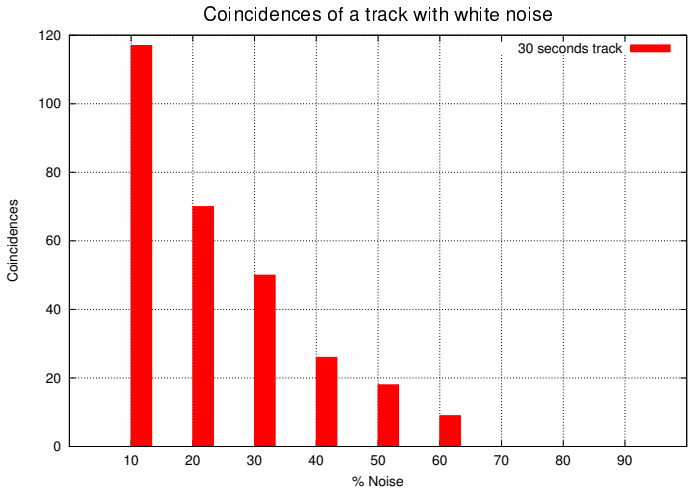

Fig. 8. Coincidences for a positive search with white noise.

V. CONCLUSIONS.

A fast shazam-based fingerprint algorithm was implemented. It was observed a high scalability of the system for parallel architectures execution.

It was observed a performance reduction of the algorithm when white noise were added to the tracks. 5 seconds tracks were more sensible to this tests than others tracks. Nevertheless for 10, 20 and 30 seconds tracks it was noticed a high robustness against noise in the algorithm for identifying tracks in the database.

It was evidenced a good performance of the algorithm in parallel platforms, determined by the speedup number compared against 1 process. This factor, demonstrates the viability to migrate the algorithm to mass parallel platforms, such as cloud computing systems, and may start an exploration topic to improve the algorithm for robust applications in investigation or to be commercialized.

The content-based audio retrieval, specially with the fingerprinting technique has many applications and may offers solutions to the copyright laws that can be infringed in any place.

REFERENCES

[1] E Batlle, J Masip, E Guaus, and P Cano. Scalability issues in an HMM-based audio fingerprinting. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 1, pages 735–738 Vol.1, 2004.

[2] D G Bhalke, C B R Rao, and D S Bormane. Dynamic time warping technique for musical instrument recognition for isolated notes. In *Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference on*, pages 768–771, 2011.

[3] Sung Bokyung, Jung Myungbum, Ham Junscok, Kim Jackyung, Park Junhyoung, and Ilju Ko. Feature Based Same and Audio Perception method for Filtering of Illegal Music Contents. *Time*, pages 194–198, 2008.

[4] P Cano, E Batlle, T Kalker, and J Haibma. A Review Algorithms Audio Fingerprinting. In *IEEE International workshop on MMSP*, volume pp, pages 169–173, 2002.

[5] P Cano, E Batlle, H Mayer, and H Neuschmied. Robust sound modeling for song detection in broadcast audio. In *Proceedings of the 112th AES Convention*, volume pp, pages 1–7, 2002.

[6] Ching-Wei Chen, R Cook, M Cremer, and P DiMaria. Content identification in consumer applications. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 1536–1539, 2009.

[7] Dalwon Jang and C D Yoo. Fingerprint matching based on distance metric learning. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 1529–1532, 2009.

[8] M D Kamaladas and M M Dialin. Fingerprint extraction of audio signal using wavelet transform. In *Signal Processing Image Processing Pattern Recognition (ICSIPR), 2013 International Conference on*, pages 308–312, 2013.

[9] D Luebke, M Harris, and N Govindaraju. GPGPU: general-purpose computation on graphics hardware. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 208. ACM, 2006.

[10] J.I. Martinez, J. Vitola, A. Sanabria, and C. Pedraza. Fast parallel audio fingerprinting implementation in reconfigurable hardware and gpus. In *Programmable Logic (SPL), 2011 VII Southern Conference on*, pages 245–250, 2011.

[11] A Ramalingam and S Krishnan. Gaussian Mixture Modeling of Short-Time Fourier Transform Features for Audio Fingerprinting. *Information Forensics and Security, IEEE Transactions on*, 1(4):457–463, 2006.

[12] A. Sanabria, J.V. Sepulveda M.J., Oyaga, and C.P. Bonilla. Algoritmo rápido para la búsqueda de audio por contenido. In *IBERCHIP Proccedings, XVIII Workshop, México*, pages 169–173, 2012.

[13] C Seidel. Content fingerprinting from an industry perspective. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 1524–1527, 2009.

[14] J S Seo, Minho Jin, Sunil Lee, Dalwon Jang, Seungjae Lee, and C D Yoo. Audio fingerprinting based on normalized spectral subband centroids. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 3, pages iii/213–iii/216 Vol. 3, 2005.

[15] V Venkatachalam, L Cazzanti, N Dhillon, and M Wells. Automatic identification of sound recordings. *Signal Processing Magazine, IEEE*, 21(2):92–99, 2004.

[16] Hongxue Wang, Xiaoqing Yu, Wanggen Wan, and R Swaminathan. Robust audio fingerprint extraction algorithm based on 2-D chroma. In *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*, pages 763–767, 2012.

[17] Wengen Wang, Xiaoqing Yu, Yun Hui Wang, and R Swaminathan. Audio fingerprint based on Spectral Flux for audio retrieval. In *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*, pages 1104–1107, 2012.