A Binary Auditory Words Model for Audio Content Identification

Alberto Gramaglia
Audioneex.com
alb.gramaglia@audioneex.com

Abstract

An Audio Content Identification method is presented, that uses Local Binary Descriptors and Machine Learning techniques to build an audio fingerprinting model based on "auditory words" inspired to the "visual words" model used for image recognition. This model forms the basis of an audio recognition system centered around a multi-level matching algorithm using the Generalized Hough Transform and Geometric Hashing.

1. Introduction

As the world becomes increasingly interconnected the production of multimedia content is exponentially growing, and this mine of information can be exploited to acquire new knowledge to be used in business processes or in our everyday life. This requires the development of tools capable of analyzing such huge amounts of data in order to extract useful information, and has drawn the attention of the research community for the design of efficient methods to perform automatic processing of audio/visual data in order to recognize high level contents. In this paper we introduce a method to identify content in audio data using techniques developed in fields outside audio processing and show that they can be used with success to design a high performance audio content recognition system.

2. Audio fingerprinting

The concept of fingerprint has been extensively used in several fields as a mean to identify objects from their unique characteristics, and audio fingerprinting is the technique that uses this concept for audio content identification. The main idea is that of extracting perceptually meaningful features that best characterize the audio signals in order to build a compact "signature" that can later be used to identify specific content from unknown audio data.

Many methods have been developed that use different fingerprint models, from physiologically motivated approaches based on a model of the inner ear that decomposes the audio signal into its frequency components using the DFT coupled with some kind of filter-bank from which to extract relevant features, such as in [1], to methods using statistical classifiers to discover perceptually meaningful audio components [2], landmark points over 2D representations of sound [3], combinatorial hashing [4] and clustering [5].

Some interesting methods have been proposed that use computer vision techniques, notably the works in [6] and [7] have shown the validity of this approach. The motivation behind this idea is that most audio fingerprinting methods use some sort of 2D representation of the audio signal (STFT, Chromagram, Cochleogram, etc.) and such representations can be directly used to extract features using 2D computer vision techniques in order to build robust and efficient fingerprints.

In the following sections we show how not only it is possible to use isolated computer vision techniques (such as descriptors extraction) but also adopt entire models and paradigms to efficiently solve audio identification problems.

3. Audio components

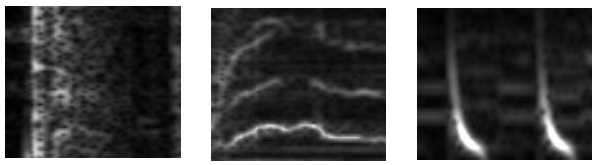
One of the most potent features of the human brain is that of recognizing objects (but also abstract concepts) using a hierarchical approach where complex entities are represented by smaller components and modeled using a structured pattern. An image recognition model based on these concepts has been proposed in the Computer Vision community [14] and has proven successful in the identification of visual objects. Following this approach, we can think that the auditory system uses low level components as well as a primary representation of sounds. These audio

components are characterized by their time-frequency distribution of intensity, the same as low level visual features are characterized by their intensity distribution on the 2D space.

Noise-like components: the main feature of these components is the absence of a structured pattern and the random distribution of energy across a broad frequency band.

Tone-like components: these are audio components characterized by energy distributions across a frequency range following a regular pattern due to the harmonic content of the sound and the presence of a fundamental frequency, which is used to determine the "pitch".

Pulse-like components: these audio components are characterized by, more or less, uniform intensity distribution across a broad frequency range with high impulsiveness (high energy released in a very short period of time).



(a) (b) (c)

Figure 1. Examples of audio components: (a) noise-like, (b) tone-like, (c) pulse-like.

Any sound can be seen as a combination of these audio components to form a time-space structure representing the auditory scene. Similar concepts were used in [2] where these components were learned from music data sets using HMMs.

Drawing on these ideas, we developed a method that aims at building a suitable representation of sound based on audio components, which we call *"auditory words"*, in order to design a robust audio fingerprinting scheme for audio content identification. In the following section a detailed explanation of the method is given.

4. Audio descriptors

A key issue is the representation of the auditory words in a manner that the resulting fingerprints are compact, with a good discriminative power and able to recognize generic audio from very short clips extracted at any point within a recording, in order to be suitable for real-time applications. Specifically, we want descriptors to be:

1. Locally informative
2. Stable
3. Compact and fast to compare
4. Time-translation invariant

We start with resampling the audio signal to cover a delimited frequency band where most of the useful information to human listeners lie (the frequency range 100-3000 Hz) and transpose it to the frequency domain by applying the Short Time Fourier Transform (STFT) with a window size of 1024 samples and a hop time of 13.88 ms smoothed using the Hamming window. We then look for Points of Interest (POIs) by searching for *onsets* in the frequency spectrum, which indicate the presence of relevant audio events (instrument notes, vocals, natural processes, etc.). These onsets generally produce peaks of consistent intensity so we scan for "consistent peaks" in the STFT spectrum. A consistent peak is defined as a small, well localized burst of energy characterized by its maximum M, width W and energy content E_P defined as follows

$$E_P = \sum_{p \in W \times W} |STFT(p)|^2 \tag{1}$$

A 2-stage approach is used to find the POIs: in the first stage we scan the STFT spectrum for candidate points by convolution with the parametric kernel (2)

$$H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & k & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{2}$$

to find local maxima in order to detect good candidates. The k parameter is a boosting factor that determines the sensitivity of the filter to the peaks and should be properly set to avoid the selection of local maxima that are not consistent enough (we used k=5). The result of this stage is the set of candidates given by

$$C_1 = \{p | STFT(p) * H > 0\} \tag{3}$$

In the second stage we perform a non-maximum suppression filtering on C_1 to get rid of inconsistent peaks by centering a window W_p in p (we used a

size of 400 ms × 340 Hz) and get the final set of detected POIs as follows

$$C_{POI}=\{p\in C_1 | p=\underset{p\in W_p}{argmax}\, E_P(p)\} \quad (4)$$

The choice of the descriptor was carefully made by looking at the class of *local binary descriptors,* which satisfy the four requirements stated earlier and that provide ease of computation, efficient storage and fast comparisons. Almost all of them are derived from or inspired to the Census Transform (5), introduced in [8], and methods based on this approach can be found in [9][10][11].

$$d(p)=\underset{p'\in N(p)}{\cup} b(p,p') \quad (5)$$

The Census Transform maps a neighborhood *N(p)* of a point *p* to a bit string *d*, using a comparison binary function *b*(-,-). Our descriptor is computed as follows: at each POI $p\in C_{POI}$ a neighborhood *N*(*p*) of extension $\Delta F_{N(p)}\times\Delta T_{N(p)}$ is considered (the used size of *N(p)* is 300 ms × 200 Hz). *N*(*p*) is then scanned sliding a small window *Wc* of size $\Delta F_{Wc}\times\Delta T_{Wc}$ with a stride of $s=(s_t,s_f)$ at evenly spaced points *p′*, as depicted in Figure 2(a). For each scanning window *Wc* at each scanning point *p′* the Mean Energy E^{μ}_{Wc} is computed and compared to those of its *k*-neighborhood, given by moving *Wc* in the *k* directions, as depicted in Figure 2(b). The Mean Energy differences are then mapped to the binary space using the following function

$$b(p',p'+\delta)=\begin{cases}1 & if\ E^{\mu}_{Wc}(p')-E^{\mu}_{Wc}(p'+\delta)\\ 0 & otherwise\end{cases} \quad (6)$$

where E^{μ}_{Wc} is given by

$$E^{\mu}_{Wc}=\frac{1}{|W_c|}\sum_{p\in Wc} STFT(p) \quad (7)$$

and **δ** is the displacement in the *k* directions. This process yields a binary vector $\boldsymbol{v}\in\{0,1\}^k$ for each scanning window in *N*(*p*) whose components are given by (6). This "sub-descriptor" captures the spatio-temporal relations between local audio events and the concatenation produces the final binary descriptor *d(p)* for *N*(*p*). You can think of this process as the assembling of a word from its phonemes. The final descriptor is given by (8).

$$d(p)=\bigoplus_{W_c\in N(p)} \boldsymbol{v}(W_c) \quad (8)$$

The number of scanning windows *Wc* and the neighborhood *k* will determine the size of the descriptor. Specifically, if N_{Wc} is the number of scanning windows, then the size of *d*(*p*) will be

$$|d(p)|=k\cdot N_{Wc} \quad (9)$$

where N_{Wc}, in turn, depends on the strides used to scan *N*(*p*). In our experiments we used a 4-neighborhood (*k*=4) in the N, E, S, W directions and a stride/displacement of 50%. Using a 4-neighborhood rather than a fully connected 8-neighborhood we didn't notice any significant reduction in the recognition accuracy, while reducing the size of the descriptors (and thus of the fingerprints database) by a factor of 2. The size of *Wc* was set to 50ms × 35Hz.

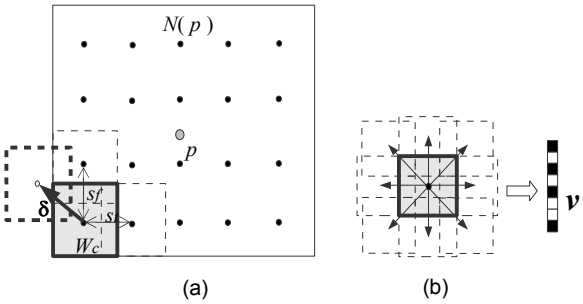


Figure 2. Local binary descriptor: (a) the scanning window is slided over the neighborhood *N(p)* and (b) the energy content of the surrounding space is assessed to produce the binary vector ***v*.**

5. Learning the auditory words

Each descriptor is a vector representing the local dynamics of the audio signal that's perceptually modeled as audio components. Specifically, they are, with the current settings, 720-bit vectors in the binary vector space. It is reasonable to think that small variations in these dynamics are still perceived as the same sound, therefore it makes sense to cluster this vector space in order to find a set of representative vectors into which to map the whole descriptor space. This set of vectors (the codebook) form the auditory words, which should be able to capture as much as possible the statistics of the data set in order to be able to describe it with good accuracy.

We find these auditory words by a learning process based on *binary vector quantization* using *k*-medians, which entails the evaluation of representative binary vectors through a quick selection procedure and the Hamming distance for similarity computation. The algorithm proceeds as follows:

1. From a dataset $D \subset [0,1]^n$ of binary descriptors extract a subset $V \subset D$. This subset (also called the "training set") should be highly heterogeneous in the chosen domain in order to capture as much as possible of the underlying statistics describing *D*.
2. Perform *k*-means++ seeding to pick the initial centroids $C(t_0)=\{c_1 \dots c_k\}$ from *V*, as follows:

   - Sample a point $v \in V$ at random from a uniform distribution as the initial centroid $c_1(t_0)$
   - For each remaining centroid $c_j(t_0)$ $j = 2 \dots k$

     1) Compute the p.d.f. of the points $v \in V$ according to (10), where $\|x\|$ denotes the Hamming distance

$$p(v)=\frac{\min_{c_j \in C} \|v-c_j\|}{\sum_{h=1}^{|V|} \min_{c_j \in C} \|v_h-c_j\|} \qquad (10)$$

     2) Sample a point $v \in V$ at random from the non-uniform distribution given by (10) using Inverse Transform Sampling, as follows

     2.1) Generate a probability value $u \in [0,1]$ at random from a uniform distribution

     2.2) Take the point $\bar{v}$ such that $F(\bar{v}) > u$ where *F* is the cumulative distribution function as in (11)

$$F(\bar{v})=\sum_{v=1}^{\bar{v}} p(v) \qquad (11)$$

     2.3) Set $c_j = \bar{v}$ and add it to $C(t_0)$

3. Perform *k*-medians according to the following algorithm

   - Form $S_j(t)=\{v \in V : \underset{c \in C(t)}{argmin} \|v-c_j\|\}$ clusters by grouping all $v \in V$ that are closer to centroid c_j than any other *v*
   - Compute new centroids $C(t+1)$ by computing a "median vector" in each cluster $S_j(t)$ as follows

$$c_j(t+1)=(M(v_1),\dots,M(v_n)) \qquad \forall v \in S_j(t)$$

     where $M(v_x)$ is given by the following function

$$M(v_x)=\begin{cases} 0 & if\ |Z_x|>|O_x| \\ 1 & if\ |Z_x|<|O_x| \\ c_{j_x} & if\ |Z_x|=|O_x| \end{cases} \qquad (13)$$

     and Z_x , O_x indicate the sets of zeroes and ones respectively for component *x* across all vectors in cluster $S_j(t)$. Equation (13) effectively computes the median of a component, and since vectors are binary this reduces to a simple count of how many ones and zeroes are in such component.
   - Repeat until all clusters are stable. The stability can be assessed by checking that the vast majority of vectors are assigned to the same clusters between iterations or that some cost function falls below a threshold.

The resulting codebook will be the set of auditory words used for recognition living in a reduced feature space than the the vector space in which the original descriptors live. Experiments show that 100 is the optimal number of auditory words, as depicted in Figure 3, which means the original 720-dimensional feature space can be mapped into a reduced 7-dimensional binary vector space, so that these words can easily fit into a machine word.

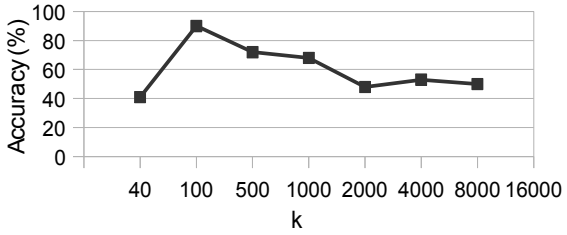


Figure 3. Optimal value for the cardinality of the Auditory Dictionary

The auditory words were learned from a music data set of 200 mixed genre songs and tested on another music data set using 200 query audio clips played over the air.

6. Matching

The most common structure used to quickly retrieve objects from a database is the *inverted index*, for which a vast literature exists, so we use this structure to quickly search the fingerprints space. A fingerprint is an ordered sequence $F = \{\boldsymbol{s} = <w, t, f, e>\}$, $\boldsymbol{s}$ being a small structure called a "local fingerprint", and each posting P in the inverted lists has the following layout

$$P=\langle F, s, t, e\rangle \tag{14}$$

where F is the id of the fingerprint in which the word w occurs, s the local fingerprint id (a sequential number), t the time location of the local fingerprint and e the quantization error of w. Each term τ_{ic} is formed by the concatenation of the i-th auditory word with the c-th channel where that word occurs, such channels being obtained by dividing the spectrum into Nch frequency channels (we used Nch=60).

The matching algorithm is based on three properties of our fingerprint model

1. *Time proximity*: the local fingerprints in an audio sequence occur all within a defined, bounded and arbitrary time frame, that is

   $$\forall(w_i, w_j)\in X \quad \Rightarrow \quad |t(w_i)-t(w_j)|\le T_b$$

   where T_b is an arbitrary time interval.

2. *Time order*: the local fingerprints in an audio sequence, by construction, are ordered in time (and monotonically labeled), that is

   $$t(w_i)\ge t(w_j) \qquad \forall i > j$$

3. *Spatio-temporal coherence*: if $F=\{s_1 \dots s_n\}$ is the set of local fingerprints extracted from an audio recording and $\Psi(F)$ a function describing the spatio-temporal relationships between the local fingerprints in F, then for any two perceptually similar audio F_1 and F_2 must be $|\Psi(F_1)-\Psi(F_2)|\leqslant\epsilon$, with ε sufficiently small.

The first stage algorithm falls into the category of Generalized Hough Transforms and uses a matrix Mc to capture similarities between the query fingerprint and the reference fingerprints in the database D exploiting property 1 and 2. The unknown query sequence $X=\{x_1, \dots, x_k\}$ is quantized using the dictionary of auditory words. The low cardinality of the dictionary and the use of binary descriptors with the Hamming distance makes this process extremely fast. After X is transformed into a sequence of auditory words, the search is carried out using the similarity matrix Mc and the inverted index. For each auditory word $w_i=q(x_i)$ in the query the corresponding inverted list is retrieved, by means of the term index τ_{ic} computed as described earlier, and the postings processed sequentially. Specifically, for each posting P in the inverted list $L(\tau_{ic})$ the cells $M_c(F, b)$ are selected in the similarity matrix, where b is computed as $b=t/T_b$ and a score is assigned according to the following scoring functions

$$S_{tp}(F, b)=\begin{cases} a_{tp}\cdot K_{tp} & \text{if } s\in M_c(F, b) \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

and

$$S_{to}(F, b)=\begin{cases} a_{to}\cdot K_{to} & \text{if } t_i\ge t_{i-1} \qquad s_i, s_{i-1}\in M \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

where s_i represents the word in the candidate fingerprint F being selected by w_i at step i, K_{tp} and K_{to} are arbitrary constants, and a_{tp}, a_{to} are weights computed as follows

$$a_{tp}=1-\frac{|e_w-e|}{|d|} \tag{17}$$

and

$$a_{to}=\frac{n_o}{n_c} \tag{18}$$

where e_w and e are the quantization errors of the query word and the word in the candidate fingerprint respectively, $|d|$ is the size of the descriptor (in bits), n_o the number of candidates satisfying the time order property in the selected matrix cell and n_c the number of candidates in the cell. The total score is then given by

$$S_{TOT}(F, b)=S_{tp}(F, b)+S_{to}(F, b) \tag{19}$$

The idea is to capture similarities between fingerprints by measuring how well property 1 and 2 are satisfied using the above scoring functions, where $S_{tp}()$

quantifies the time proximity and $S_{to}()$ the temporal order. For each candidate word that gets clustered in the matrix we also save a list of pairs $C=\{\langle s, x_i\rangle\}$ that will be used at a later stage. The cells of *Mc* hold the following information in an appropriate structure

$$M_c(F,b)=\langle S_{TOT}, t_{s_{i-1}}, n_o, C\rangle$$

To find the best matches we then proceed as follows: be $M_c(F)=[b_1,...,b_m]$ a row (column) of *Mc* for some *F* in *D*. We denote this row (column) vector by $H_F(b)$ and, since it represents time-related value distributions, we call it a *time histogram* for *F*. For each time histogram in *Mc* find the bin with maximum value b_{max}, that is

$$b_{max}=\underset{b\in[1...m]}{argmax}\, H_F(b) \qquad (20)$$

and store the values $H_F(b_{max})$ in a heap. The set of top-k elements in the heap are the candidate matches to the query *X*. We denote this set by $D_c=\{F_1,...,F_k\}$. It represents a partition of *D* with a high probability of finding the true match. Figure 4 shows a pseudo-code for the algorithm above described

In the second stage, property 3 is verified using the set of candidate pairs $C=\{\langle s_{ij}, x_k\rangle\}$ associated with the top-*k* candidates in the similarity matrix *Mc* as seeds to perform sequence alignment and graph matching.

Be $X^k=\{x_{k-\Delta},...,x_k,...,x_{k+\Delta}\}$ a sub-sequence extracted from the query sequence centered at x_k and $G(X^k,E^k)$ the fully connected graph having as nodes the local fingerprints in X^k and as edges E^k the position vectors between nodes in the time-frequency space. Each query local fingerprint x_k processed in the time clustering stage picks a set of similar local fingerprints belonging to some reference fingerprints in *D*. These are stored in the cells of the matrix *Mc* as a set of pairs, where s_{ij} is the *j*-th local fingerprint from fingerprint *Fi* in *D*. We use the sets *C* taken from the top-*n* bins of the time histograms $H_F(b)$ for each candidate in the top-*k* set $D_c=\{F_1,...,F_k\}$ instead of just from the bin b_{max} in order to increase the probability of match. To each query local fingerprint x_k is then associated a set of candidate local fingerprints from the database, as illustrated in the following relation

$$x_k \rightarrow C_k=\{s_{ij}\} \qquad (21)$$

Be $F^j=\{s_{i,j-\Delta},...,s_{ij},...,s_{i,j+\Delta}\}$ a sub-sequence from the top-*k* candidate fingerprint *Fi* centered at s_{ij}, and $G(F^j,E^j)$ the fully connected graph having as nodes the local fingerprints in F^j and as edges E^j the position vectors between them. Property 3 can then be evaluated by aligning the query sequences X^k, determined by the local fingerprints x_k, and the candidate sequences F^j, determined by the set C_k, and matching the graphs $G(X^k,E^k)$ and $G(F^j,E^j)$ using a scoring function $S_{coh}(G(X^k,E^k),G(F^j,E^j))$, which will be defined later, as depicted in Figure 5. The Δ parameter determines the size of the neighborhoods around the reference local fingerprints to be matched, set to a value in the range [10,20].

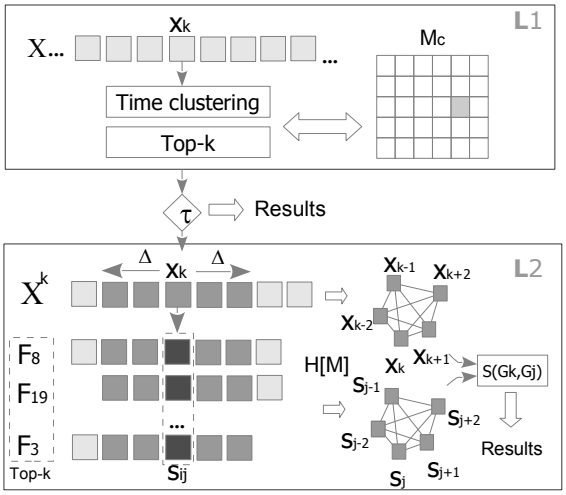


Figure 5. The 2-Level matching process.

The correspondence between the produced graphs is evaluated by means of a graph matching procedure called *Pairwise Geodetic Hashing*, an algorithm based on the Geometric Hashing technique, a well known method for object matching introduced in [15] and [16].

Given two attributed graphs $G(X^k,E^k,A^k)$ and $G(F^j,E^j,A^j)$ the adjacency matrices M_X and M_F can be built so that each element of the matrix represents the position vector $\vec{v}$ between two local fingerprints. Since the graphs are simple and undirected, these adjacency matrices are symmetrical and triangular, so for a $n \times n$ matrix we only need to consider $n(n-1)/2$ elements. These elements can be indexed in a lookup table, where each index is

associated to the pair of local fingerprints connected by that position vector, along with some attributes A. The similarity between the graphs can then be quickly computed using these lookup tables by scoring the matching edges and the similarity between the nodes according to the set of attributes. The time and frequency values are quantized to allow some deformation in the graphs and account for errors due to noise. To disambiguate the edges we pair them to a reference edge represented by the position vector between the local fingerprint being processed and a reference local fingerprint. This reference node should be one that's common to both the query graphs and the candidate graphs in order for the match to succeed. As reference nodes we then choose the query local fingerprints x_k and the associated candidates $C_k=\{s_{ij}\}$ picked in the time clustering stage. If x_k and s_{ij} happen to belong to the same fingerprint then the graphs they induce will be constructed using the same reference system and they will match to some degree.

The lookup tables are implemented using hash tables to get O(1) time complexity and have the following structure

$$H[M(i,j)]=(s_i,s_j) \tag{22}$$

where the keys are created by hashing the elements of the adjacency matrix using the following hashing function

$$M(i,j)=v_{ij}^t \oplus v_{ij}^f \oplus v_{i,ref}^t \oplus v_{i,ref}^f \tag{23}$$

the $\oplus$ operator indicates concatenation, v_{ij}^t and v_{ij}^f the time component and frequency component respectively of the position vector between the *i*-th node and *j*-th node, $v_{i,ref}^t$ and $v_{i,ref}^f$ the time component and frequency component of the position vector between the *i*-th node and reference node. These hashed values will be the keys *e* of the hash table.

The scoring is performed by considering both the geometric relationships and the similarity between local fingerprints using specific node attributes according to the following scoring function

$$S_{coh}(G(X^k,E^k,A^k),G(F^j,E^j,A^j))=¿ \sum_{e\in E^i\cap E^j} K_c+a_s(e)\cdot K_s \tag{24}$$

where

$$a_s(e)=2-\frac{1}{|d|}\left[D_H(x_i^e,s_i^e)+D_H(x_j^e,s_j^e)\right] \tag{25}$$

For each common edge *e*, that is the intersection between E^k and E^j which can be computed by finding common keys between the hash tables H_X and H_F, the function gives a constant score K_c for the spatio-temporal coherence and a weighed score K_s for the similarity between the local fingerprints connected by the edge. $D_H(x^e,s^e)$ denotes the Hamming distance between the query local fingerprint and the candidate local fingerprint at both ends of the common edge *e*. In our implementation we used the same value for K_c and K_s (K=1000). Note that the nodes attributes in this case coincide with the local fingerprint descriptors but other parameters may be used to measure the similarity, as we'll see below.

The similarity function (25) requires the binary descriptors be stored in order to compute the Hamming distance, which can result in high storage space requirements for large data sets. To avoid this, the following similarity function can be used

$$a_s(e)=\frac{1}{|d|}\left[\beta(w_{x_i^e},w_{s_i^e})+\beta(w_{x_j^e},w_{s_j^e})\right] \tag{26}$$

where $w=q(x)$ are the auditory words resulting from the binary vector quantization, $\beta()$ a function defined as follows

$$\beta(w_1,w_2)=\begin{cases}|d|-|\varepsilon(w_1)-\varepsilon(w_2)| & w_1=w_2\\ 0 & w_1\neq w_2\end{cases} \tag{27}$$

$\varepsilon()$ being the quantization error of w. This implementation uses simple comparison operators instead of the Hamming distance as a measure of similarity and only requires storing the auditory words and their quantization errors (which can be pre-computed) rather than the whole descriptors, with massive savings in terms of storage space. The results of this second stage are stored in a final top-*k* list. The effect of this “reranking” process is that of increasing the probability of match in the top-*k* list when the scores between the candidates are too close. The use of two different stages makes the matching algorithm a multi-level (2-level) process, which can be used to make the system adaptive, as depicted in Figure 6. Here we use a simple threshold to decide whether to activate the second stage, but something more sophisticated may be devised.

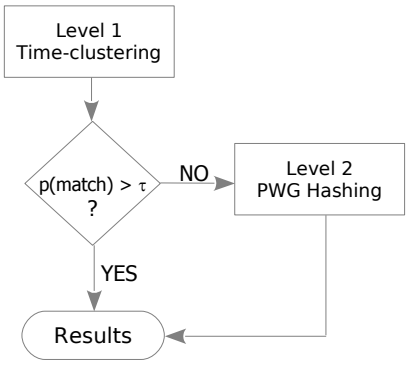


Figure 6. Multilevel matching scheme.

7. Performance

The performance of the proposed method have been evaluated using 10,000 query audio clips extracted from a set of 1,000 music recordings of different genres. To test the time-translation invariance of the fingerprint model the clips were taken at random offsets within the recordings.

7.1 Robustness

We applied various transformations to the query clips that have been typically used to test other audio identification systems and that are listed in Table 1. The first column reports the transformation applied while the second and third column report the accuracy for different lengths of the query clips. The results clearly show the validity of our method. However, applying single processing steps to audio signals is not very realistic, as in real world scenarios the audio typically undergoes different transformations before being delivered to the recipients. In this regard, we use different audio models in an attempt to represent different scenarios where ACI technology may be applied.

Table 1

Transformation	5-second	10-second
Equalization (as in [1])	99.9%	100%
Echo (delay = 250ms, decay = 0.2)	99.9%	100%
Tempo scaling (+10%)	99.7%	99.9%
Linear Speed Change +2%	88.9%	96.7%
Linear Speed Change -2%	87.2%	96.6%
MP3 encoding @ 44.1KHz, 32Kb/s	99.9%	99.9%
GSM encoding @ 8KHz, 13Kb/s	92.2%	98.10%

Model 1

This model represents a typical processing chain that is common, for example, in radio broadcasting. This chain is not standard and may vary considerably depending on the application, but after some research it appears that typical processing chains in broadcast audio include the stages depicted in Figure 7 below

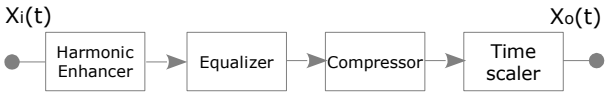


Figure 7. Model 1 test.

For the harmonic enhancer we added even order harmonics at a crossover frequency of 4kHz with a drive of 6dB, the equalization stage applied the EMI 78 curve to emphasize the bass while reducing noise at high frequencies, the compressor had a threshold of -35dB and ratio, attack and release of 6:1, 100ms and 1s respectively, while the tempo was scaled up by 10%.

Model 2

This model represents a tougher scenario where the original audio signal is passed through the processing chain of Model 1 and additively combined to another audio signal. A typical application that can be roughly modeled in this way is over-the-air identification, where pre-processed audio (for example a radio broadcast or other source) is played in an environment and affected by background noise before being recorded by the ACI system. The background noise sample used in the tests was recorded in a marketplace and has been randomly added to the query clips at various Signal to Noise Ratios.

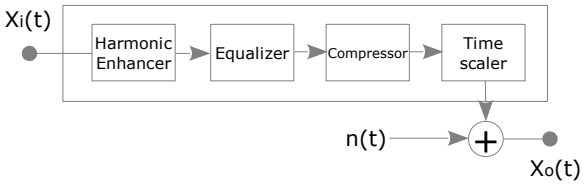


Figure 8. Model 2 test.

The results for the two models are reported in Table 2. It shows that the proposed method is also effective for recognition of distorted audio in the presence of additive noise. The tests were conducted following a forced-choice approach where all the query audio clips were taken from recordings present in the database and the system forced to choose a best match.

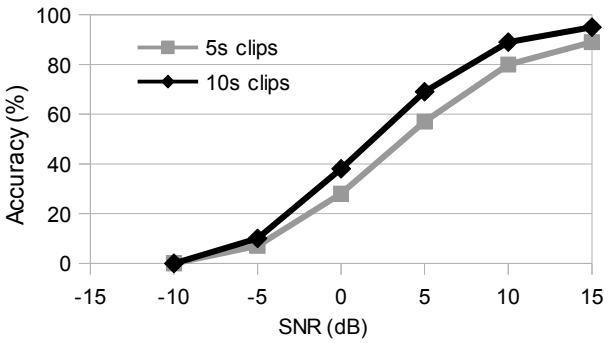


Figure 9. Model 2 identification accuracy

Table 2

Processing	5-second	10-second
Model 1	97.2%	99.4%
Model 2	See Figure 9	See Figure 9

7.2 Speed

The recognition time basically depends on the speed of the fingerprinting and matching processes, which have a constant and linear time complexity of O(T_i) and O(|D|) respectively at each processing step, where T_i is the duration of an input audio block (fixed) and |D| the size of the fingerprint database (generally variable). Extracting an audio signature from 1 second of audio took approximately 6 ms on average, while searching the fingerprint space for a match depends on the size of the fingerprint database and the distribution of the words in the index. On a simulated database of 10,000 recordings with an average duration of 5 minutes and a quasi-uniform distribution of words it took on average 210ms to search for a match. These figures where obtained using non-optimized algorithms (aside from the FFT computation) written in C++ and running on a laptop computer with a 2.53GHz i5 processor, 3MB L3 cache and 4GB of RAM.

7.3 Compactness

Our fingerprinting method is adaptive as it selects the interest points based on the characteristics of the audio, so the granularity is not fixed and depends on the size of the maxima suppression window. A rough estimation of the expected number of (local) fingerprints per time unit can be calculated using the following formula (28)

$$\mu_{LF} \simeq \left\lfloor \frac{\Delta T}{\Delta T_{W_p}} \right\rfloor \cdot \left\lfloor \frac{\Delta f}{\Delta f_{W_p}} \right\rfloor \qquad (28)$$

which, with the current settings, corresponds to 18 fingerprints per second. A statistical analysis on the music database used for our experiments reveals that this is indeed the average granularity, as can be seen in Figure 10. Considering that the auditory words are 7-bit binary vectors, the p.d.f. in Figure 10 indicates an average rate of about 18 words per second, that is a mean throughput of 126 b/s. Thus, our method produces very compact fingerprints allowing, for example, a music database of 1 million recordings with an average duration of 4 minutes be fingerprinted using less than 4GB of storage space (excluding support information).

8. Conclusions

We have presented an audio content identification method based on a binary Auditory Words model built using local binary descriptors and inspired to techniques and paradigms from the Computer Vision and Information Retrieval communities. Previous works have shown the efficacy of some methods used for visual recognition to solve audio content identification problems and our work demonstrates how not only specific processing techniques but entire models and paradigms can be adopted with success. Further work should probably be focused on the search algorithm, particularly on the use of more efficient data structures. A thorough analysis of the parameters of the system (especially those used to build the descriptors) to observe how the variation of these values impact the system's performances is also worth further investigation.

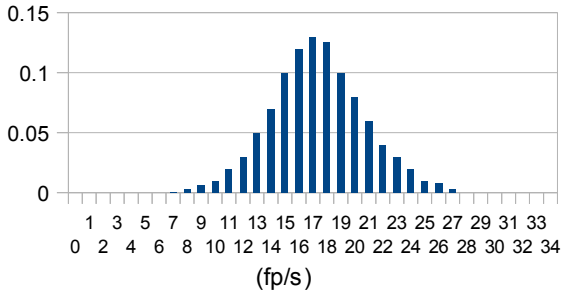


Figure 10. Distribution of fingerprint rates.

9. References

[1] J. Haitsma, T. Kalker. "A Highly Robust Audio Fingerprinting System", Proc. *International Conf. Music Information Retrieval,* 2002.

[2] P. Cano, E. Batlle, H. Mayer, and H. Neuschmied. "Robust sound modeling for song detection in broadcast audio", in *Proc. AES 112th Int. Conv.*, Munich, Germany, May 2002.

[3] Cheng Yang. "MACS: Music Audio Characteristic Sequence Indexing For Similarity Retrieval", in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.

[4] A.Wang. "An industrial-strength audio search algorithm", in *Proc. of International Conference on Music Information Retrieval (ISMIR)*, 2003.

[5] J. Herre, E. Allamanche, and O. Hellmuth. "Robust matching of audio signals using spectral flatness features", in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2001, pp. 127–130.

[6] Y. Ke, D. Hoiem, and R. Sukthankar. "Computer vision for music identification", in *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 597–604.

[7] Shumeet Baluja, Michele Covell. "Content Fingerprinting Using Wavelets", *Proc. CVMP*, 2006.

[8] Ramin Zabih1 and John Woodll. "Non-parametric Local Transforms for Computing Visual Correspondence", in *Proceedings of European Conference on Computer Vision, Stockholm, Sweden*, May 1994, pages 151-158

[9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "BRIEF: Binary Robust Independent Elementary Features", in *Computer Vision–ECCV 2010*, pages 778–792, 2010.

[10] A. Alahi, R. Ortiz, and P. Vandergheynst. "FREAK: Fast Retina Keypoint", in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "Orb: An efficient alternative to SIFT or SURF", in *International Conference on Computer Vision*, (Barcelona), 2011.

[12] Pasi Fränti, Timo Kaukoranta. "Binary vector quantizer design using soft centroids", in Signal Processing: *Image Communication* 14, 1999.

[13] C. Grana, D. Borghesani, M. Manfredi**,** R. Cucchiara, "A Fast Approach for Integrating ORB Descriptors in the Bag of Words Model", in press on *Proceedings of IS&T/SPIE Electronic Imaging: Multimedia Content Access: Algorithms and Systems*, San Francisco, California, US, Feb. 2-7, 2013

[14] J. Sivic and A. Zisserman. "Video Google: A Text Retrieval Approach to Object Matching in Videos", in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003.

[15] Jacob T. Schwartz and Micha Sharir. "Identification of Partially Obscured Objects in Two and Three Dimensions by Matching Noisy Characteristic Curves", in *The International Journal of Robotics Research,* 1987.

[16] Y.Lamdan, J.Schwartz, and H.Wolfson. "Affine invariant model-based object recognition", IEEE Trans. Robotics and Automation, 6:578-589, 1990.

[17] Justin Zobel and Alistair Moffat. "Inverted files for text search engines", in *ACM Computing Surveys,* 38(2):6, 2006.

[18] H. Turtle and J. Flood. "Query evaluation: Strategies and optimizations", in *Information Processing and Management,* 31(6):831–850, 1995

[19] M. Fontoura , V. Josifovski , J. Liu , S. Venkatesan , X. Zhu , J. Zien. "Evaluation Strategies for Topk Queries over MemoryResident Inverted Indexes",