
Journal of Intelligent Information Systems, 24:2/3, 159–177, 2005

Drum Loops Retrieval from Spoken Queries

OLIVIER GILLET olivier.gillet@enst.fr
GAËL RICHARD* gael.richard@enst.fr
GET-ENST (TELECOM Paris), 37-39, rue Dareau, 75014 Paris, France

Received May 10, 2004; Revised October 22, 2004; Accepted October 30, 2004

Abstract. Recent efforts in audio indexing and music information retrieval mostly focus on melody. If this is appropriate for polyphonic music signals, specific approaches are needed for systems dealing with percussive audio signals such as those produced by drums, tabla or djembé. In this article, we present a complete system allowing the management of a drum patterns (or drumloops) database. Queries in this database are formulated with spoken onomatopoeias—short meaningless words imitating the different sounds of the drumkit. The transcription task necessary to index the database is performed using Hidden Markov Models (HMM) and Support Vector Machines (SVM) and achieves a 86.4% correct recognition rate. The syllables of spoken queries are recognized and a relevant statistical model allows the comparison and alignment of the query with the rythmic sequences stored in the database, in order to provide a set of the most relevant drum loops.

Keywords: drum loops retrieval, percussive instrument recognition, audio indexing, content-based retrieval, music information retrieval

1. Introduction

Pre-recorded audio databases of drum signals are now widely used in modern music production. These signals are mostly short drum patterns intended to be repeated or "looped" to create rhythmic parts—and are therefore called *loops* or *drumloops* by professional musicians. Most of these databases are available as collections of audio CD or CD-ROM containing hundreds of loops, without any other information than their tempo or general style. The user has no other alternative than to manually browse the whole content of the CDs and listen to each individual file. Therefore, there is a need for more elaborated tools that will at the same time include content-based methods to efficiently search in these databases and propose more user-friendly interfaces to formulate a query.

An essential aspect of such a tool is the necessity to obtain an automatic transcription of the drum loop signals.

Most of the work in the domain of audio indexing is dedicated to melodic instruments (see for example Herrera et al. (2000) for a review on instrument recognition), however the transcription of percussive signals (such as drum signals for example) has gained much interest in the past few years. McDonald and Tsang (1997) identified isolated percussive sounds based on spectral centroid trajectories and Sillanpää et al. (2000), presented a classification system in five broad categories (Bass drum, snare drum, hi-hat, cymbal, and toms). More

*Author to whom all correspondence should be addressed.

recently, (Herrera et al., 2003) evaluated several methods for natural and synthetic drum signals recognition. These technics proved to be successful but were limited to isolated sounds. Other works deal with more complex signals and aim at extracting the drum tracks from polyphonic music signals (Zils et al., 2002), or use source separation approaches to pre-process drum loops signals (FitzGerald et al., 2002). A peculiarity of drum loops signals is that each event can be produced by simultaneous strokes on different instruments (for example bass drum and hi-hat). Lawlor et al. (2002) showed very promising results but this work was limited to three instruments (snare drum, kick drum and hi-hats) and tested on only fifteen manually selected loops.

A drum loop indexing system should also be able to characterize the rhythmic content of the drum loop and in particular its tempo. A number of tempo, beat tracking and rhythm estimation algorithms have been proposed (see for example Laroche (2003), Scheirer (1998), Alonso et al. (2004), Goto (2001), Raphael (2002)). Though, it is worth to mention that for drum loop transcription simple methods are often sufficient and that an explicit and quantized rhythmic description is not mandatory for a retrieval system.

Another particularity of drum loops is that they contain a succession of events (or strokes). As a consequence, drum loop signals or drum tracks often exhibit a temporal structure. Two concurrent studies have exploited such a structure by means of a sequence model, or "language model" by analogy with large vocabulary speech recognition systems (Paulus and Klapuri (2003) for drum sequences transcription, or Gillet and Richard (2003) for the transcription of tabla signals).

Similarly to audio indexing, most of the works in music retrieval focus on melody or on query by example (see Byrd and Crawford (2002) for example). One of the most popular approach, the so-called "Query by humming", aims at retrieving music files from a sung melody. A "query by rhythm" approach based on a simple similarity measure is also proposed in Chen and Chen (1998). Various systems are already implemented and show promising results: Ghias et al. (1995), Rolland et al. (1999), McNab et al. (1997) and Kornstädt (1998). However, they all require a high-level representation of the whole searched database (for example as a collection of MIDI files) and are mostly based on melody.

In the context of percussive signals where melody is hardly present, a different approach needs to be followed. One of the most natural ways to describe a drum signal is by means of spoken onomatopoeia—short meaningless words imitating the different sounds of the percussive instruments (drum in this context). This paper then introduces a novel music retrieval system for drum loops databases where queries are formulated as spoken onomatopoeia or possibly as drum loop examples ("query by example").

The paper is organized as follows. Section 2 presents the architecture of our drum loop retrieval system and describes the drum loop database used in this study. The next section details the different steps of the automatic transcription of drum loops (feature extraction, classification) and evaluates the transcription performance. Then, Section 4 is dedicated to the spoken onomatopoeia recognition. This section also provides the results of a short perceptual experiment that justifies the choice of the different onomatopoeia for a given drum instrument. Section 5 details the approach followed to align the query with the database examples and provides some evaluation results. Following a section dedicated to implementation and applications issues, Section 7 suggests some conclusions.

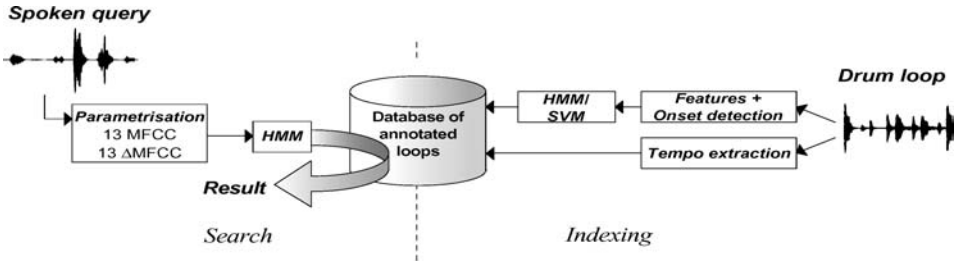


Figure 1. System architecture.

2. System architecture, database and taxonomy

2.1. Components

The overall architecture of the system is depicted in figure 1. It is based on two major components:

- *An automatic transcription tool for drum loops:* this tool consists in automatically indexing a drum loop by segmenting the audio in successive strokes and in recognizing the instrument played for each of these strokes.
- *A retrieval system:* the queries are spoken using onomatopoeia (meaningless short words that sound similar to the targeted percussive sounds). The system transcribes the given query and searches the indexed database for the drum loops that best correspond to the query.

The transcription system is only briefly described in the next section since a more complete and detailed presentation can be found in Gillet and Richard (2004). However, it is important to note that the current system which presents a number of extensions compared to the system described in Gillet and Richard (2004) leads to improved performances. These extensions include the use of Principal Component Analysis (PCA) on the input features and the introduction of time dependencies for the Support Vector Machine (SVM) approach.

2.2. Drum loops database

The database used for this study consists of 315 drum loops containing 5327 strokes. The average duration for a loop is 3.4 seconds. This database was manually annotated using eight basic categories: *bd* for bass drum, *sd* for snare drum, *hh* for hi-hat, *clap* for hands clap, *cym* for cymbal, *rs* for rim shot, *tom* for any other tom of a drum and *perc* for all other percussive instruments with more definite pitch such as congas, djembé or tabla. When two or more instruments are played at the same time, the event is labelled by all the corresponding categories (for example if bass drum and cymbal are hit simultaneously, both labels are attached to the corresponding stroke). Combinations of up to four simultaneous instruments exist in the database (although they are not frequent).

All drum loops were extracted from commercial samples CDs. The loops are representative of different styles including rock, funk, jazz, hip-hop, drum'n'bass and techno and are played on different drum kits including electronic kits. Some loops also have special effects. The loop duration is between two and fifty seconds. If our database is comparable (or larger) in size to the dataset used in most other studies, it is important to emphasize that it contains more important situations commonly encountered in modern audio recordings including simultaneous percussive instruments and audio effects.

2.3. Taxonomy

In theory, all instruments from the eight basic categories can be played simultaneously leading to 2^n possible combinations. In practice (i.e. in our database) only 45 out of 256 combinations are observed. As a consequence, the first taxonomy (*detailed taxonomy*) is defined where each combination is characterized by a label.

For a better analysis of the results, another taxonomy is also used. Considering that the database can be indexed only for querying with onomatopoeia, the number of different categories can be reduced. For the so-called *simplified taxonomy*, each segment is annotated only with the most salient instrument, or with the two most salient instruments. For example, frequent mixtures (such as bass drum + snare drum) will be rendered by a specific onomatopoeia ([ta] in this case), while for other mixtures, only the most salient instrument will be rendered (bass drum + cymbal will be rendered with the onomatopoeia of the bass drum which is [pum], see Section 4 for more details on onomatopoeia selection). Note that the simplified taxonomy is only used to provide an additional interpretation of the results but that the same models have been used for both (i.e. same training and decoding).

3. Transcription and recognition of drum loops

3.1. Features extraction

3.1.1. Segmentation and tempo extraction. Due to the impulsiveness of the drum loops signals, it is appropriate to segment the signal into individual events. Each segment then corresponds to a stroke on a given instrument or to simultaneous strokes on several instruments and can be labelled accordingly. To segment the drum loops signals, an onset detection algorithm based on sub-band decomposition was used (Klapuri, 1999). Since the drum loops signals consist in localized events with abrupt onsets, this algorithm obtains very satisfying results.

Concurrently, the overall tempo is estimated using a slightly modified version of Scheirer's algorithm (Scheirer, 1998). It consists in associating a filter bank with an onset detector in each band and with a robust pitch detection algorithm such as the spectral sum or spectral product (Alonso et al., 2003).

3.1.2. Features set. To select an appropriate features set, a simple classifier (k-Nearest Neighbors) was used. The recognition scores on the different feature sets envisaged were

compared and the results obtained have, for a large part, confirmed those obtained by Herrera et al. (2003). Finally, our features set includes:

- *Mean of 13 MFCC.* The Mel Frequency Cepstral Coefficients (MFCC) including c_0 are calculated on 20 ms frames with an overlap of 50%. The mean is then obtained by averaging the coefficients over the stroke duration. In our work, c_0 is not excluded since it led to better classification performance.
- *4 Spectral shape parameters.* defined from the first four order moments.
- *6 Band-wise Frequency content parameters.* These parameters correspond to the log-energy in six pre-defined bands (in Hertz: [10–70] Hz, [70–130] Hz, [130–300] Hz, [300–800] Hz, [800–1500] Hz, [1500–5000] Hz). These bands were chosen according to a meticulous observation of the frequency content of each drum instrument. Such a choice led to better performances compared to a more classical Bark scale filterbank (as used in Herrera et al. (2003)).

Because some of these parameters are correlated, and to improve the performance of the transcription, a Principal Component Analysis is performed on the data set. As a consequence, the feature set used as input to the classifiers is obtained by a linear transformation of the previous set. Finally, each stroke n is represented by an observation vector of $N = 23$ features: $o_n = (f_{1,n}, f_{2,n} \cdots f_{N,n})$.

3.2. *Classification*

Since drum signals exhibit some kind of context dependencies, an efficient way to integrate these dependencies is to use *Hidden Markov Model (HMM)*. This class of models is particularly suitable for modelling short term time-dependencies and it has been successfully used for a wide variety of problems ranging from speech recognition (Rabiner and Juang, 1993) to tabla signals transcription (Gillet and Richard, 2003). If we consider that the sequence of feature vectors is the output of a Hidden Markov Model, the transcription task is equivalent to the search of the most likely states (strokes), and can be carried out using the traditional Viterbi algorithm (Viterbi, 1967; Forney, 1973).

Another classification approach used is the *Support Vector Machines (SVM)* (Vapnik, 1995). Support Vector Machines non-linearly map (using a Kernel function) their n-dimensional input space into a higher dimensional feature space where the two classes are linearly separable with an optimal margin. SVM have very interesting generalization properties since the decision surface in the data space can be well defined even in the case where a complex surface would be necessary to separate the data.

It is important to emphasize that each drum loop segment can have one or many labels among the n instruments in the kit. This specific problem suggests two possible approaches:

- *One 2^n -ary classifier.* In a first approach, only one classifier is used, in which each possible combination of strokes is represented by a distinct class. Our study uses 8 instruments, implying thus the use of a 255 classes classifier. Since only 45 different combinations of strokes are present in our database, such a classifier in our case includes only 45 classes.

- *n binary classifiers.* In a second approach, one binary classifier per instrument is trained. This binary classifier decides whether the instrument is played or not in each segment.

Finally, due to the high variability of the data, a *hierarchical approach* was also tested. Instead of using one generic classifier, four classifiers "specialized" in four different kinds of drumkits were trained by splitting the training database according to style/drumkit criteria. The four categories roughly correspond to four styles of drum kits (electronic, light/heavy acoustic kits, hip-hop).

Further details on the transcription system can be found in Gillet and Richard (2004). However, the results published here (see Section 3.3) are slightly improved thanks to:

- the use of a Principal Component Analysis (PCA) on the input features. PCA is often used in classification applications in order to reduce the dimensionality of the feature space (Partridge and Jabri, 2000). In fact, it may be used to "de-noise" the signal in the sense that the most relevant information is concentrated in the first few components of the transformed feature vectors which correspond to directions of maximum energy. PCA was performed as in Essid et al. (2004). Combined with our HMM models that use mixtures of Gaussians with diagonal covariance matrix as probabilities densities associated to each state, PCA on the input features leads to slightly improved performances of the drum loop transcription system.
- the introduction of time dependencies for the Support Vector Machine (SVM) approach. Practically, it consists of replacing the feature vector of one stroke $o_n = (f_{1,n}, f_{2,n}, \ldots, f_{N,n})$ (see Section 3.1) by a combined vector containing also the features of the previous stroke $[o_n, o_{n-1}]$.

3.3. Results

The results obtained on our dataset, using the ten-folds validation protocol are summarized in Table 1. This protocol consists in splitting the whole database in 10 subsets randomly selected and in using nine of them for training and the last subset (i.e. 10% of the data) for testing. The procedure is then iterated by rotating the 10 subsets used for training and testing. The results are computed as the average values for the ten runs.

It can be observed that SVM clearly outperforms the non-hierarchical HMM approach for both taxonomies. This may be explained by the fact that the rather simple acoustic model used with HMM cannot cope with the high variability of the dataset, and also that the HMM models need a large dataset to be reliably trained.

This is confirmed by the experiment implementing a hierarchical approach. When a hierarchical model is used for HMM, performances of both approaches are comparable. Therefore, a two step approach permits to split the data according to the drum kit used and thus to decrease the variability of data within a given class which is appropriate for HMM and GMM (Gaussian Mixture Model).

The use of the PCA has no significant effect with SVM models, but slightly improve the results of the HMM classifiers, especially for the detailed taxonomy.

Table 1. Drum instruments recognition results.

Taxonomy	Detailed (%)	Simplified (%)
one 2^n-ary classifier		
HMM, 3-grams, 1 mixture	59.6	79.2
HMM, 3-grams, 2 mixtures	57.5	76.8
HMM, 4-grams, 1 mixture	59.8	78.2
SVM	64.9	83.0
SVM with context	68.9	85.9
n binary classifiers		
HMM, 3-grams, 1 mixture	44.3	64.0
HMM, 3-grams, 2 mixtures	42.7	65.6
HMM, 4-grams, 1 mixture	34.4	53.6
SVM	64.8	83.8
SVM with context	68.1	86.4
Hierarchical approach using drum kit recognition		
HMM, 3-grams, 1 mixture	62.7	82.9
HMM, 3-grams, 2 mixtures	59.9	83.6
HMM, 4-grams, 1 mixture	60.8	77.5

4. Recognition of onomatopoeias in spoken queries

4.1. Selection of onomatopoeias

Contrary to rhythmic instruments such as North Indian Tabla, in which there is a well-defined and widely used set of vocables for each stroke of the instrument (Patel and Iversen, 2003), there is no commonly accepted set of vocables used by musicians to denote the instruments of the drum kit, probably due to the comparatively limited importance of oral tradition in Western music. To allow the use of natural queries, a trade-off had to be found between a set of onomatopoeias that is clearly imitating the instruments of the drum kit; and a set of syllables that are phonetically distant. The set of onomatopeia finally chosen is given in Table 2.

Table 2. Vocabulary (onomatopoeias) used for the spoken queries.

Instrument	Onomatopoeia
Bass drum	[pum]/[bum]
Cymbal, hi hat	[ti]/[ts]
Snare drum,	[t$\int$ a]
Snare drum + Bass drum mixture	[ta]
Tom, other percussive instrument	[do]/[dom]/[tom]

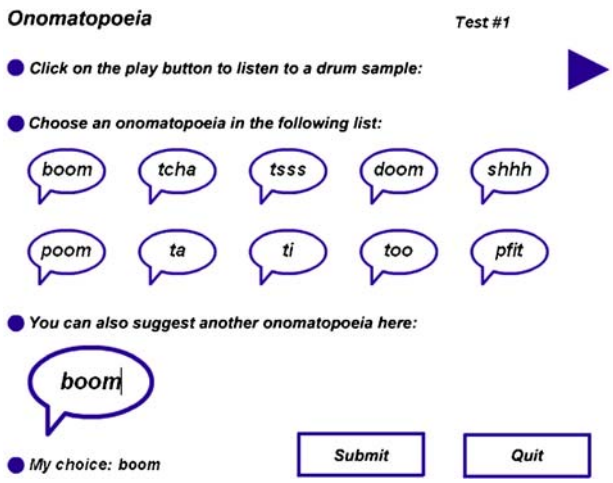


Figure 2. Interface of the web-based perception experiment.

4.2. Perception experiment

This choice was validated by a web-based perception experiment (see figure 2 for the interface of this experiment).

Forty (40) drum samples were extracted from the loops: 7 bass drums, 13 snare drums, 8 hi-hats, 7 toms/percussive instruments with definite pitch, and 5 mixtures of bass-drums + snare drums.

The experiment then consisted in randomly selecting a sample from the forty pre-selected drum samples and in asking the subject to pick the onomatopoeia that best described it. The subject was provided with a list of 10 common onomatopoeias: (*boom* [bum], *poom* [pum], *tcha* [t$\int$ a], *tss* [ts], *shh* [$\int$], *doom* [dum], *ta* [ta], *ti* [ti], *too* [tu], *pfit* [pfit]), but could also suggest new onomatopoeias by typing in a new word. When the subject submitted their choice, a new drum sound was proposed, and the experiment continued until the subject clicked on the **Quit** button.

A total of 572 answers were collected from 28 different subjects.

The results are gathered in Table 3. The onomatopoeias in bold are those we chose for the corresponding instrument.

In the *non-significative* category are gathered onomatopoeias with less than 3 occurrences. Participants formulated relatively often very odd onomatopoeias, probably due to the lack of commonly accepted vocables to denote drum sounds.

If the results of this experiment confirm the relevance of our chosen onomatopoeia, they also show that:

- The same onomatopoeia [ta] and [t$\int$ a] are often used to denote the snare drum and the mixture of snare drum and bass-drum. This can be explained by the fact that the subject tends to link an onomatopoeia to the most salient instrument.

Table 3. Results of the perception experiment: the chosen onomatopoeias to represent an instrument are given in bold.

Instrument	Onomatopoeia	Frequency	Onomatopoeia	Frequency
Bass drum	**[pum]**	**36**	**[bum]**	**16**
	[tu]	17	Non-significative	29
Snare drum	**[t∫ a]**	**48**	[dum]	11
	[ta]	34	[pfit]	10
	[tu]	18	[pum]	7
	[ts]	14	[bum]	7
	[ti]	12	[tok]	5
	[∫]	11	Non-significative	14
Hi-hat or	**[ts]**	**48**	[∫]	5
Cymbal	**[ti]**	**16**	Non-significative	29
Tom or	**[tom]**	**33**	[tum]	12
other percussive instrument	**[dom]**	**21**	[bum]	11
	[pum]	21	Non-significative	31
Bass drum +	**[ta]**	**20**	[ts]	7
snare drum mixture	[t∫ a]	17	Non-significative	12

- Onomatopoeia with the vowel [u] are relatively often formulated for toms or other percussive instruments with a definite pitch, and thus, can lead to confusion with the onomatopoeia [pum] chosen for the bass-drum.

4.3. Recognition of spoken onomatopoeias

To train and to evaluate the recognition of spoken onomatopoeias, a query database was built. It consisted of 60 queries recorded using a standard home-studio microphone, according to the following protocol: a loop was picked randomly in the database, listened to 4 times, then was "performed" with onomatopoeia by one same speaker. A tempo difference (interpretation speed ratio) varying from 0.8 to 1.1 times the original tempo was noticed. Moreover, deletions are occurring very frequently, especially when the original loop have "flams" or "rolls"—that is to say quickly repeated snare drum strokes—and of course only the most salient stroke is performed.

This corpus was subsequently manually segmented, and annotated with the symbols corresponding to the instruments imitated by the onomatopoeias; that is to say, the segments containing [pum] or [bum] in a query were given the same label, "*bass drum*." The whole database contains 762 onomatopoeias.

The recognition of the onomatopoeias contained in the query is performed by a simple Bakis (left-right) HMM model with 4 states; the probability distribution associated to each state being a mixture of 2 Gaussians. This study is limited to a speaker-dependent recognition due to the lack of appropriate training data. In this context, such a configuration

appears to be well adapted but it would obviously need to be extended for a speaker-independent approach.

The features used for the recognition are the 13 MFCC + 13 ΔMFCC. Each model is trained from the database using the Expectation-Maximisation (EM) algorithm. These models are then connected to form a single graph. Recognition is achieved by maximizing the likelihood of the sequence of features vectors of the query by the Viterbi algorithm. Another model using an extra state modelling silences was tested, but it was outperformed by the model described, in which the silence that may follow a vocable is modelled as a part as the vocable itself.

The output of the query transcription system is a sequence of couples (t_i, S_i), where S_i is the stroke (or compound stroke, like *bass drum* + *snare drum*) played at time t_i. The recognition rate of our speaker-dependent system is 90.7% on our database using the ten-fold evaluation protocol.

5. Scoring and aligning the query

Several approaches have been proposed in the literature to compare a music query with a database of musical phrases. Most of them compare the query to each entry of the database using a distance on descriptors. Typical distances range from a simplified melodic contour to a complex hierarchic description of the melody (Ghias et al., 1995; Sonoda et al., 1998; Rolland et al., 1999). Another approach summarizes a whole song by a HMM, and computes the likelihood of the query from this model (Shifrin et al., 2002). Some improvements can be achieved by modelling the common errors that are likely to occur when a user with no musical training is humming a melody (Jin and Jagadish, 2002).

However, these approaches are not suitable for the matching of drum queries. Firstly, because the notion of melody and melodic contour is useless when dealing with drum loops for which only the instrument used for each note is important. Secondly, because most of these studies are indeed ignoring the rhythmic information and only focus on the sequence of intervals between notes; and finally because in our case the indexed patterns are very short. If it is possible to use HMMs or n-grams statistics to produce a generative model of the queries on a whole musical piece containing 500 notes, it is not possible to do so on a short loop containing 20 strokes.

We consequently chose a novel approach based on a generative statistical model of the loop interpretations. As such, the query task can be reformulated as 'find the loop(s) in the database that is (are) most likely to be performed as the given spoken onomatopoeia query'.

5.1. Statistical model of interpretation

We propose here a statistical model taking into account the various edition operations likely to occur when a complex rhythmic phrase is interpreted with onomatopoeias: the non-formulation of a stroke contained in the loop (*deletion*), the formulation of a stroke which is not contained in the searched loop (*insertion*), and the approximative formulation (*substitution*) of a note contained in the searched loop, possibly with timing errors (*alignment*).

This model allows us to compute the probability that a query is actually a good interpretation of one of the loops contained in the database, in other words the likelihood of the interpretation q knowing the loop l. Let e be the sequence of edition operations made by the user when formulating the query while having in mind the searched loop. This sequence will be considered as a hidden variable such that:

$$P(q \mid l) = \sum_{e} P(q, e \mid l)$$

where $q = (t_i, Q_i)_{i\in[1,M]}$ is a query and $l = (u_j, L_j)_{j\in[1,N]}$ a loop in the database. For the sake of clarity, we assume, in a first step, that q and l are at the same tempo, and that the first event of q is an interpretation of the first event of l.

5.1.1. Model parameters. The parameters of our model are:

- The likelihood of the interpretation of each stroke b , knowing that it is not present in the loop $P(\{b\} \mid \emptyset)$.
- The likelihood of the deletion of each stroke a, knowing that it is present in the loop $P(\emptyset \mid \{a\})$. (For example, it is very likely that the user will not formulate a hi-hat).
- A distribution for the timing errors $P_a(t)$ from which can be derived the likelihood of a timing error of t between a stroke and its interpretation.
- A distribution for the duration of deleted (resp. inserted) strokes $P_s(t)$ (resp. $P_i(t)$), giving the likelihood that a stroke of duration t is deleted (resp. inserted). Such errors are likely with short strokes, for example in rolls or flams.

Let A be a set of strokes and B its interpretation by the user. The likelihood of B knowing A is given by:

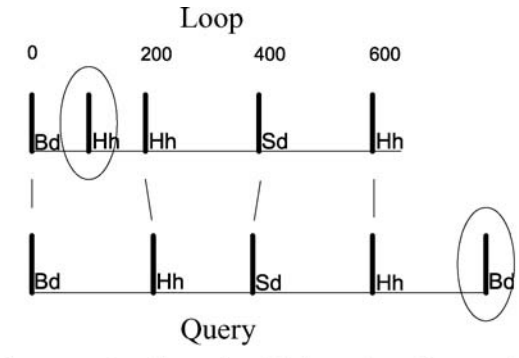
$$P(B \mid A) = \prod_{stroke\in S} P(\{stroke\} \cap B \mid \{stroke\} \cap A)$$

where S is the set of possible strokes $S = \{bd, sd, cym, hh, tom, perc, rs, clap\}$. Let (u, A) be a stroke A at time u in a loop, and (t, B) an interpretation of A at time t. If we consider that time-aligning errors are independent of the confusions between strokes, the likelihood of (t, B) knowing (u, A), is:

$$P((t, B) \mid (u, A)) = P(B \mid A)P_a(t, u)$$

where $P_a(t, u)$ is the likelihood of a timing error between the two events. It seems appropriate to assume that this likelihood only depends on the absolute time difference between the two events and that it follows an exponential distribution of the form $P_{exp}(x) = \frac{1}{C} exp^{-\frac{x}{C}}$ where C is a constant. $P_a(t, u)$ can then be rewritten as:

$$P_a(t, u) = P_{\exp}(|t - u|) = \frac{1}{C} \exp^{-\frac{|t-u|}{C}}$$



	Bd 0	Hh 100	Hh 200	Sd 400	Hh 600
Bd 0					
Hh 220					
Sd 390					
Hh 600					
Bd 800					

e_i	align.	deletion	align.	align.	align.	insertion
$e_{Li}(l,q)$	$(\{bd\},0)$	$(\{hh\},100)$	$(\{hh\},200)$	$(\{sd\},400)$	$(\{hh\},600)$	$\emptyset$
$e_{Qi}(l,q)$	$(\{bd\},0)$	$\emptyset$	$(\{hh\},220)$	$(\{sd\},390)$	$(\{hh\},600)$	$(\{bd\},800)$

Figure 3. Alignment.

Using the same notations, $P((t, B) \mid \emptyset)$ is the likelihood of an insertion of B at time t. If this onomatopoeia has a duration equal to d, then $P((t, B) \mid \emptyset) = P(B \mid \emptyset)P_i(d)$, where $P_i(d)$ is the likelihood of the insertion of an onomatopoeia of duration d. Similarly, $P(\emptyset \mid (u, A)) = P(\emptyset \mid A)P_s(d)$, where $P_s(d)$ is the likelihood of the deletion of a stroke of duration d. This is aimed at penalizing the insertions or deletions of long strokes, while the insertion or deletion of short strokes (for example in flams or rolls) is more common and should not be penalized. In this study we used $P_s = P_i = P_a = P_{exp}$ to reduce the number of parameters of the system.

5.1.2. Alignment. The aim of the alignment between the loop and the interpretation is to find the sequence of edition operations e^* maximizing the likelihood of $P(q, e \mid l)$:

$$P(q, e \mid l) = \prod_i P(e_{Qi} \mid e_{Li})$$

where the sequences $(e_{Qi})_{i\in[1,E]}$ and $(e_{Li})_{i\in[1,E]}$ describe the alignment resulting from the edition operations e^* (refer to figure 3).

The search of such an optimal alignment is possible with dynamic programming:

Initialization. $dp(0, 0) = 1$.

Recursion. $\forall i \in [1, M], j \in [1, N]$:

$$dp(i, j) = \max\{dp(i-1, j-1)P((t, Q) \mid (u, L)), dp(i, j-1)P((t, Q) \mid \emptyset), dp(i-1, j)P(\emptyset \mid (u, L))\}$$

The sequence e^* is obtained by backtracking and $P(q, e^* \mid l) = dp(M, N)$. The joint-probability $P(q, e^* \mid l)$ obtained can be used as a score, and as an approximation of the marginal probability $P(q \mid l)$ as suggested by Shalev-Shwartz et al. (2002).

It is worth to mention that by using the opposite of the log-likelihood instead of the likelihood, $D(i, j) = -\log dp(i, j)$, we obtain the classic recursion used to compute an edit distance. The aligning cost is then equal to $C((t, B), (u, A)) = -\log P(B \mid A) + C|t-u|$, the deletion cost is equal to $C(\emptyset, (u, A)) = -\log P(\emptyset \mid A) + Cd$ and the insertion cost is equal to $C((t, B), \emptyset) = -\log P(B \mid \emptyset) + Cd$. Using the logarithmic form has several advantages (decreased complexity, robustness to numeric errors) and is therefore the distance kept for this study.

5.1.3. Parameters initialization. The parameters of the model were empirically obtained by aligning a set of queries and the corresponding loops in the database, and by counting the most common errors: $P(\emptyset \mid \{bd\}) = 3\%$, $P(\emptyset \mid \{sd\}) = 10\%$ (same for rs and clap), $P(\emptyset \mid \{hh\}) = 70\%$ (same for cym), $P(\emptyset \mid \{perc\}) = 10\%$ (same for tom), $P(\{hh\} \mid \emptyset) = 15\%$, 3 % for all the other; $C = 15$.

5.2. Tempo and loop start alignment

In the maximization computed previously, we assume that the query is an interpretation of the whole loop and that both have the same tempo. However, it is likely that the query is just an interpretation of its first notes. Therefore to avoid too short alignments, a scaling factor is used:

$$D^*(query, loop) = \min_{j \in [1,N]} \frac{D(M, j)}{\sqrt{L^2 + j^2}}$$

Furthermore, it is necessary to take into account the fact that the query is not always an interpretation from the beginning of the loop, but might be an interpretation of a fragment located at any time onset u_{start} within the loop.

$$D'(query, loop) = \min_{start \in [1,N-1]} D^*(query, \left(u_j - u_{\text{start}}, L_j\right)_{j \in [start,N]})\Big)$$

It is also necessary to deal with the fact that the query is not always formulated at the same tempo as the loop:

$$D(query, loop) = \min_{\lambda \in [0.9,1.2]} D\Big(query, (\lambda u_j, L_j)_{j \in [1,N]})\Big)$$

where λ is the tempo scaling ratio. Note that the range of tolerated tempo scaling ratios is [0.9, 1.2] and that the minimum distance is only computed on a discrete set of λ values. The complexity of this algorithm is thus $O(LM^2)$. On a Pentium II 350 MHz, a query can be processed in 0.2s for a database of 315 loops.

5.3. *Query by example*

The model described can also be used in a "Query by example" system where the query is a drum loop or part of a drum loop. In this case, the likelihoods $P(l_1 \mid l_2)$ expressing the substitution cost between two strokes have been symmetrized so that the measure D provided by the recursion can be interpreted as a distance.

5.4. *Results and evaluation*

5.4.1. Graphical display. For a query d, the matching candidates are $\mathcal{E}(q, \tau) = \{L, D(q, L) < \tau\}$, where τ is a variable threshold to be tuned by the user.

The results of the queries are graphically represented (see figure 4). The first 30 loops giving the best scores are plotted in a two-dimensional space using multi-dimensional

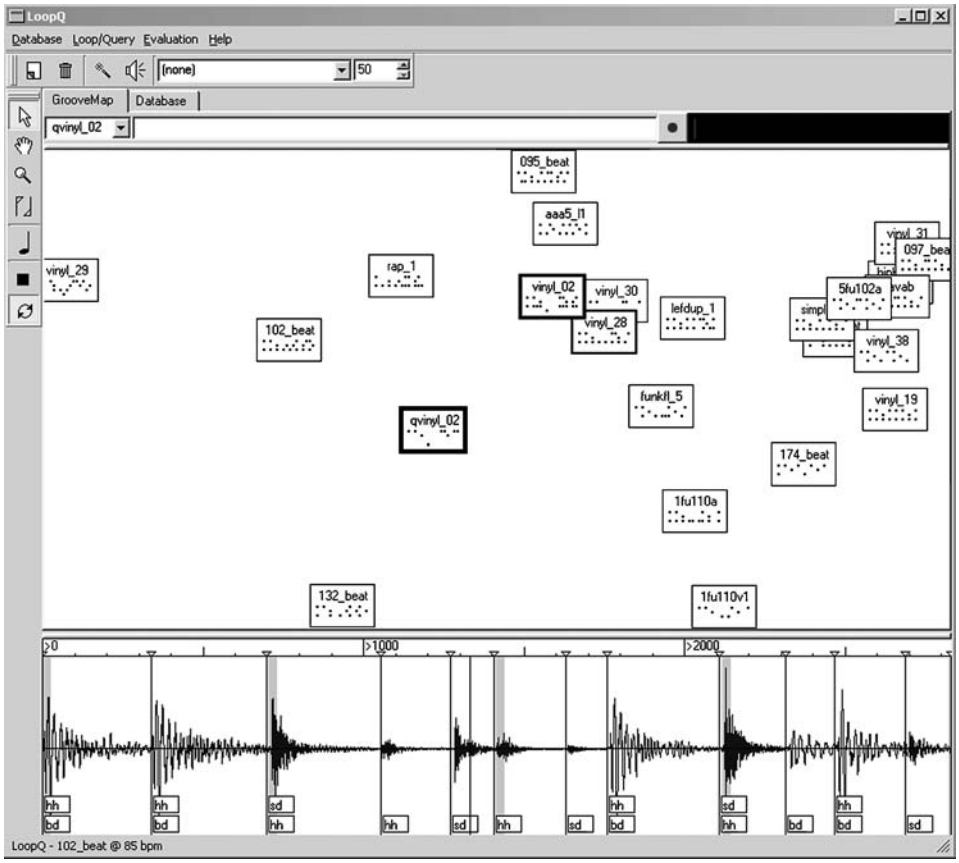


Figure 4. Graphical representation of the query results.

scaling from the distance matrix. It is therefore possible to quickly pick among the results those which are variations of the same loop, or those corresponding to the same rhythm played with a different kit or at different tempi.

5.4.2. Evaluation. Evaluation of a Music Information Retrieval (MIR) system is not a straightforward task (see (Downie, 2003) for a collection of white papers on MIR evaluation).

The chosen procedure for the evaluation of our query system is based on the generation of a virtual database of queries and is described below:

1. A loop l_i was randomly selected from the database.
2. A segment q_i was randomly extracted from this loop; its length varying from 3 to 5 seconds.
3. A query was synthesized by concatenating onomatopoeia contained in a test database (compound of 80 instances of each of the onomatopoeia), using the generative model described in Section 5.1.
4. This query was transcribed by the onomatopoeia recognition system.
5. The loops giving the best score were searched and selected, using a given threshold τ.
6. Steps 1 to 5 of the procedure are iterated 500 times

We used the traditional information retrieval performance measures: precision and recall. In a text retrieval system, precision is the ratio between the number of relevant documents $RETREL$ the system retrieved for a specific query and the total number of documents RET retrieved for this query; while recall is the ratio between the number of relevant texts $RETREL$ retrieved; and the total number of relevant texts REL in the database. In our system, the query is the sequence of onomatopoeia, and searched documents are drum loops. It is important to note that our evaluation procedure assumes that for a given query, only one loop is relevant (REL is always equal to one). This is not always the case with our database in which very similar loops are present. Another relevance measure, such as those given by subjective perceptual tests, could have been used and would have given a higher value for REL.

The precision of a single query is thus 0 if the loop searched is not present in the matches; $1/N$ where N is the number of matches otherwise; while the recall of a single query is 0 if the loop searched is not present in the set of matches; 1 if it is present. For each value τ of the threshold, a couple of precision/recall values could be computed by averaging the precision/recall ratios of each single query:

$$Recall(\tau) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\mathcal{E}(q_i,\tau)}(l_i)$$

$$Precision(\tau) = \frac{1}{N}\sum_{i=1}^{N} \frac{\mathbf{1}_{\mathcal{E}(q_i,\tau)}(l_i)}{|\mathcal{E}(q_i,\tau)|}$$

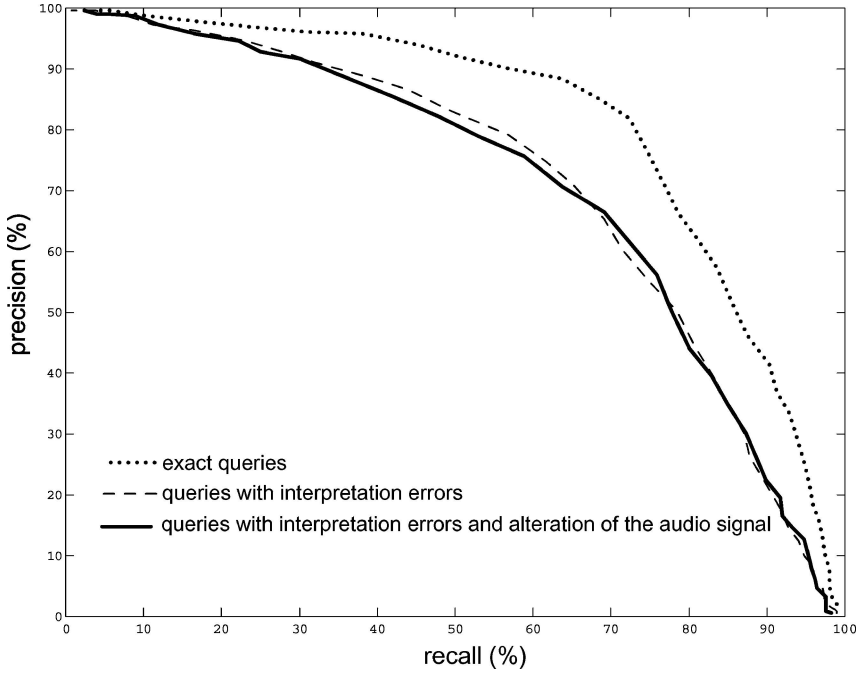


Figure 5. Precision/recall curves.

where $\mathcal{E}(q_i, \tau)$ is the set of matching loops returned for the query q_i with the distance threshold τ, and $|X|$ denotes the cardinal of the set X.

Three sets of results were obtained, from which precision/recall curves were plotted (see figure 5). Firstly, no time alignment mistakes or substitutions are present, that is to say, the synthetic queries are exact interpretations of the loop (dotted line). Secondly, time alignment errors and substitutions/deletion/insertion are produced using the generative model described in Section 5.1 (dashed line). Finally, the sound signal at the input of the speech recognition module is altered with a slight reverberation and resampling at 8 kHz (with a low-pass filtering at 4 kHz), aiming at simulating the use of low-end microphones (solid line).

A recall rate of 80% can be obtained with a precision of 50%. This result is quite satisfying since our database contains very similar loops—or identical loops performed with different kits. However, the lack of common databases for evaluation, and the absence of other similar works does not allow us to compare these results to other systems. It will also be important to confirm these results on larger databases.

6. Implementation and applications

The different modules presented in this article, loop segmentation, transcription, and query were integrated in a single, graphical application, **LoopQ**, developed in C++ with the Qt

library. The audio files themselves are stored as standard WAV files on the computer; the tempo, style and transcription are gathered in a single XML file, an example of the structure of which is given below:

```
<loopbase>
  <vocabulary>
    <word>bd</word>
    <word>sd</word>
      ...
  </vocabulary>

  <styles>
    <style>hip-hop</style>
    <style>electro</style>
      ...
  </styles>

  </loop><loop filename='loops/eye.wav' style='electro'
    bpm='94' sr='44100'>
    <stroke in='0' out='6957'>
      <label>bd</label>
      <label>hh</label>
    </stroke>
    <stroke in='6957' out='13851'>
      <label>hh</label>
    </stroke>
      ...
```

The “vocabulary”, that is to say the set of single strokes names can be modified, thus, it is possible to support new strokes and new drum instruments; as long as the features set used is still relevant for this instrument.

This application allows an intelligent management of a database of annotated drum loops. Everytime a drum loop is imported and added to the database, its tempo, as well as a transcription is immediately made available. Optionally, after having validated the transcription of an imported loop, the user has the possibility to retrain the whole recognition models. The performances of the retraining step is improved by a cache that stores the features of each loop - the pre-calculated features could as well be directly saved in a RIFF chunk of the WAV files. However, both the EM algorithm for HMMs and the SVM learning algorithm require to be re-run on the whole dataset.

The query by onomatopoeia allows to efficiently search specific loops in a large drum loops database, and the graphical representation offers another way to easily navigate in the database. After having submitted a query, the best matches are graphically represented. The user can play one of the result simply by moving the mouse cursor on one of the small boxes representing the matches. A click on this box will perform a “query by example”, that is to say, display another set of similar loops, ordered by similarity. It is therefore possible to graphically explore the database, using similarities. It enables a creative use of the loops, for instance by showing that two loops of different styles and tempi have indeed the same transcription and could be interchanged.

7. Conclusion

Query and indexing tools are necessary to facilitate the use of drum loops database. However, most of the music transcription and query by humming systems are focused on melody. This article described a novel way to query drum loops database using spoken onomatopoeias or drum loop examples based on an efficient indexing system. Using support vector machines on a relevant set of cepstral and spectral features, promising results (86.4% using a simplified taxonomy) were obtained for the transcription task. A statistical model of the interpretation of rhythm allows the search of the most relevant loops with a good precision/recall trade-off. Finally, all these features were integrated in a single environment enabling an easy and user-friendly exploration of a drum loops database. Future work will be dedicated to the improvement of the speech recognition front-end and to the development of a combined HMM/SVM approach for the transcription system. Finally, new modalities will be considered to query the system (MIDI input, bass lines for which a matching rhythmic accompaniment must be found).

References

Alonso, M., David, B., and Richard, G. (2003). A Study of Tempo Tracking Algorithms from Polyphonic Music Signals. In *Proceddings of 4th COST276 Workshop*, Bordeaux, France.

Alonso, M., David, B., and Richard, G. (2004). Tempo and Beat Estimation of Musical Signals. In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR2004)*, Barcelona, Spain.

Byrd, D. and Crawford, T. (2002). Problems of Music Information Retrieval in the Real World. *Information Processing and Management*, 38, 249–272.

Chen, J.C.C. and Chen, A.L.P. (1998). Query by Rhythm: An Approach for Sond Retrieval in Music Databases. In *Proceedings of the 8th IEEE Workshop on Research Issues on Data Engineering (RIDE1998)*, Orlando, Florida, pp. 139–146.

Downie, S. (2003). The mir/mdl evaluation project white paper collection. http://music-ir.org/evaluation/wp.html.

Essid, S., Richard, G., and David, B. (2004). Musical Instrument Recognition on solo Performances. In *Proceedings of the 12th European Conference on Signal Processing (EUSIPCO2004)*, Vienna, Austria.

FitzGerald, D., Coyle, E., and Lawlor, B. (2002). Sub-band Independent Subspace Analysis for Drum Transcription. In *Proceedings of the 5th International Conference on Digital Audio Effects (DAFX'02)*, Hamburg, Germany.

Forney, G. D. (1973). The viterbi algorithm. In *Proc. IEEE*, pp. 268–278.

Ghias, A., Logan, J., Chamberlin, D., and Smith, B. (1995). Query by Humming: Musical Information Retrieval in an Audio Database. In *Proceedings of the 3rd ACM International Conference on Multimedia (ACM Multimedia'95)*, San Francisco, California.

Gillet, O. and Richard, G. (2003). Automatic Labelling of Tabla Signals. In *Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR2003)*, Baltimore, Maryland.

Gillet, O. and Richard, G. (2004). Automatic Transcription of Drum Loops. In *Proceedings of the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2004)*, Montreal, Quebec.

Goto, M. (2001). An Audio-Based Real-Time Beat Tracking System for Music with or Without Drum-Sounds. *Journal of New Music Research*, 30(2), 159–171.

Herrera, P., Amatriain, X., Battle, E., and Serra, X. (2000). Towards Instrument Segmentation for Music Content Description: A Critical Review of Instrument Classification Techniques. In *Proceedings of the 1st International Symposium on Music Information Retrieval (ISMIR2000)*, Plymouth, Massachusetts.

Herrera, P., Dehamel, A., and Gouyon, F. (2003). Automatic Labeling of Unpitched Percussion Sounds. In *Proceedings of the 114th Audio Engineering Society Convention (AES'2003)*, Amsterdam, The Netherlands.

Jin, H. and Jagadish, H. (2002). Johny Can't sing: A Comprehensive Error Model for Sung Music Queries. In *Proceedings of the 3rd International Symposium on Music Information Retrieval (ISMIR2002)*, Paris, France.

Klapuri, A. (1999). Sound onset Detection by Applying Psychoacoustic Knowledge. In *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP1999)*, Phoenix, Arizona.

Kornstädt, A. (1998). Themefinder: A Web-Based Melodic Search Tool. *Computing in Musicology*, 11, 231–236.

Laroche, J. (2003). Efficient Tempo and Beat Tracking in Audio Recordings. *Journ. of Audio. Eng. Soc.*, 51(4).

McDonald, S. and Tsang, C. (1997). Percussive sound Identification Using Spectral Centroid Trajectories. In *Proceedings of the 1997 Postgraduate Research Conference*, University of Western Australia.

McNab, R., Smith, L., Bainbridge, D., and Witten, I. (1997). The New Zealand Digital Library Melody Index. *D-Lib Magazine*. http://www.dlib.org/dlib/may97/meldex/05witten.html.

Partridge, M. and Jabri, M. (2000). Robust principal component analysis. In *Proceedings of the 2000 IEEE Signal Processing Society Workshop*, Sydney, Australia, pp. 289–298.

Patel, A. and Iversen, J. (2003). Acoustic and Perceptual Comparison of Speech and Drum Sounds in the North Indian Tabla Tradition: An Empirical Study of Sound Symbolism. In *Proceedings of the 15th International Congress of Phonetic Sciences (ICPhS)*, Barcelona, Spain.

Paulus, J. and Klapuri, A. (2003). Conventional and Periodic n-Grams in the Transcription of Drum Sequences. In *Proceedings of the 2003 IEEE International Conference on Multimedia and Expo (ICME2003)*, Baltimore, Maryland.

Rabiner, L. and Juang, B. (1993). *Fundamentals of Speech Recognition*. NJ: Englewood Cliffs.

Raphael, C. (2002). A Hybrid Graphical Model for Rhythmic Parsing. *Artificial Intelligence*, 137, 217–238.

Rolland, P., Raskinis, G., and Ganascia, J. (1999). Musical Content-Based Retrieval: An Overview of the Melodiscov Approach and System. In *Proceedings of the 7th ACM International Conference on Multimedia (ACM Multimedia'99)*, Orlando, Florida, pp. 81–84.

Scheirer, E. (1998). Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, 103(1), 588–601.

Shalev-Shwartz, S., Dubnov, S., Friedman, N., and Singer, Y. (2002). Robust Temporal and Spectral Modeling for Query by Melody. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2002)*, Tampere, Finland.

Shifrin, J., Pardo, B., Meek, C., and Birmingham, W. (2002). Hmm-Based Musical Query Retrieval. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL2002)*, Portland, Oregon.

Sillanpää, J., Klapuri, A., Seppänen, J., and Virtanen, T. (2000). Recognition of acoustic noise mixtures by combined bottom-up and top-down approach. In *Proceedings of the 10th European Conference on Signal Processing (EUSIPCO2000)*, Tampere, Finland.

Sonoda, T., Goto, M., and Muraoka, Y. (1998). A www-Based Melody Retrieval System. In *Proceedings of the 1998 International Computer Music Conference (ICMC'98)*, Ann Arbor, Michigan, pp. 349–352.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Viterbi, A.J. (1967). Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm. In *IEEE Trans. Informat. Theory*, pp. 260–269.

Zils, A., Pachet, F., Delerue, O., and Gouyon, F. (2002). Automatic Extraction of Drum Tracks from Polyphonic Music Signals. In *Proceedings of the 2nd International Conference on Web Delivering of Music (WEDELMUSIC'2002)*, Darmstadt, Germany.