INDEXING AND QUERYING DRUM LOOPS DATABASES

Olivier Gillet and Gaël Richard

GET-TELECOM Paris
37, rue Darreau
75015 Paris, France
[olivier.gillet, gael.richard]@enst.fr

ABSTRACT

Large databases of short drums signals, known as drum loops, are widely used for the composition of modern music. This paper presents a complete and integrated system to index and query such databases. The transcription task necessary to index the database can be performed with a range of different classifiers such as Hidden Markov Models (HMM) or Support Vector Machines (SVM) and achieves a 89.9% correct recognition rate on a simplified taxonomy. Queries can be formulated on this indexed database with spoken onomatopoeia - short meaningless words imitating the different sounds of the drum kit. The syllables of spoken queries are recognized and a relevant statistical model allows the comparison and alignement of the query with the rhythmic sequences stored in the database. This same model can be used to provide a distance measure and allows queries by example. Query results can be graphically displayed and grouped by similarity.

1. INTRODUCTION

Pre-recorded audio databases of drum loops are widely used in the production of modern music, especially in genres such as hip-hop, r'n'b, house, drum'n'bass or techno. These databases, available as collections of CDs or CD-ROMs, gather a large number of short drum signals which are used as a raw material for composition: Either individual notes are extracted and rearranged with music software such as ReCycle, or the whole signal is repeated to build an entire drum track - hence the name, *loop*. Most of the drum-loops collections do not provide any other information than the tempo and style of each loop. As a result, the musician has no other alternative than browsing the entire CD and listening to each individual file. There is therefore a need for more elaborated retrieval and indexing tools that will provide content-based methods in a user-friendly interface, to efficiently search these databases.

An important aspect of such a tool is the necessity to obtain an automatic transcription of the drum loop signals - the indexing stage. Most of the work in the domain of audio transcription is dedicated to melodic instruments (see for instance [8] 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. Gouyon & al. [9] evaluated several classifiers and feature sets for natural and electronic drum signals recognition: these approaches proved to be successful but were limited to isolated sounds. A specifity of drum loops signals is that each event can be produced by simultaneous strokes on different instruments (for example bass drum and hi-hat). Another specifity 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.

Similarly to audio indexing, most of the works in music retrieval focus on melody and on query by example. A very popular approach called "Query by humming", aims at retrieving music files from a sung melody. Various systems are already implemented and show promising results ([3], [13]). However, most of them require a high-level representation of the whole searched database, for example as a collection of MIDI files, and only take into account melodic information. In the context of percussive signals where melody is hardly present, a different approach needs to be followed. One of the most natural ways of describing a pure rhythmic content is by means of spoken onomatopoeia - short meaningless words imitating the different sounds of the percussive instruments (drums in this context). The use of spoken onomatopoeia is a rather new approach to drum pattern retrieval which was presented in [5], independently of the works by Nakana & al. [14] and Kapur & al. [10].

This paper details and extends our first works presented in [5]. It is organized as follows. Section 2 presents the overall system architecture of our drum loop retrieval system and describes the new database used in this study. The next section details the different steps of the automatic transcription of drum loops (features extraction, classification) and evaluates the transcription performance. Then, section 3 is dedicated to the spoken onomatopoeia recognition, using a new speaker-independent system. Section 4 describes

ISBN: 952-15-1364-0

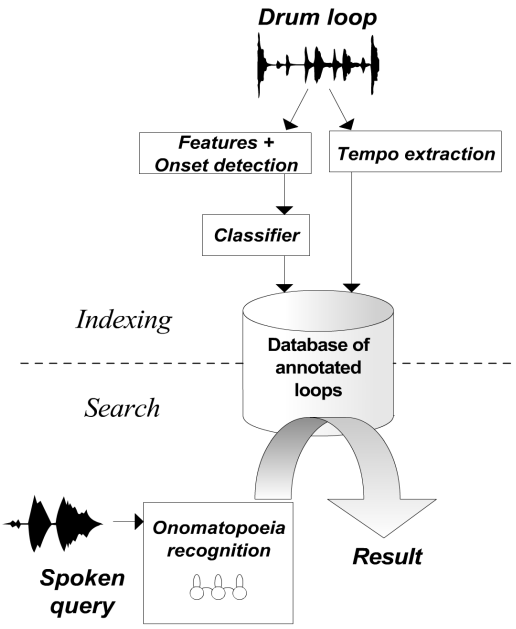

Fig. 1. System architecture

in details the approach followed to align the query with the loops contained in the database, and provides some evaluation results. Following a section dedicated to implementation and applications issues, section 6 suggests some conclusions.

2. SYSTEM ARCHITECTURE, DATABASE AND TAXONOMY

2.1. Components

The overall architecture of the system is depicted in figure 1. The first important component is the automatic drum loops transcription (indexing) module. Each drum loop is individually indexed by segmenting it in successive strokes and by recognizing the instrument or combination of instruments played for each of these strokes. The second important component is the retrieval system: the spoken queries are recognized into a sequence of onomatopoeia, each of them associated to a target drum sound. The indexed database is searched for the drum loops that best correspond to the query.

The rest of this section will focus on the different improvements and extensions of our first transcription system presented in [4] and in [5].

2.2. Drum loops database

Our previous work used a database, B_1 consisting in 315 loops (5327 strokes). We gathered a new collection of loops B_2, containing 128 loops (2685 strokes). This new set includes loops downloaded from the web or extracted from drum solos occuring in songs from the RWC Popular Music Database [6]. The loops from B_1 and B_2 are representative of different styles including rock, funk, jazz, hip-hop, drum and bass and techno and of different recording conditions or production techniques commonly encountered in modern recordings: use of acoustic or electronic drum kits, reverberation or distorsion effects, equalization and compression. The loop duration ranges from two to fifty seconds.

B_1 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 toms-toms and *perc* for all other percussive instruments. 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). B_2 was semi-automatically annotated by using a SVM classifier trained on B_1 (see [5] for more details about this classifier) - and then by manually correcting the recognition errors.

B_1 and B_2 were finally merged to build up the database used in this work.

2.3. Taxonomy

In theory, $2^n - 1$ combinations are possible by playing simultaneously the instruments from the $n = 8$ basic categories. In our database, after having discarded the combinations occuring less than 40 times, only 18 out of the 255 combinations were observed. The first taxonomy (*detailed taxonomy*) is defined when each stroke is characterized by a distinct label, among the 18 possible combinations. For a better analysis of the results, a simplified taxonomy is also defined: Each segment is annotated with only the most salient instrument, or the two most salient instruments. It is worth precising that the simplified taxonomy is only used to provide an additional interpretation of the results: practically, results for this simplified taxonomy are computed by grouping blocks from the confusion matrix obtained with the detailed taxonomy.

2.4. Segmentation and tempo extraction

The segmentation is obtained by applying an onset detection algorithm based on sub-band decomposition [11]. Concurrently, the overall tempo of the loop is estimated using the algorithm described in [1].

2.5. Features set

The features extracted from the audio signal include:

- **Mean of 13 MFCC** The mean of the Mel Frequency Cepstral Coefficients (MFCC) including c_0, calculated on 20 ms frames with an overlap of 50 % and averaged the coefficients over the stroke duration.

- **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 predefined bands (in Hertz: [10-70] Hz, [70-130] Hz, [130-300] Hz, [300-800] Hz, [800-1500] Hz, [1500-5000] Hz).

To eliminate correlations between some of these 23 parameters, a Principal Component Analysis is performed on the data set. The feature vector used as an input for the classifiers is thus a linear transformation of the features set mentioned above.

2.6. Classifiers

Our first paper [4] presented two classifiers: Hidden Markov Models (HMM) and Support Vector Machines (SVM). HMM took advantage of the short-term time dependencies of drum signals. Considering that the sequence of feature vectors observed is the output of a Hidden markov Model, the transcription task is equivalent to searching the most likely states (strokes) sequence, carried out using the traditional Viterbi algorithm. SVM basically does not take into account time dependencies, but provide very interesting generalization properties. Our article [5] introduced a new model in which time dependencies were taken into account in the SVM model. It consisted practically in replacing the feature vector of one stroke $(f_{1,n}, f_{2,n}, ..., f_{23,n})$ (see section 2.5) by a combined vector containing also the features of the previous stroke $(f_{1,n-1}, f_{2,n-1}, ..., f_{23,n-1}, f_{1,n}, f_{2,n}, ..., f_{23,n})$.

We propose several new improvements to these approaches.

2.6.1. SVM with probabilistic outputs and coupling

Support Vector Machines (see [16] for a detailed presentation) are typically used for discriminating two classes. However, our problem is a multi-class problem, each class being a combination of strokes (for example *bass drum + hi-hat* is one class). A classical implementation of SVM for such multiclass problems uses a *one versus one* approach also known as *pairwise classification* ([12]). Following this approach, $\frac{n(n-1)}{2}$ binary classifiers are trained, each of them discriminating a pair of class. If x is the input vector, (i, j) a pair of classes, (x_{ijk}) (resp. (v_{ijk})) the support vectors (resp. the weights), c_{ij} the parameter of the binary SVM classifier trained to discriminate the classes i and j, the decision function commonly used is:

$$f_{ij}(x) = \sum_k w_{ijk} K(x, x_{ijk}) + c_{ij} \quad (1)$$

$$\Omega_{ij}(x) = \begin{cases} i & \text{if } f_{ij}(x) > 0, \\ j & \text{otherwise} \end{cases} \quad (2)$$

To classify a stroke, the decisions of the $\frac{n(n-1)}{2}$ classifiers are aggregated by a simple vote counting (each Ω_{ij} being a vote).

This approach is not fully satisfying for two reasons. Firstly, vote-counting does not take into account the amount of confidence of each individual decision of the pairwise classifications. Secondly, this method does not provide any kind of probabilistic output: thus, it does not enable post-processing - for example, language modeling, or decision fusion.

Our first improvement consists in replacing the "hard" decision function $\Omega_{ij}(x)$ by a probabilistic one, which can be interpreted as a posterior probability $P_{ij}(class = i|x)$. Platt describes in [15] a method to obtain such posterior probabilities. The output of the SVM $f_{ij}(x)$ is mapped to the interval $]0, 1[$ with a sigmoid function: $D'_{ij}(x) = \frac{1}{1+e^{Af_{ij}(x)+B}}$. The parameters A, B are fit using maximum likelihood estimation on a subset of the training data.

The final decision is taken by coupling the pairwise probabilities given by each classifier, in order to compute a global probability for each class. This coupling is performed with the iterative algorithm presented by Hastie and Tibshirani in [7].

As a result, we obtain a posterior probability $P(class = i|x)$ which can be used for an additional post-processing stage, or for direct classification - in this case, the class that maximizes $P(class = i|x)$ is selected.

2.6.2. SVM with language modeling

N-grams Markov models provide an efficient way of modeling context (short-term) dependencies in drum playing ([4]). In these models, a succession of strokes $S_{k-m}, ..S_k$ is associated to each state q_t. Intuitively, the state q_t represents the stroke S_k in the context of $S_{k-m}..S_{k-1}$ at time t. The model is thus clearly context dependent. The transition probabilities from state i to state j are given by (in the case of 3-grams):

$$\begin{aligned} a_{ij} &= p(q_t = j|q_{t-1} = i) \\ &= p(s_t = S_3|s_{t-1} = S_2, s_{t-2} = S_1) \end{aligned}$$

The transition probabilities a_{ij} can be estimated by counting occurrences of each N-gram in the training database.

Traditionally, such models use mixtures of Gaussian distributions to model the observation probability associated

Taxonomy	Detailed	Simplified
HMM, 3-grams, 2 mixtures	60.5% (4.3%)	79.3%
HMM, 4-grams, 2 mixtures	59.5% (3.5%)	77.7%
SVM	70.6% (2.5%)	86.5%
SVM **prob**	70.7% (2.6%)	86.4%
SVM **ctxt**	72.4% (2.7%)	89.1%
SVM **ctxt prob**	72.6% (2.4%)	89.9%
SVM **prob lang**	75.5% (2.8%)	88.0%

Tab. 1. Drum loop transcription results

to each state. Employing such distributions results in overfitting when a large number of mixtures is used; while a smaller number of mixtures cannot efficiently represent the complex decision surface between classes.

An alternative approach is to use the probabilistic output of our SVM classifier to estimate the probability that a stroke performed at time t corresponds to a given state of the model. The probabilistic information given by the recognition of each individual stroke with the SVM classifier, and the context information obtained with the language model are both taken into account to choose the most likely sequence of strokes. This is done using the classical Viterbi algorithm.

2.7. Results

A 10-fold cross-validation approach was followed. It consists in splitting the whole database in 10 subsets, training the classifier on nine of them, and keeping the last subset for evaluation. The procedure is then iterated by rotating the 10 subsets used for training and testing. The results are summarized in table 1. Standard deviations were computed using the cross-validation variance estimator $\hat{\theta_3}$ presented in [2] and are given in the table. Modified SVM models have the following labels: **ctxt** when contextual features are used, **prob** when probabilistic outputs and coupling are used, **lang** for language modeling (trigrams).

It can be seen that the best results are obtained with the SVM classifiers. The use of probabilistic outputs and coupling does not significantly improve the performances. It can be explained by the fact that our problem involves a rather large number of classes $N = 18$, allowing a good level of accuracy even with a simple voting scheme. Thus, it seems that the use of SVM with probabilistic outputs and coupling is relevant only when the number of classes is smaller, or when the results need to be post-processed.

The use of SVM with a language-modeling stage increases the recognition performances for the detailed taxonomy; but does not give the best results for the simplified taxonomy. A further analysis of recognition errors shows that language modeling allows a more accurate discrimination of simple and compound strokes (especially the presence or absence of hi-hats), but fails to recognize unusual or rare combinations of strokes. For example, *bass drum* and *bass drum + hi-hat* are less likely to be confused, since the language modeling incorporates information about whether or not a hi-hat is played in the sequence; while *rim shot + hi-hat*, which is much less common than *snare drum + hi-hat*, is very likely to be classified as this first stroke.

Instrument	Onomatopoeia
Bass drum	[pum] / [bum]
Cymbal, hi hat	[ti] / [ts]
Snare drum,	[tʃa]
Snare drum + Bass drum mixture	[ta]
Tom, other percussive instrument	[do] / [dɔm] / [tɔm]

Tab. 2. Language used for spoken queries

3. RECOGNITION OF ONOMATOPOEIA IN SPOKEN QUERIES

3.1. Onomatopeia set

While several rhythmic instruments such as North Indian Tabla have a well-defined set of onomatopoeia (known as *bols* in the case of Tabla) denoting each stroke of the instrument, there is no commonly accepted set of vocables to denote the instruments of the drum kit. This can be explained by the fact that notation plays a more important role than oral tradition in the transmission and teaching of Western popular music.

A possible approach, used by Kapur et al. in their *Bionic BeatBox Voice Processor* [10], is to let the users freely use their own set of onomatopoeia, after having trained the system by providing a few examples of each vocable.

We followed a different approach in which we imposed a set of onomatopoeia to the user. The set chosen for our work is given in the table 2. It has been validated by a perception experiment ([5]) which consisted in randomly playing a drum stroke, and in asking the subjects to pick the onomatopoeia that best described it.

3.2. Recognition of spoken onomatopoeia

In order to train and evaluate the recognition of spoken onomatopoeia in a speaker-independent way, a new database was recorded from 13 speakers, 11 males and 2 females. Most of these speakers practice music regularly, 2 of them practicing electronic music and DJing. The database was recorded according to the following protocol: During an introductory stage, the subject was presented the different instruments of the drum kit and the vocabulary used. During a

first recording stage, a computer animation displayed a random sequence of onomatopoeia, and the subject was asked to pronounce each onomatopoeia as soon as it flashed on the screen. During a second stage, the subject was asked to "perform" or "beatbox" four simple sequences. The voices were recorded using a Shure WH20 headworn directional microphone on an Edirol UA-5 soundcard, at 44.1 kHz.

This corpus was manually segmented and annotated. The annotation includes onomatopoeia ([pum], [ta]...), silences, and a last category for miscellaneous events such as breathes or pops. The entire database contains 1057 utterances.

Training, recognition and evaluation was performed using the HTK Speech Recognition Toolkit. The features used for the recognition are the 13 MFCC + 13 ΔMFCC + 13 $\Delta\Delta$MFCC. Each onomatopoeia is represented by a Bakis (left-right) HMM model with 3 states, at the exception of the silence model which uses 4 states and a different topology. The probability distribution associated to each state is a mixture of 3 gaussians - using a higher number of mixtures resulted in overfitting. These HMM models are trained for each onomatopoeia using the EM algorithm. Given a simple "task grammar" to model the succession of silences and vocal activity (onomatopoeia), all the models were connected to form a network, on which the recognition is performed with the Viterbi algorithm. The output of this query transcription system is a sequence of pairs (t_i, S_i), where S_i is the stroke (or compound stroke, like *bass drum + snare drum*) played at time t_i. This output is post-processed by removing the silence labels, the onomatopoeia shorter than 100ms, and by replacing the recognized onomatopoeia by the rythmic instrument it represents - for example [pum] is replaced by *bass drum*.

3.3. Evaluation

This query recognition system was evaluated using a leave-one-speaker-out validation protocol. This protocol consists in dividing the anotated corpus in $K = 13$ subsets, each subset containing the utterances of a given speaker. The recognition model is trained on $K - 1$ of them, and the last subset is used for evaluation. By rotating each subset, the data recorded for each speaker is used $K-1$ times for training, and once for evaluation.

Once a transcription output was obtained for each of the original utterances, these transcriptions were analyzed and compared to the reference transcriptions. More precisely, the original and output transcriptions were matched using a dynamic programming algorithm. A label insertion or deletion carry a score of 3.3, a label substitution carries a score of 4. The label alignement with the lowest score is found, and the number of substitution (S), insertion (I), deletion (D) errors is counted. Then, the accuracy of the transcription for a total of N onomatopoeia is given by:

$$Accuracy = \frac{N - S - I - D}{N}$$

The accuracy of our speaker independent system is 84.4%.

4. QUERY SCORING AND ALIGNING

4.1. Statistical modeling of interpretation errors

Query by humming systems often use approaches based on string matching. These approaches are not suitable for the scoring of drum queries, for two reason. Firstly, the notion of melody and melodic contour is not relevant when dealing with drum loops. Secondly, most of these approaches are ignoring the rythmic information and only focus on the intervals between notes - a criterion which cannot be defined for drum sounds. On the other hand, tempo or beat histogram features are not sufficient to accurately represent the rythmic information - for example the way snare drums and bass drums are played on downbeats and upbeats.

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 a performance with real drums instruments of the interpretation given by the spoken onomatopoeia". This model takes into account the various editing operations likely to occur when a complex rhythmic phrase is interpreted with onomatopoeia: 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 (*substitutions*) of a note contained in the searched loop, possibly with timing errors (*alignment*). It allows the computation of the probability that a query is actually a good formulation of one of the loops contained in the database, in other words the likelihood of the interpretation q knowing the loop l. The sequence of editing operations e made by the user when performing the searched loop is considered as a hidden variable:

$$P(q|l) = \sum_e P(q, e|l)$$

Our model is described in details in [5]. It is parametrized by the likelihood of the interpretation of each drum sound b, knowing that it is not present in the loop $P(\{b\}|\emptyset)$ (insertion of strokes not present in the original loop), the likelihood of the deletion of each drum sound a, knowing that it is present in the loop $P(\emptyset|\{a\})$ (non-formulation), a probability 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, and a distribution for the duration of deleted (resp. inserted) strokes $P_d(t)$ (resp. $P_i(t)$). These parameters can be empirically chosen to reflect common mistakes made when vocally performing a rhythm (such as ignoring

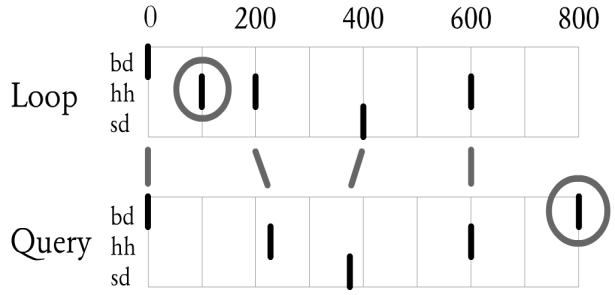

Fig. 2. Interpretation of a loop

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

Tab. 3. Corresponding editing operations

hi-hats, or snare drum flams), or learned by gathering statistics from original drumloops and their vocal interpretations.

We define $P((t,B)|(u,A))$ as the likelihood that a combination of strokes B at time t is the interpretation of a combination of strokes A occuring at time u. If we consider that time-aligning errors are independent of the confusions between strokes, it can be expressed as: $P((t,B)|(u,A)) = P(B|A)P_a(|t-u|)$, where $P_a(|t-u|)$ is the likelihood of a timing error equal to $|t-u|$ between two events. Using the same notations, $P((t,B)|\emptyset)$ is the likelihood of an insertion of a stroke B, and $P(\emptyset|(u,A))$ is the likelihood of the deletion of a stroke of duration d.

Finally:

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

where the sequences $(e_{Qi})_{i\in[1,E]}$ and $(e_{Li})_{i\in[1,E]}$ describe the alignment resulting from the editing operations e on the loop L and the vocal query Q (refer to figure 2 for an example of interpretation, and the corresponding values of e in 4.1).

The aim of the alignment between the loop and the interpretation is to find the sequence of edit operations e^* maximizing the likelihood of $P(q,e^*|l)$. The search of such an optimal alignment is possible with dynamic programming, and can be efficiently implemented by computing log-likelihoods rather than likelihoods.

4.2. Tempo and loop start alignment

In the maximization computed previously, we assumed that the query was an interpretation of the whole loop. However, it is likely that the query is just an interpretation of a short fragment located at any time offset within the loop. This problem is solved by searching the optimal alignement for a range of time offset and loop durations.

Finally, it is also necessary to deal with the fact that the query is not always formulated at the same tempo as the searched loop. In our previous approach, an optimal alignement was searched for a discreet set of tempo scaling factors, and it resulted in a *tempo independent distance*. The distance used in this article is slightly different since it also incorporates a penalty on the tempo difference: $D = D_{\text{tempo independent}} + C|\log \textit{tempo scaling}|$. The parameter C can be modified to find a trade-off between a tempo independent search based only on the contents of the loop, and a tempo-dependent search that will emphasize on the absolute time structure of the rhythm rather than on its contents.

4.3. Query and comparison

For a query d, given a threshold τ, the matching candidates are:

$$C(\tau,q) = \{L, D(q,L) < \tau\}$$

A model similar to this one can be used to compare two loops from the database. The likelihoods $P(l_1|l_2)$ expressing the substitution cost between two strokes have been symmetrised so that the measure D provided by the recursion can be interpreted as a distance. Not only this allows the grouping of results by similarity, by it also allows query by example - in the case, the example playing the role of the vocal query.

4.4. Evaluation

In order to evalute the query system, the following procedure was iterated $N = 500$ times:

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 8 seconds.
3. A query was synthesized by concatenating onomatopoeia contained in a test database (compound of 80 instances of each of the onomatopoeia). This query contains time alignment mistakes, substitutions, deletions and insertions.
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.

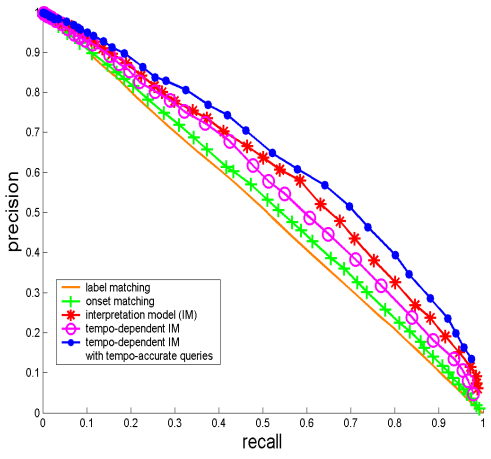

Fig. 3. Precision / Recall curves

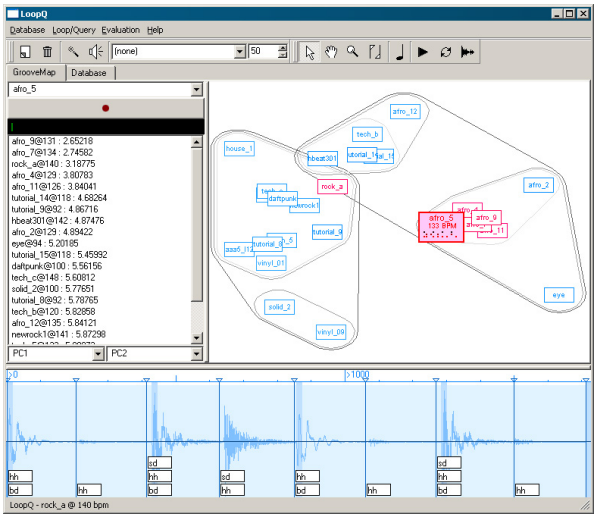

Fig. 4. User interface of the **LoopQ** application

We used the traditional information retrieval performance measures: precision and recall. For each value of the threshold τ, a pair of precision/recall values can be computed by averaging the precision/recall ratios of each single query. Since in our case only one loop is to be retrieved, 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. The precision of a single query is 0 if the loop searched is not present in the matches; $1/N$ where N is the number of matches otherwise.

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

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

Several sets of results were obtained, from which precision/recall curves were plotted (figure 3). A first set was obtained using a simple string matching algorithm, that is to say, only the contents of the loop was considered, without regard to the temporal information (label matching). Reversely, the second set was obtained using a distance D taking into account only relative temporal information (onset matching). The third set was obtained with the distance used in our previous work (interpretation model). The fourth set was obtained with a distance taking into account both the rhythmic contents and the tempo information. Finally, the fifth set was obtained using the same protocol and distance as previously, except that the queries were performed at exactly the same tempo as the searched loop.

It can be clearly observed that our interpretation model outperforms label or onset matching approaches. Incorporating tempo information can also improve the overall performance of the retrieval system, provided the queries are formulated at the exact tempo - a condition that can be reasonably satisfied if a click track is played in the background when the user records a query.

5. IMPLEMENTATION

All the modules presented in this paper are integrated in a graphical application, **LoopQ**, developed in C++ with the Qt library. Users can submit vocal queries by clicking the *record* button. The vocal input is subsequently recognized, displayed in the bottom of the screen with tags corresponding to the recognized onomatopoeia, and submitted as a query. At this stage, it is also possible to generate a synthetic drum loop by replacing each onomatopoeia by the corresponding drum sample.

The loops matching the queries are displayed on the left pane, sorted by similarity. The right pane displays the 25 best candidates in a 2D plane. Several axis can be selected to visually group the results: tempo, complexity (number of drum events per second), density (number of drum events per bar), and the 3 first axis obtained by multi-dimensional scaling (MDS) of the resulting data set - using the similarity measure. By default, the first axis obtained by MDS are selected, allowing a visual grouping of similar loops. Each loop is represented by a box containing its name.

Different kind of interactions are possible with this representation. Moving the mouse cursor on a box zooms it, and displays additional information about the loop, such as its tempo and a transcription of its first bar. Clicking on a box plays the corresponding loop. Right-clicking performs a query, using the pointed loop as an example. This allows the user to perform incremental searches and navigate in the database the same way one would follow hyperlinks on

the World Wide Web. An additional interaction mode, the *Jam* mode, specific to DJing uses, allows a continual sound feedback: whenever the mouse cursor hovers over a box, the corresponding loop is continuously played, until another loop is pointed.

6. CONCLUSION AND FUTURE WORK

Content-based indexing and querying systems are necessary to assist composers and DJs, who use large collections of sound files daily. This paper presented an innovative system for indexing and querying drum loops, and its recent improvements. New SVM classifiers, and hybrid approaches using HMM and SVM were experimented, on a larger database, resulting in a 75.5% correct recognition rate for the drum loop transcription task with a detailed taxonomy. Better results could be achieved by using more complex language models than the trigram Markov models presented here - for example by taking into account the cyclic and repetitive characteristics of rhythmic sequences, or by making a better use of time and duration information.

A speaker-independent onomatopoeia recognition front-end has been successfully integrated and gives a 84.4% accuracy. At this stage, further usability experiments should be conducted with drummers and DJs, to evaluate how this recognition front-end deals with the different onomatopoeia used. It is very likely that each drummer or DJ uses his own vocabulary. However, this does not invalidate our intuition that vocal input is one of the most efficient modality to specify rhythmic queries.

Finally, further works will focus on the detection on drum events in polyphonic music signals - our goal being to index not only drum loops, but also the drum tracks of entire songs.

7. REFERENCES

[1] M. Alonso, B. David, and G. Richard. A study of tempo tracking algorithms from polyphonic music signals. In *Proceddings of 4th COST276 Workshop, Bordeaux, France*, march 2003.

[2] Y. Bengio and Y. Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. CIRANO Working Papers 2003s-22, CIRANO, May 2003. available at http://ideas.repec.org/p/cir/cirwor/2003s-22.html.

[3] A. Ghias, J.Logan, D. Chamberlin, and B.C. Smith. Query by humming: Musical information retrieval in au audio database. In *Proceedings of ACM Multimedia'95*, 1995.

[4] O. Gillet and G. Richard. Automatic transcription of drum loops. In *Proceedings of the IEEE ICASSP 2004 Conference*, May 2004.

[5] O. Gillet and G. Richard. Drum loops retrieval from spoken queries. In *Journal of Intelligent Information Systems*, To be published 2005.

[6] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. Rwc music database: Popular, classical, and jazz music databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pages 287–288, October 2002.

[7] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *Advances in Neural Information Processing Systems*, volume 10, 1998.

[8] P. Herrera, X. Amatriain, E. Battle, and X. Serra. Towards instrument segmentation for music content description: a critical review of instrument classification techniques. In *Proceedings of ISMIR2000*, 2000.

[9] P. Herrera, A. Dehamel, and F. Gouyon. Automatic labeling of unpitched percussion sounds. In *Proceedings of the 114th AES convention*, March 2003.

[10] A. Kapur, M. Benning, and G. Tzanetakis. Query by beatboxing: Music information retrieval for the dj. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, October 2004.

[11] A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1999.

[12] U. H.-G. Kressel. Pairwise classification and support vector machines. In *Advances in kernel methods: support vector learning*, pages 255–268. MIT Press, 1999.

[13] R.J. McNab, L.A. Smith, D. Bainbridge, and I.H. Witten. The new zealand digital library melody index. In *D-Lib Magazine*, 1997.

[14] T. Nakano, J. Ogata, M. Goto, and Y. Hiraga. A drum pattern retrieval method by voice percussion. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, October 2004.

[15] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classiers*, pages 61–74, 2000.

[16] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.